User Guide

Data Pipeline R-5.0

Contents

1. Introduction

1.1. Introducing Data Pipeline

The Data Pipeline is an Enterprise data orchestration and transformation tool, that allows you to seamlessly design and deploy your DataOPs or MLOps workflow. It can handle both Streaming and batch data seamlessly. The Data pipeline offers an extensive list of data processing components that help you automate the entire data workflow, Ingestion, transformations, and running AI/ML models.

In Data pipeline we treat data as events, Data Processing components can listen to events, as data hits those events, the process kick starts automatically, these processes then publish the output to another event. This allows data engineers to chain the process and build large data flows.

1.2. Overview

This guide covers:

- Features and menus provided on the Data Pipeline homepage
- Steps to create and manage Pipeline
- Create and manage various pipeline components

1.3. Target Audience

The document is targeted to the following audience:

- Data Engineers
- Data Scientists
- ML-Ops Engineers

2. Supported Web Browsers

The BDB Platform is a web browser-based application. The users can run the BDB Platform and its various plugins on the below given versions of the browsers:

Google Chrome	Latest Version (recommended web browser)
Mozilla Firefox/ Firefox ESR	Latest Version
Microsoft Edge	Latest Version
Apple Safari	10

The supported browser versions are driven by the capabilities the UI employs and the dependencies it uses. UI features will be developed and tested against the supported browsers.

2.1. Unsupported Browsers

While the UI may run successfully in unsupported browsers, it is not actively tested against them. Additionally, the UI is designed as a desktop experience and is not currently supported in mobile browsers.

3. Terminology

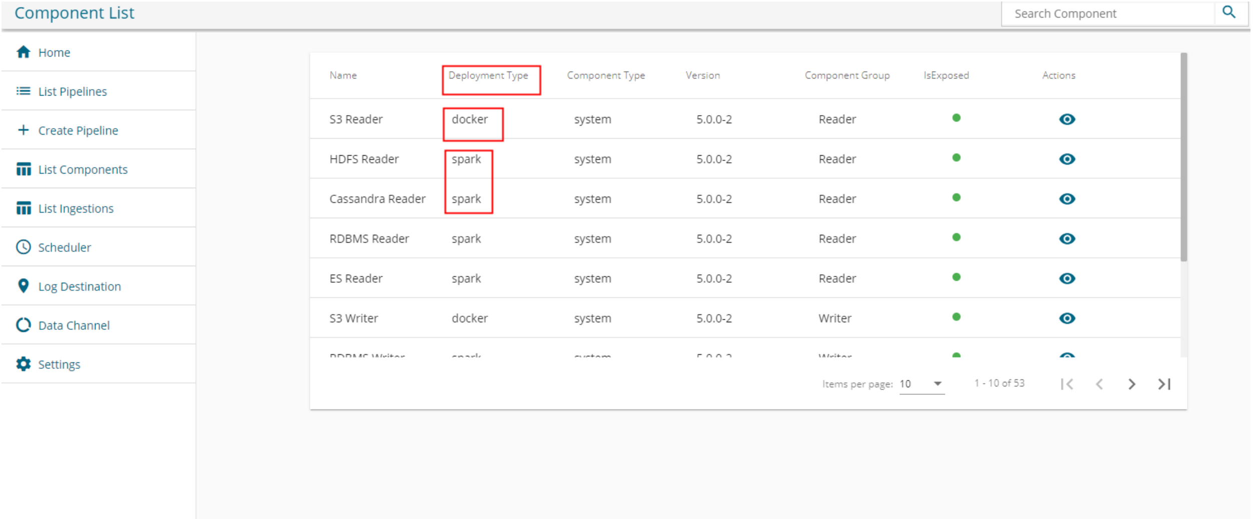
Component Deployment: Component Deployment means the Deployment on Kubernetes.

Components (Deployment Type): There are two types of component categories in context to deployment-type (Spark and Docker).

Spark: Drivers and Executers.

When these component gets deployed, we get two pods one is for the component driver and another one is the executor. The number of executors can be increased according to the processing load or the throughput required by the user.

Docker: a single pod gets deployed in this case.



Name	Deployment Type	Component Type	Version	Component Group	IsExposed	Actions
S3 Reader	docker	system	5.0.0-2	Reader	●	👁
HDFS Reader	spark	system	5.0.0-2	Reader	●	👁
Cassandra Reader	spark	system	5.0.0-2	Reader	●	👁
RDBMS Reader	spark	system	5.0.0-2	Reader	●	👁
ES Reader	spark	system	5.0.0-2	Reader	●	👁
S3 Writer	docker	system	5.0.0-2	Writer	●	👁
RDBMS Writer	spark	system	5.0.0-2	Writer	●	👁

Items per page: 10 1 - 10 of 53

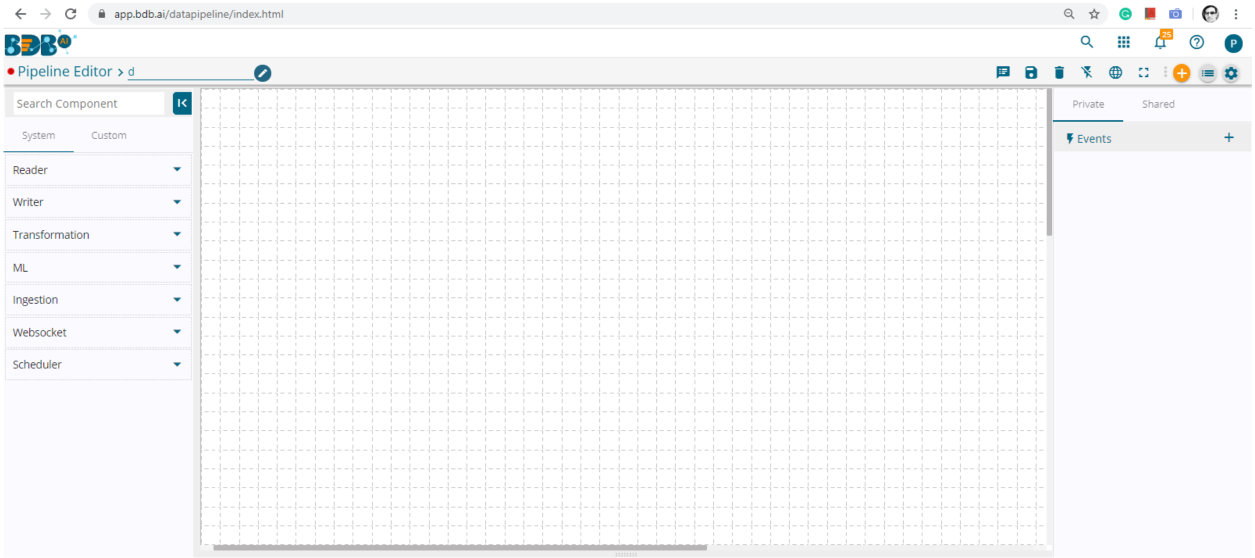
Base Components

These are the components that are responsible for the backend functionality of the pipeline.
Each of these components' Pod should be UP and Running for the smooth functioning of the pipeline module.
Base components are as given below:

- event-trigger
- log manager
- pipeline-scheduler
- task invoker
- task manager
- webservice
- ws-producer
- pipeline-scheduler

Workflow Editor

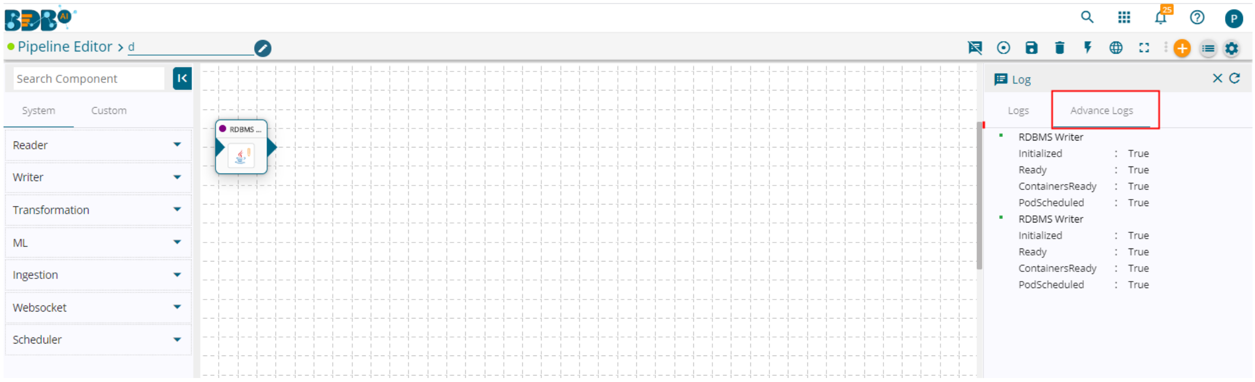
The canvas where we create/develop the pipeline is the workflow-editor.



Logs

Advanced Logs: when someone refers to advanced logs, that person is asking to monitor whether the components on the workflow-editor are deployed or not.

Note: This tab appears when the pipeline is active inside the log panel.



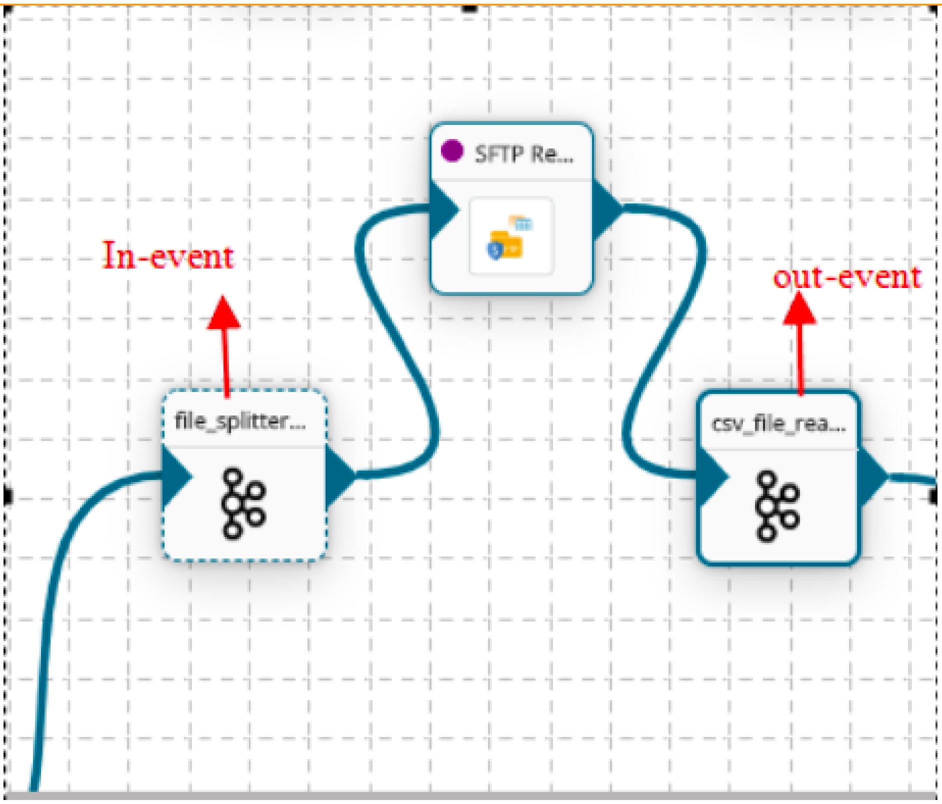
```
RDBMS Writer
Initialized       :  True
Ready             :  True
ContainersReady   :  True
PodScheduled      :  True
RDBMS Writer
Initialized       :  True
Ready             :  True
ContainersReady   :  True
PodScheduled      :  True
```

Event

An event, in general, is referred to as the Kafka topic that works as a communication channel between two components.

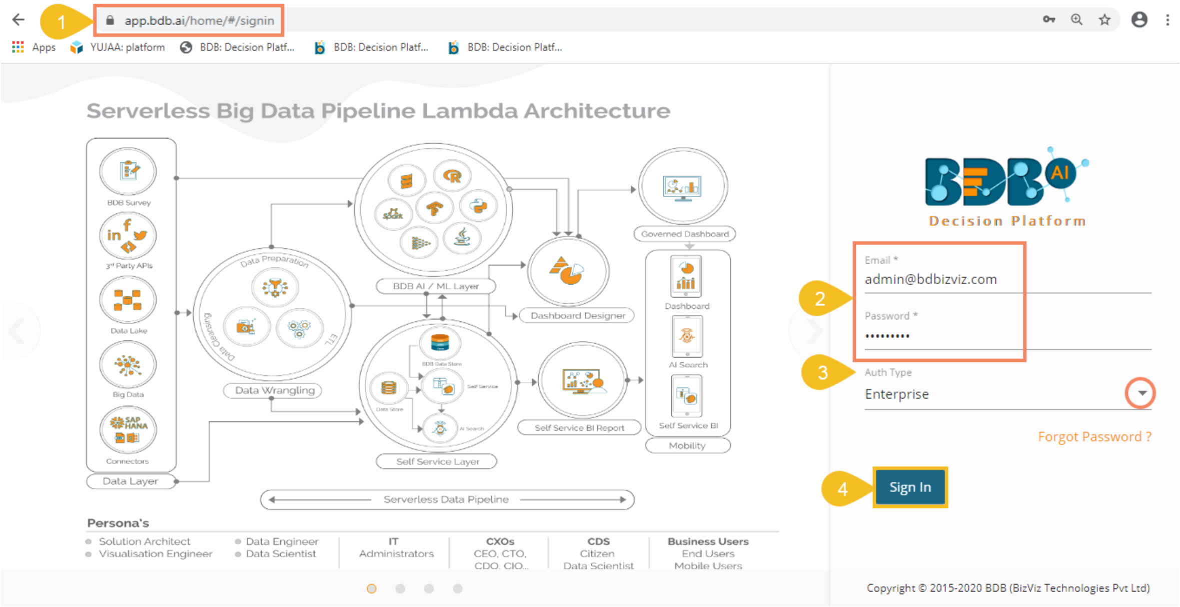
In-event: the previous event/ Kafka-topic. This gives input to the component.

Out-event: the event after the component is out event. This is where the output of the component is present.



4. Logging In

i) Browse the URL link: https://app.bdb.ai to open the Sign In page.
ii) Enter user-specific credentials (Email ID with valid Password).
iii) Select an authentication type using the drop-down.
iv) Click the ‘Sign In’ option.



1 app.bdb.ai/home/#/signin

Serverless Big Data Pipeline Lambda Architecture

Decision Platform

2 Email * admin@bdbizviz.com

Password * •••••••••

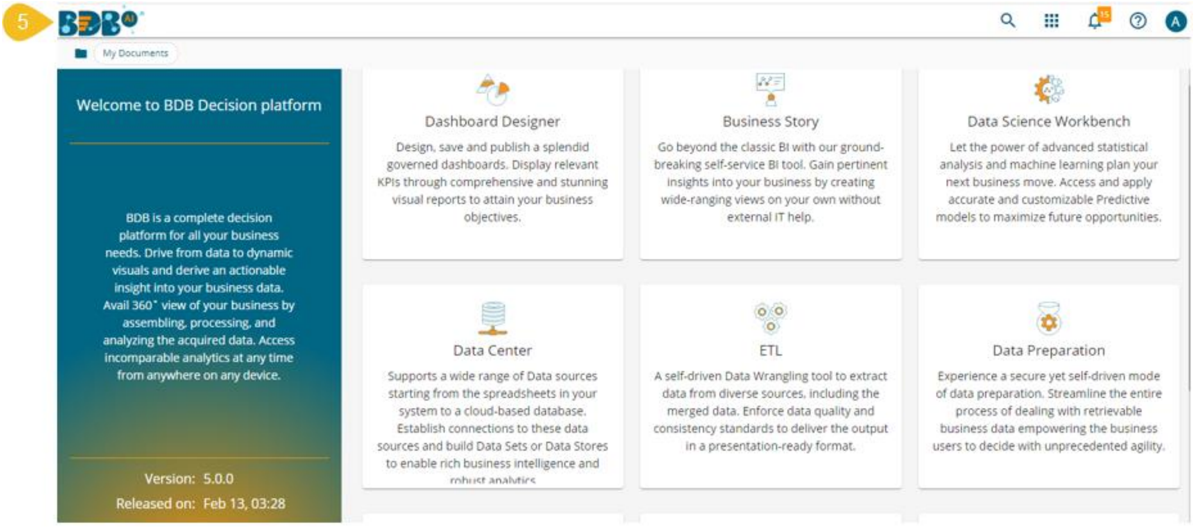
3 Auth Type Enterprise

Forgot Password ?

4 Sign In

Copyright © 2015-2020 BDB (BizViz Technologies Pvt Ltd)

v) The Platform homepage opens.



Welcome to BDB Decision platform

BDB is a complete decision platform for all your business needs. Drive from data to dynamic visuals and derive an actionable insight into your business data. Avail 360˚ view of your business by assembling, processing, and analyzing the acquired data. Access incomparable analytics at any time from anywhere on any device.

Version: 5.0.0

Released on: Feb 13, 03:28

Dashboard Designer

Design, save and publish a splendid governed dashboards. Display relevant KPIs through comprehensive and stunning visual reports to attain your business objectives.

Business Story

Go beyond the classic BI with our ground-breaking self-service BI tool. Gain pertinent insights into your business by creating wide-ranging views on your own without external IT help.

Data Science Workbench

Let the power of advanced statistical analysis and machine learning plan your next business move. Access and apply accurate and customizable Predictive models to maximize future opportunities.

Data Center

Supports a wide range of Data sources starting from the spreadsheets in your system to a cloud-based database. Establish connections to these data sources and build Data Sets or Data Stores to enable rich business intelligence and robust analytics

ETL

A self-driven Data Wrangling tool to extract data from diverse sources, including the merged data. Enforce data quality and consistency standards to deliver the output in a presentation-ready format.

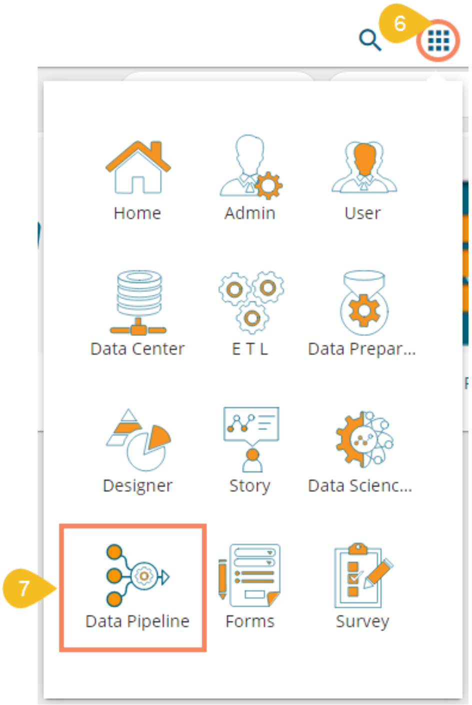
Data Preparation

Experience a secure yet self-driven mode of data preparation. Streamline the entire process of dealing with retrievable business data empowering the business users to decide with unprecedented agility.

Note: Make sure that the user has permission to access the Data Pipeline plugin.

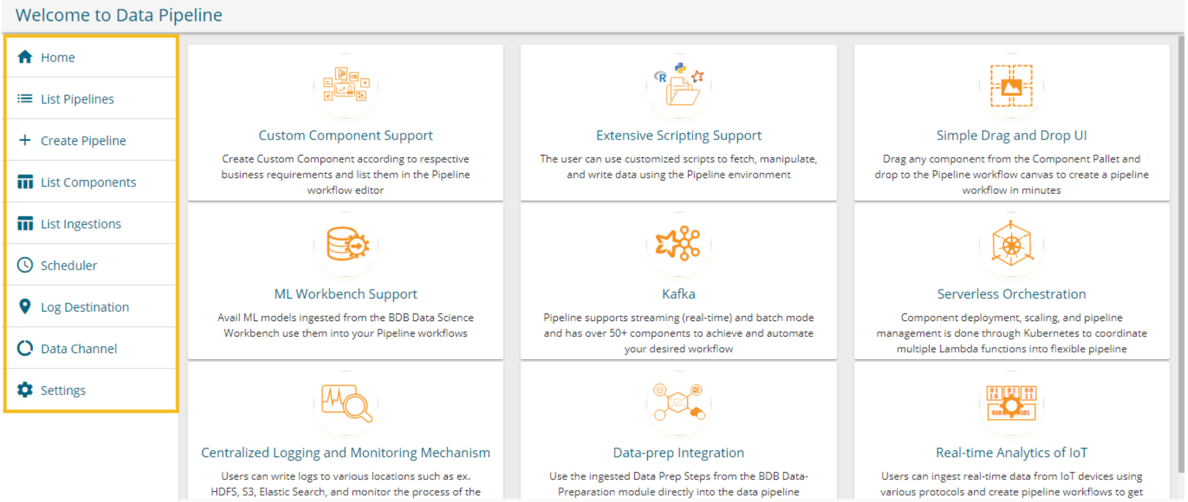
vi) Click the Apps menu.

vii) Select the Data Pipeline Plugin.



5. Homepage

Data Pipeline homepage opens displaying a menu panel on the left side of the screen as highlighted in the image.



Custom Component Support
Create Custom Component according to respective business requirements and list them in the Pipeline workflow editor

Extensive Scripting Support
The user can use customized scripts to fetch, manipulate, and write data using the Pipeline environment

Simple Drag and Drop UI
Drag any component from the Component Pallet and drop to the Pipeline workflow canvas to create a pipeline workflow in minutes

ML Workbench Support
Avail ML models ingested from the BDB Data Science Workbench use them into your Pipeline workflows

Kafka
Pipeline supports streaming (real-time) and batch mode and has over 50+ components to achieve and automate your desired workflow

Serverless Orchestration
Component deployment, scaling, and pipeline management is done through Kubernetes to coordinate multiple Lambda functions into flexible pipeline

Centralized Logging and Monitoring Mechanism
Users can write logs to various locations such as ex. HDFS, S3, Elastic Search, and monitor the process of the

Data-prep Integration
Use the ingested Data Prep Steps from the BDB Data-Preparation module directly into the data pipeline

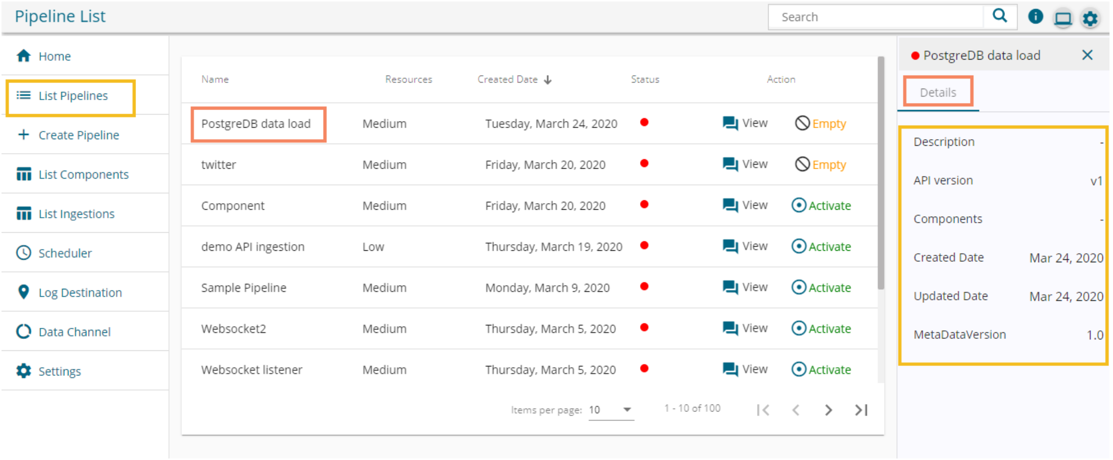
Real-time Analytics of IoT
Users can ingest real-time data from IoT devices using various protocols and create pipeline workflows to get

The user can access the various options from the Data Pipeline Homepage to work upon the data flow.

Homepage Panel

1. List Pipeline

The ‘List Pipelines’ option opens the available Pipeline List for the logged-in user. All the saved pipelines by a user get listed on this page. The ‘Details’ tab on the right side of the page displays the basic details of the selected pipeline.



Name	Resources	Created Date ↓	Status	Action	
PostgreDB data load	Medium	Tuesday, March 24, 2020	●	View	Empty
twitter	Medium	Friday, March 20, 2020	●	View	Empty
Component	Medium	Friday, March 20, 2020	●	View	Activate
demo API ingestion	Low	Thursday, March 19, 2020	●	View	Activate
Sample Pipeline	Medium	Monday, March 9, 2020	●	View	Activate
Websocket2	Medium	Thursday, March 5, 2020	●	View	Activate
Websocket listener	Medium	Thursday, March 5, 2020	●	View	Activate

Items per page: 10 1 - 10 of 100

● PostgreDB data load ✕

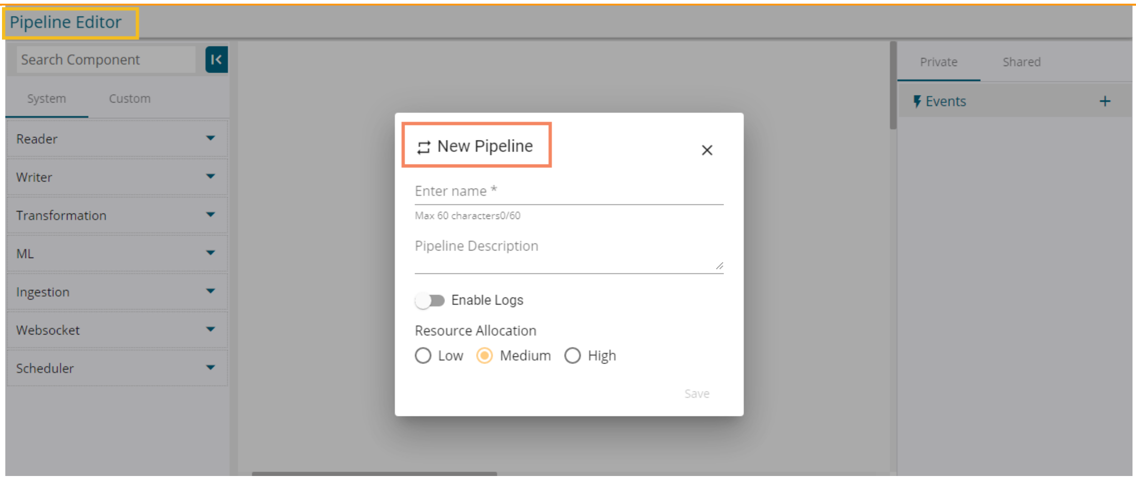
Details

Description	-
API version	v1
Components	-
Created Date	Mar 24, 2020
Updated Date	Mar 24, 2020
MetaDataVersion	1.0

Note: The user can open the pipeline editor for the selected pipeline from the list by clicking the ‘View’ option.

2. Create a Pipeline

The ‘Create Pipeline’ option redirects the user to create a new pipeline and eventually to the Pipeline Editor (the user interface) to create, update, activate, and save the Pipeline Workflow.



New Pipeline

Enter name *

Max 60 characters0/60

Pipeline Description

Enable Logs

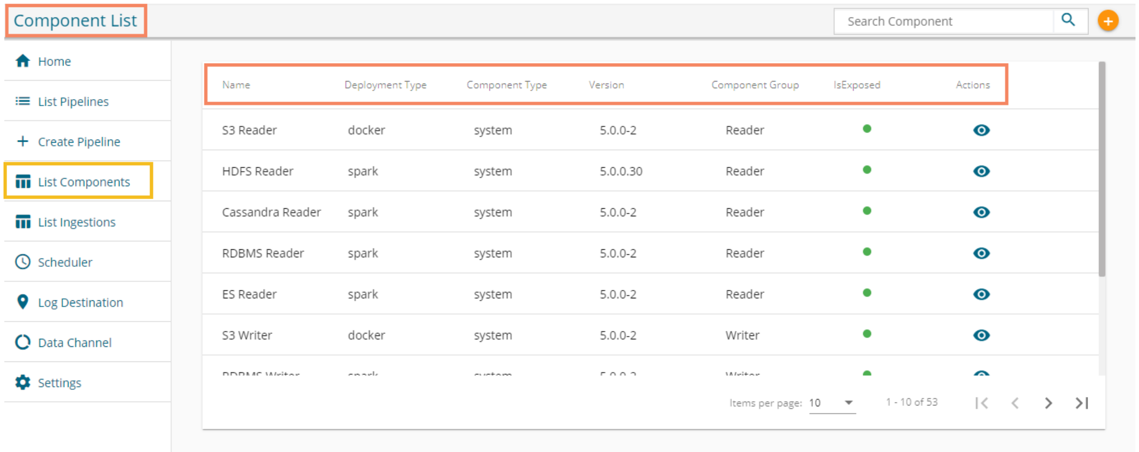
Resource Allocation

○ Low ● Medium ○ High

Note: The stepwise process to create a Pipeline workflow is described later in this document.

3. List Components

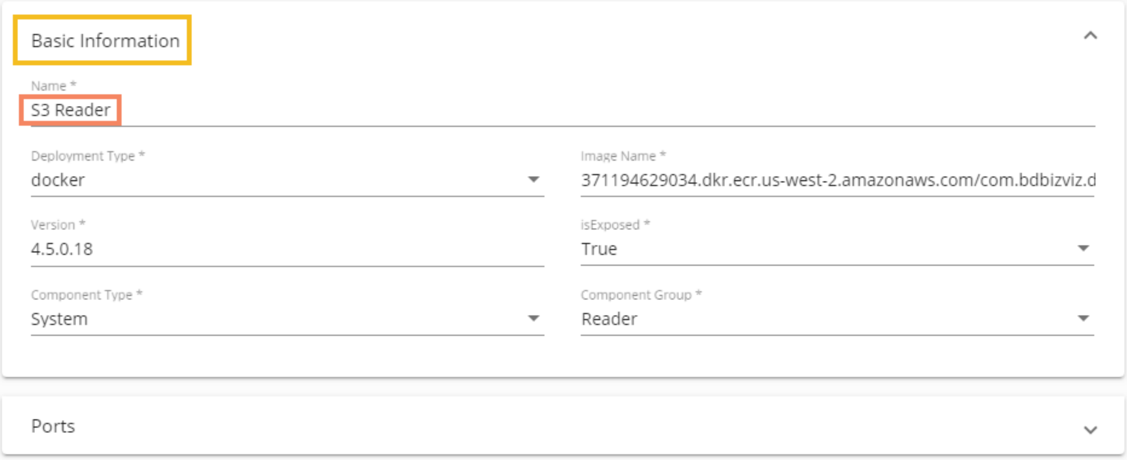
All the available component names are listed on this page. The user gets to know about the Deployment Type, Component Type, Version, Component Group about a component. The Expose status is indicated through the green color coding if the related component is exposed.



Name	Deployment Type	Component Type	Version	Component Group	IsExposed	Actions
S3 Reader	docker	system	5.0.0-2	Reader	●	👁
HDFS Reader	spark	system	5.0.0.30	Reader	●	👁
Cassandra Reader	spark	system	5.0.0-2	Reader	●	👁
RDBMS Reader	spark	system	5.0.0-2	Reader	●	👁
ES Reader	spark	system	5.0.0-2	Reader	●	👁
S3 Writer	docker	system	5.0.0-2	Writer	●	👁
RDBMS Writer	spark	system	5.0.0-2	Writer	●	👁

Items per page: 10 1 - 10 of 53

The user can get the Basic Information about the selected component by clicking the 'Actions' icon. The basic information can be edited from this window if needed. The user gets Ports and Spark Component Information if the selected Deployment Type option is 'Spark'.



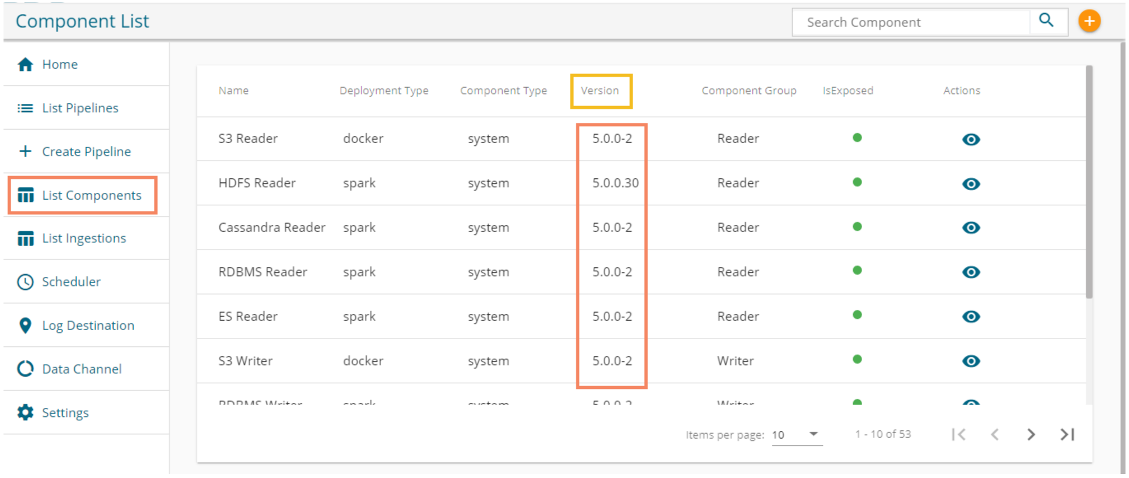
Basic Information

Name *
S3 Reader

Deployment Type * docker	Image Name * 371194629034.dkr.ecr.us-west-2.amazonaws.com/com.bdbizviz.d
Version * 4.5.0.18	isExposed * True
Component Type * System	Component Group * Reader

Ports

The user gets the Component version information from the List Component page. The version of the components get updated with each new release of the respected components.

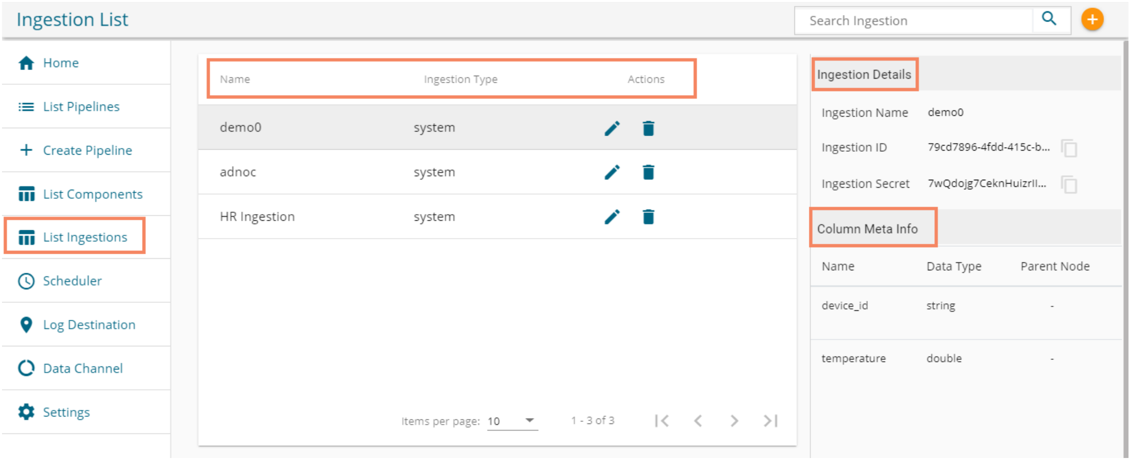


Name	Deployment Type	Component Type	Version	Component Group	IsExposed	Actions
S3 Reader	docker	system	5.0.0-2	Reader	●	👁
HDFS Reader	spark	system	5.0.0.30	Reader	●	👁
Cassandra Reader	spark	system	5.0.0-2	Reader	●	👁
RDBMS Reader	spark	system	5.0.0-2	Reader	●	👁
ES Reader	spark	system	5.0.0-2	Reader	●	👁
S3 Writer	docker	system	5.0.0-2	Writer	●	👁
RDBMS Writer	spark	system	5.0.0-2	Writer	●	👁

Items per page: 10 1 - 10 of 53

4. List Ingestions

This page displays Ingestion List containing the created API ingestions to consume data into the data pipeline. This can be used in the meta-info of the API Server Ingestion component.



Name	Ingestion Type	Actions
demo0	system	✎ 🗑
adnoc	system	✎ 🗑
HR Ingestion	system	✎ 🗑

Items per page: 10 1 - 3 of 3

Ingestion Details

Ingestion Name demo0

Ingestion ID 79cd7896-4fdd-415c-b...

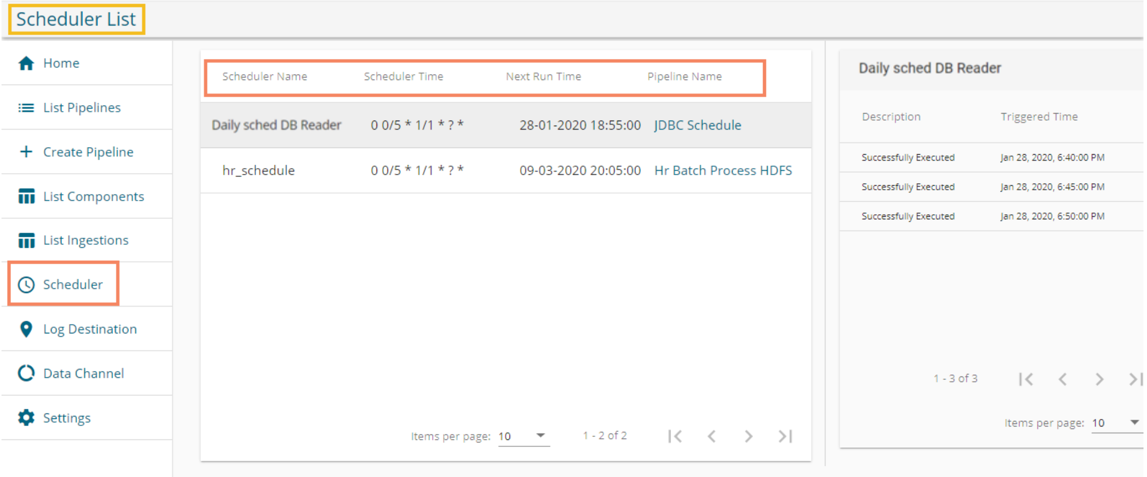
Ingestion Secret 7wQdojg7CeknHuizrIl...

Column Meta Info

Name	Data Type	Parent Node
device_id	string	-
temperature	double	-

5. Scheduler

It opens the Scheduler List with the status and Triggered Time of the scheduled pipeline components mentioned in the right-side panel.



Scheduler List

Scheduler Name	Scheduler Time	Next Run Time	Pipeline Name
Daily sched DB Reader	0 0/5 * 1/1 * ? *	28-01-2020 18:55:00	JDBC Schedule
hr_schedule	0 0/5 * 1/1 * ? *	09-03-2020 20:05:00	Hr Batch Process HDFS

Items per page: 10 1 - 2 of 2

Daily sched DB Reader

Description	Triggered Time
Successfully Executed	Jan 28, 2020, 6:40:00 PM
Successfully Executed	Jan 28, 2020, 6:45:00 PM
Successfully Executed	Jan 28, 2020, 6:50:00 PM

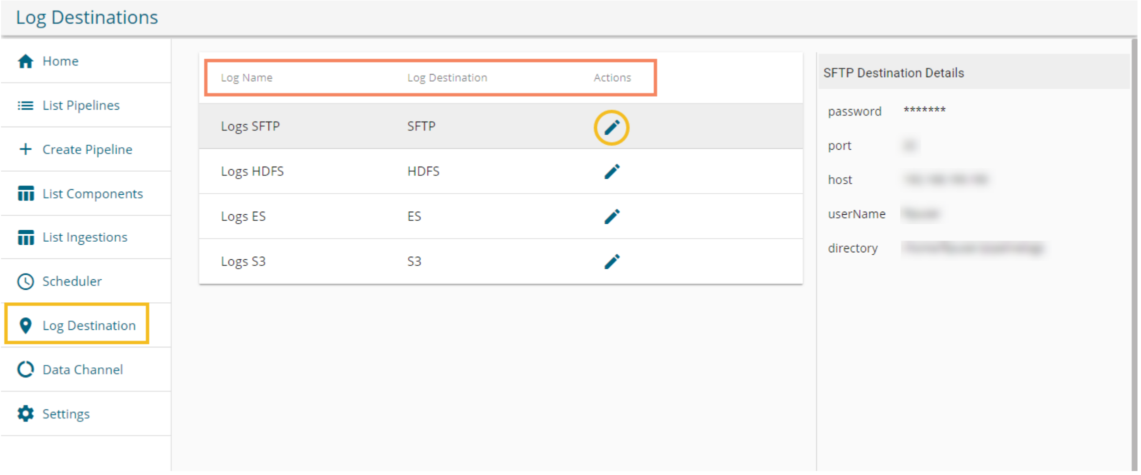
1 - 3 of 3

Items per page: 10

Note: The Scheduler Time column displays the time as Cron expressions.

6. Log Destination

The Log Destination page lists Log Name and Log Destination details. The user can edit the log destination details as per their preference, choice, and availability of the respective resources. The user needs to provide proper configuration details to edit the log destination information.



Log Destinations

Log Name	Log Destination	Actions
Logs SFTP	SFTP	✎
Logs HDFS	HDFS	✎
Logs ES	ES	✎
Logs S3	S3	✎

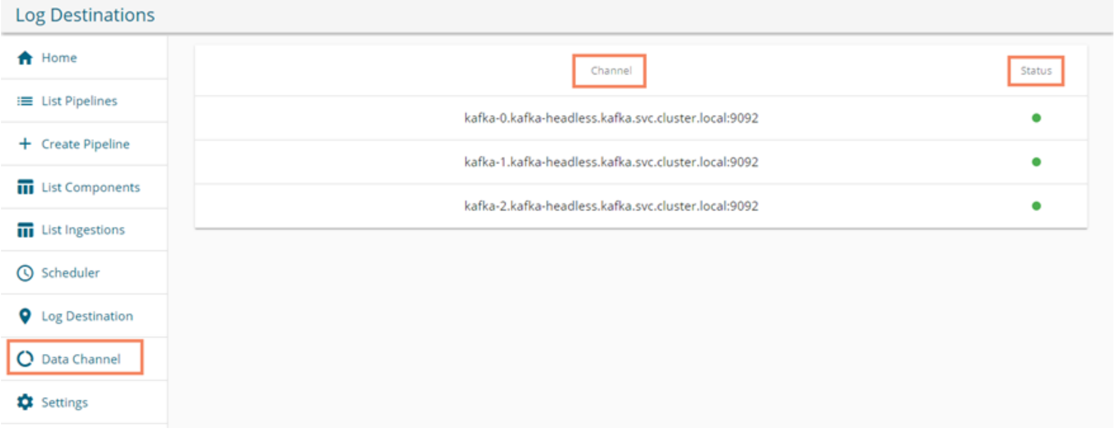
SFTP Destination Details

password	*******
port	[redacted]
host	[redacted]
userName	[redacted]
directory	[redacted]

Note: Logs for all the pipelines get written at the configured location. This is like a master configuration for logs destinations.

7. Data Channel

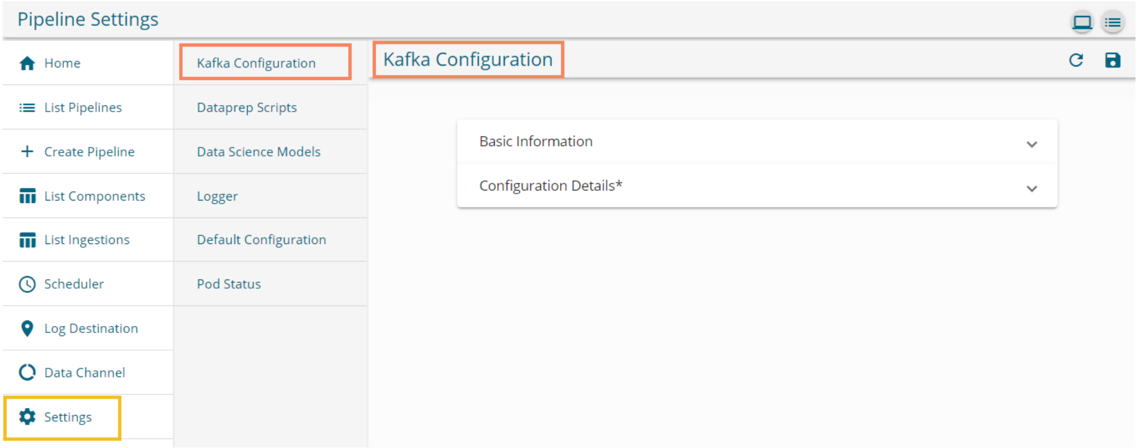
The Data Channel window displays the status of the Kafka Server.



Channel	Status
kafka-0.kafka-headless.kafka.svc.cluster.local:9092	●
kafka-1.kafka-headless.kafka.svc.cluster.local:9092	●
kafka-2.kafka-headless.kafka.svc.cluster.local:9092	●

8. Settings

The user gets settings information about various pipeline functionalities. The Settings page provides information about Kafka Configuration, Data Prep Scripts, Data Science Models, and Logger. The Default Configuration and Pod Status also get displayed under the Settings option.



The details of each Settings option is described under the Settings section of this document.

6. User Interface

The User Interface of the Data Pipeline provides a canvas for the user to build the data flow (Pipeline Workflow).

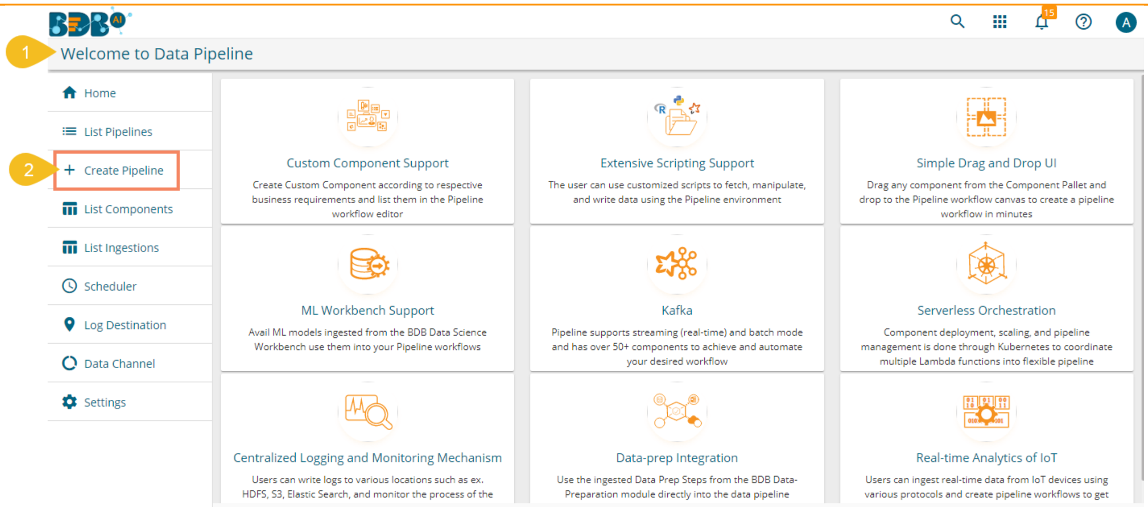
The Data Flow creation process can be divided into three parts as mentioned below:

1. Creating a Pipeline
2. Adding Components to the Canvas
3. Connecting Components

6.1. Creating a Pipeline

i) Navigate to the Data Pipeline landing page.

ii) Click the ‘**Create Pipeline**’ option provided on the top right side of the Pipeline landing page.



Welcome to Data Pipeline

- Home
- List Pipelines
- Create Pipeline
- List Components
- List Ingestions
- Scheduler
- Log Destination
- Data Channel
- Settings

Custom Component Support
Create Custom Component according to respective business requirements and list them in the Pipeline workflow editor

Extensive Scripting Support
The user can use customized scripts to fetch, manipulate, and write data using the Pipeline environment

Simple Drag and Drop UI
Drag any component from the Component Pallet and drop to the Pipeline workflow canvas to create a pipeline workflow in minutes

ML Workbench Support
Avail ML models ingested from the BDB Data Science Workbench use them into your Pipeline workflows

Kafka
Pipeline supports streaming (real-time) and batch mode and has over 50+ components to achieve and automate your desired workflow

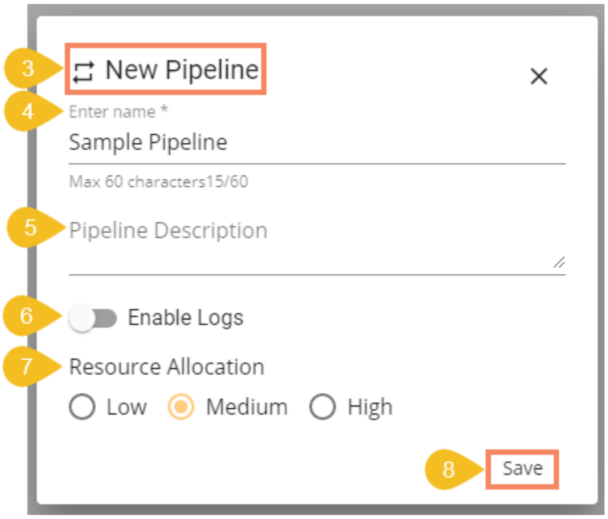
Serverless Orchestration
Component deployment, scaling, and pipeline management is done through Kubernetes to coordinate multiple Lambda functions into flexible pipeline

Centralized Logging and Monitoring Mechanism
Users can write logs to various locations such as ex. HDFS, S3, Elastic Search, and monitor the process of the

Data-prep Integration
Use the ingested Data Prep Steps from the BDB Data-Preparation module directly into the data pipeline

Real-time Analytics of IoT
Users can ingest real-time data from IoT devices using various protocols and create pipeline workflows to get

iii) The '**New Pipeline**' window opens asking the basic information.
iv) Enter a name for the new Pipeline.
v) Describe the Pipeline (Optional).
vi) Enable the logs to it as a file or elastic cube.
vii) Select a resource allocation option using the radio button- the given choices are:
   i. Low
   ii. Medium
   iii. High
viii) Click the '**Save**' option to create the pipeline. By clicking the 'Save' option, the user gets redirected to the pipeline workflow editor.



New Pipeline

Enter name *
Sample Pipeline
Max 60 characters15/60

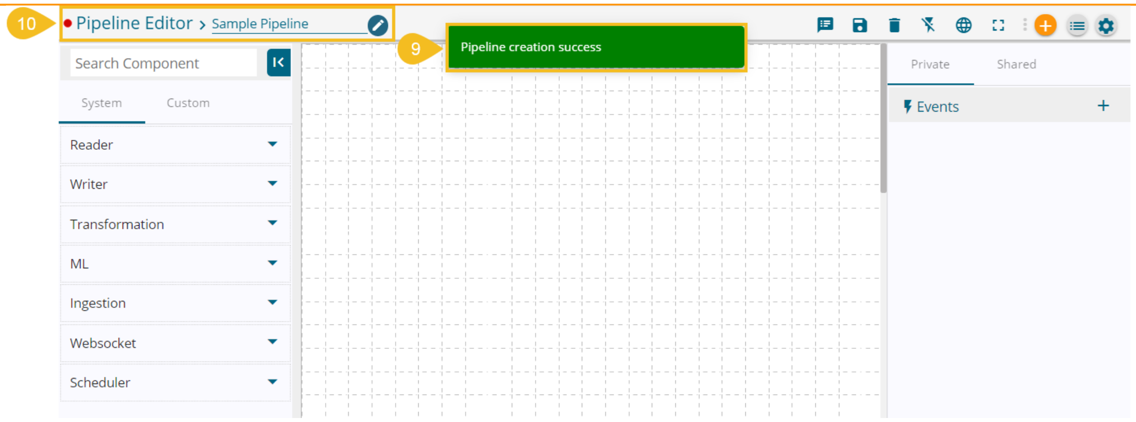
Pipeline Description

Enable Logs

Resource Allocation
○ Low ● Medium ○ High

Save

ix) A success message appears to confirm the creation of a new pipeline.
x) The Pipeline Editor page opens for the newly created pipeline.



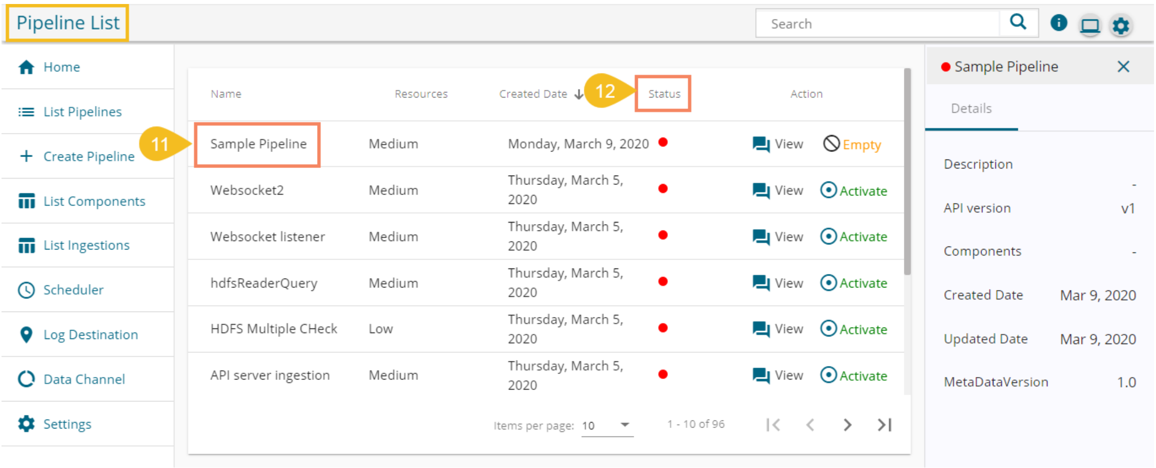
Pipeline creation success

xi) The newly created pipeline appears at the top of the Pipeline List.

xii) The status of the pipeline is indicated through color-coding under the 'Status' column.

a. Green-Activated Pipeline

b. Red- Deactivated Pipeline



Name	Resources	Created Date	Status	Action
Sample Pipeline	Medium	Monday, March 9, 2020	●	View Empty
Websocket2	Medium	Thursday, March 5, 2020	●	View Activate
Websocket listener	Medium	Thursday, March 5, 2020	●	View Activate
hdfsReaderQuery	Medium	Thursday, March 5, 2020	●	View Activate
HDFS Multiple CHeck	Low	Thursday, March 5, 2020	●	View Activate
API server ingestion	Medium	Thursday, March 5, 2020	●	View Activate

● Sample Pipeline

Description	-
API version	v1
Components	-
Created Date	Mar 9, 2020
Updated Date	Mar 9, 2020
MetaDataVersion	1.0

The following icons get displayed on the Pipeline Editor page:

Icon	Name	Function
⇤	Open/ Close the Components Pallet	Opens or Closes the Component Pallet
✎	Edit the Pipeline Name/resource allocation type	Allows to modify or change the given Pipeline name or change default resource configuration for pipeline components.
☰	Toggle Log Panel	Displays the UI Logs.
⊙	Activate/Deactivate Pipeline	Activates or deactivates the Pipeline. It appears once the user clicks the Update Pipeline icon
💾	Update Pipeline	Saves and updates the pipeline
🗑	Delete Pipeline	Deletes the pipeline
⚡ & ⚡	Toggle Event Panel	Opens the event panel to create and display events
🌐	Global Variable	Opens the Global Variables window to add new variables.
⛶	Full Screen	Displays the present page in full screen

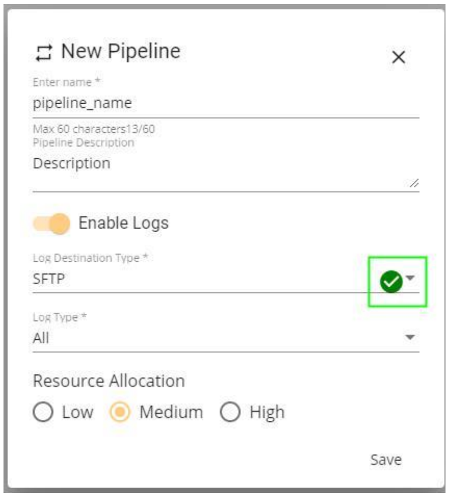
⊕	Create a New Pipeline	Redirects the users to create a new pipeline
☰	List Pipeline	Redirects the users to the page listing all the existing pipelines
💻	Pipeline Workflow	Redirects the users to the Pipeline Workflow
⚙	Settings	Redirects the users to the Pipeline Settings page

Note:

a. The user needs to choose 'Log Destination Type' and 'Log Type' information from the drop-down menu if Enable Logs option is enabled. As per the user requirement, the pipeline logs can be saved at a location.

b. Available location choices are HDFS, AWS S3, Elastic and SFTP.

c. Available Log Type choices are Info, Exception and All (wherein both the choices get selected).



New Pipeline

Enter name *
pipeline_name
Max 60 characters13/60

Pipeline Description
Description

Enable Logs

Log Destination Type *
SFTP

Log Type *
All

Resource Allocation
○ Low ● Medium ○ High

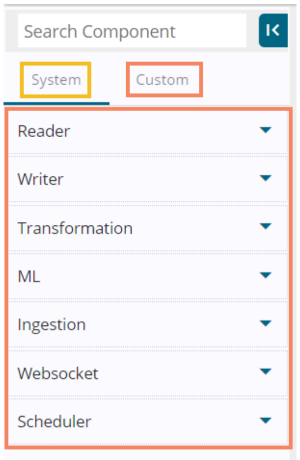
Save

d. The location of the above-mentioned storages can be configured through the Log Destination menu. Refer to the Log Destination section to know more information.

e. If Enable Logs option is enabled files get created at the given location and can be used further.

6.2. Adding Components to the Canvas

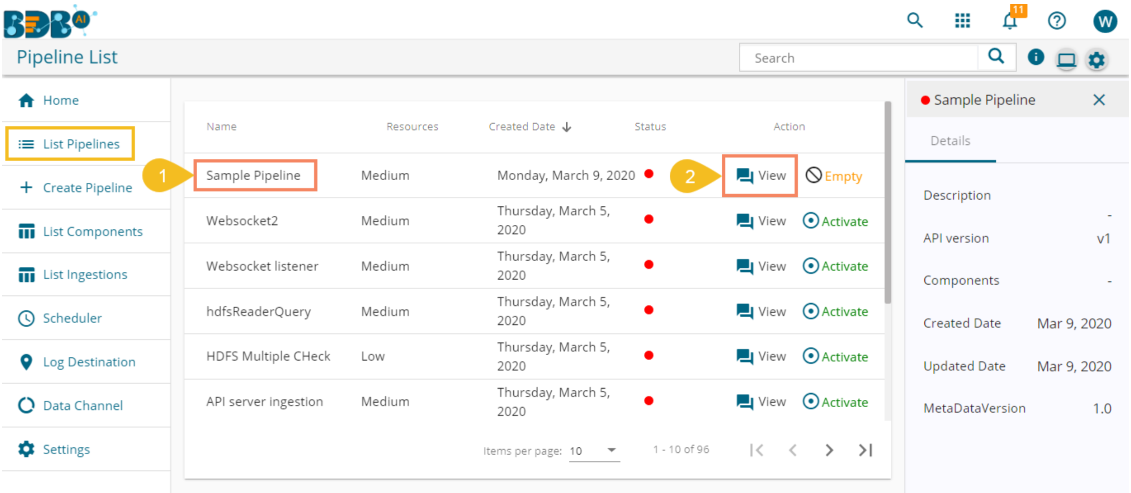
The Component Pallet is situated on the left side of the User Interface. It has a System and Custom Components tab listing the various components.



Once the Pipeline gets saved in the pipeline list, the user can add components to the canvas. The user can drag the required components to the canvas and configure it to create a Pipeline workflow or Data flow.

i) Navigate to the existing data pipeline from the Pipeline List page.

ii) Click the '**View**' icon for the pipeline.
(The user can perform this step-in continuation to the pipeline creation, but in case if the user has come out of the Pipeline Editor the above steps help him to access it again.)

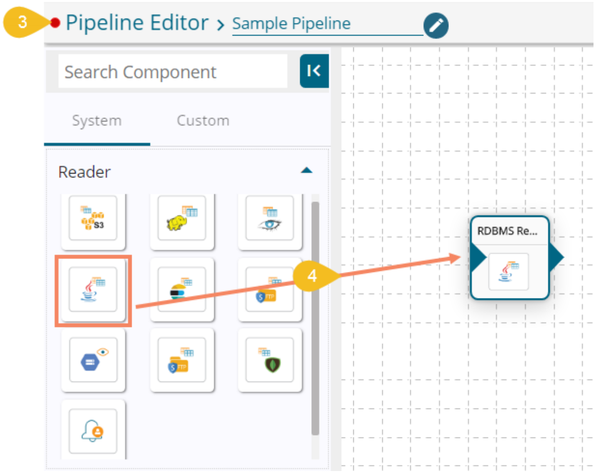


Name	Resources	Created Date	Status	Action
Sample Pipeline	Medium	Monday, March 9, 2020	●	View Empty
Websocket2	Medium	Thursday, March 5, 2020	●	View Activate
Websocket listener	Medium	Thursday, March 5, 2020	●	View Activate
hdfsReaderQuery	Medium	Thursday, March 5, 2020	●	View Activate
HDFS Multiple CHeck	Low	Thursday, March 5, 2020	●	View Activate
API server ingestion	Medium	Thursday, March 5, 2020	●	View Activate

Description	-
API version	v1
Components	-
Created Date	Mar 9, 2020
Updated Date	Mar 9, 2020
MetaDataVersion	1.0

iii) The Pipeline Editor opens for the selected pipeline.

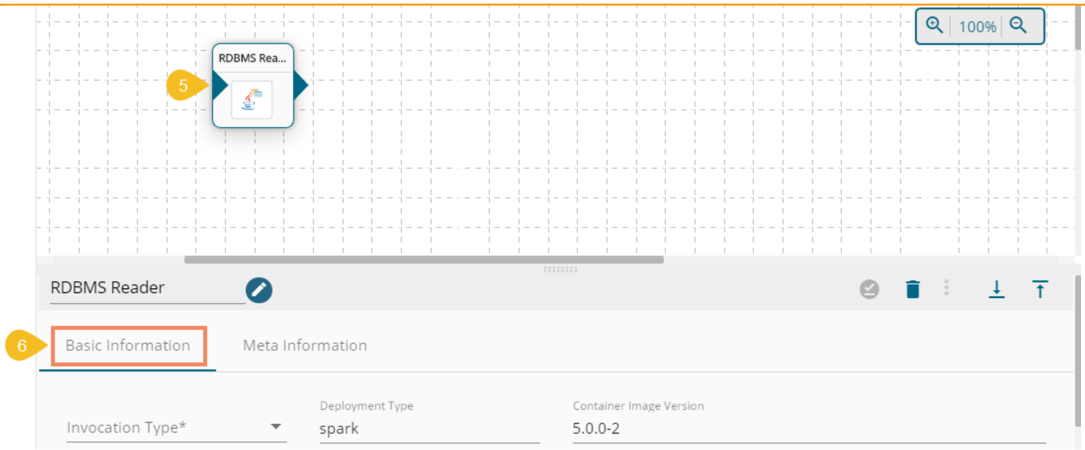
iv) Drag and drop the new required components or make changes in the existing component's meta information or change the component configuration (E.g., the RDBMS Reader is dragged to the workspace in the below given image).



v) Click on the component.

vi) The component-specific fields open asking the basic information about the dragged component:

a. Select an Invocation Type using the drop-down menu.



vii) Open the Meta Information tab and configure the required information for the dragged component.

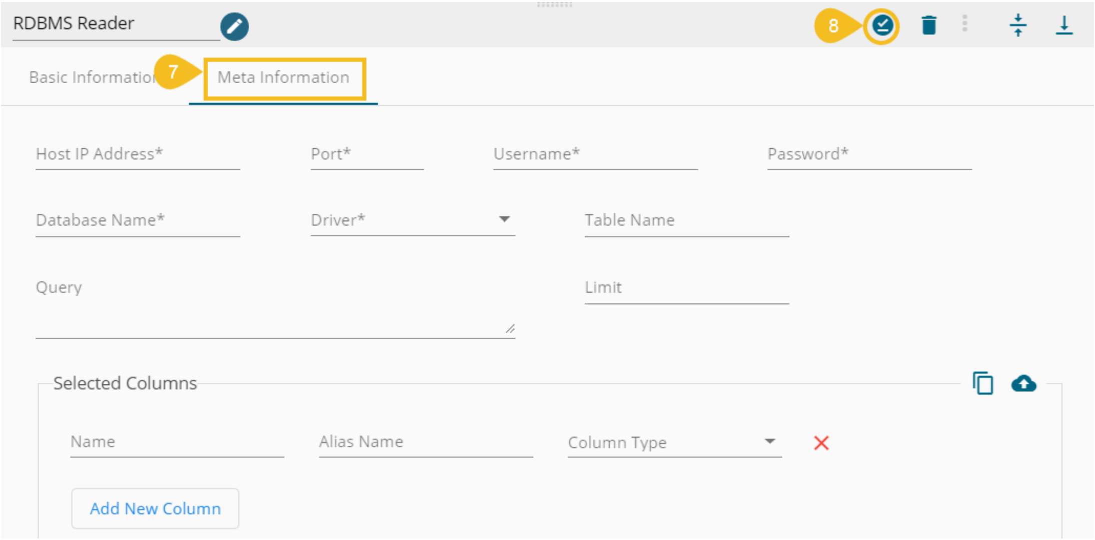
i. Host IP Address
ii. Port number
iii. Username
iv. Password
v. Database Name
vi. Driver- Select from the drop-down menu
vii. Table Name
viii. Query
ix. Limit
x. Selected Columns: Fill the following details manually to select the required columns
   1. Name
   2. Alias Name
   3. Column Type

   Or

   Use ‘Download Data’ and ‘Upload File’ options to select the desired columns.

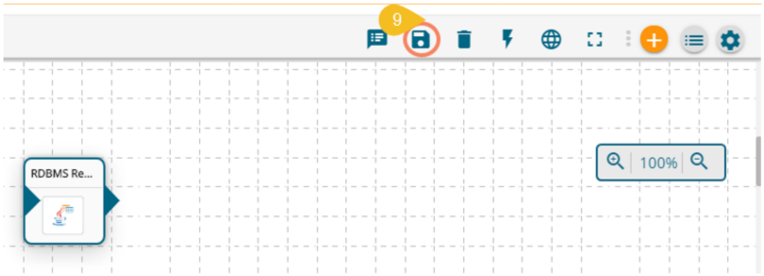
i. Upload File: The user can upload the existing system files (CSV, JSON) using the ‘Upload File’ icon (file size must be less than 2 MB).
ii. Download Data (Schema): Users can download the schema structure in JSON format by using the ‘Download Data’ icon.

viii) Make sure to click the save icon to update the component details and pipeline to reflect the recent changes in the pipeline.



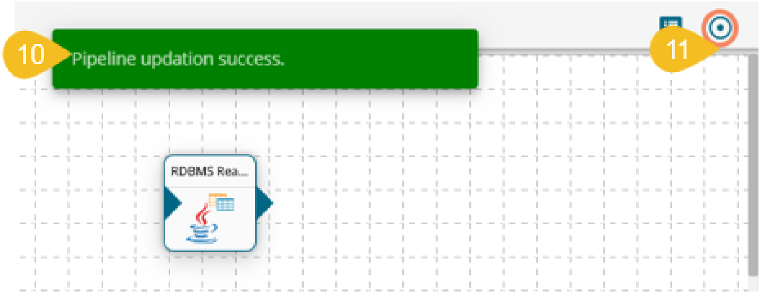
Note: The users can also upload the downloaded JSON file to another component using the 'Upload File' icon.

ix) Click the 'Update Pipeline' 💾 icon to save the changes.



x) A success message appears to assure that the pipeline has been successfully updated.

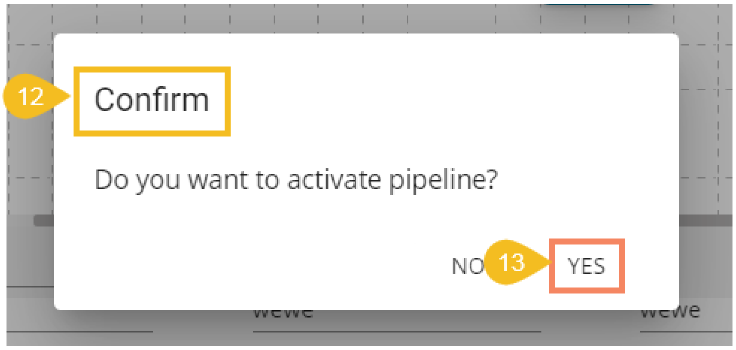
xi) Click the 'Activate Pipeline' ⊙ icon to activate the pipeline (It appears only after the newly created pipeline gets successfully updated).



Pipeline updation success.

xii) A dialog window opens to confirm the action of pipeline activation.

xiii) Click the '**YES**' to activate the pipeline.

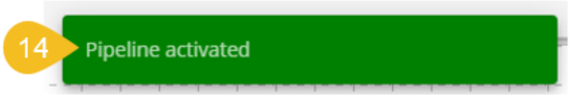


Confirm

Do you want to activate pipeline?

NO YES

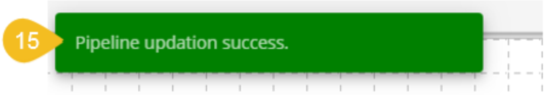
xiv) A success message appears confirming the activation of the pipeline.



Pipeline activated

Note: The details of the selected pipeline from the Pipeline List appears on the right side of the page.

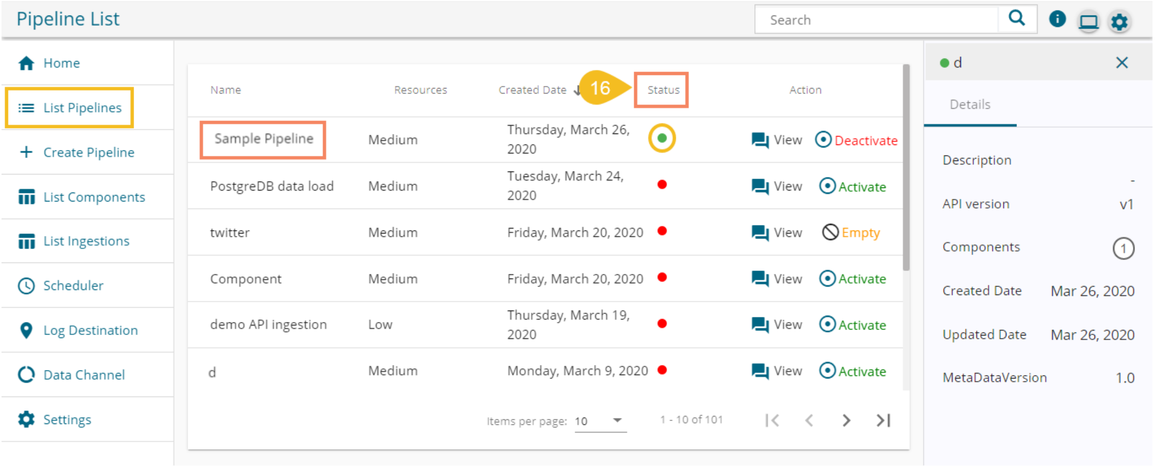
xv) Another success message appears to confirm that the pipeline has been updated.



15

Pipeline updation success.

xvi) The Status for the pipeline gets changed on the Pipeline List page.



Name	Resources	Created Date	Status	Action
Sample Pipeline	Medium	Thursday, March 26, 2020	●	View Deactivate
PostgreDB data load	Medium	Tuesday, March 24, 2020	●	View Activate
twitter	Medium	Friday, March 20, 2020	●	View Empty
Component	Medium	Friday, March 20, 2020	●	View Activate
demo API ingestion	Low	Thursday, March 19, 2020	●	View Activate
d	Medium	Monday, March 9, 2020	●	View Activate

Items per page: 10   1 - 10 of 101

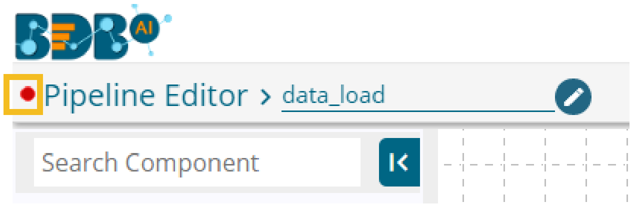
● d

Details

Description	-
API version	v1
Components	1
Created Date	Mar 26, 2020
Updated Date	Mar 26, 2020
MetaDataVersion	1.0

Note: Click the 'Delete' icon from the Pipeline Editor page to delete the selected pipeline. The deleted Pipeline gets removed from the Pipeline list.

The status (activated/deactivated) of the pipeline is indicated through color-coding as highlighted in the below image. It turns green if the pipeline is active and remains red for the inactive pipeline.



Please refer to the Component Pallet section to get detailed information on the Components.

6.3. Connecting Components

Each component in the Data Pipeline works as an independent entity. The user needs to create an Event as a Kafka based messaging channel to create a data flow. An Event helps the users to read data from various readers or use the data coming from one pipeline into another pipeline. The Kafka Event helps to deal with the scenarios where one pipeline is dependent on another pipeline's data.

There are two tabs in the Event Panel as described below:

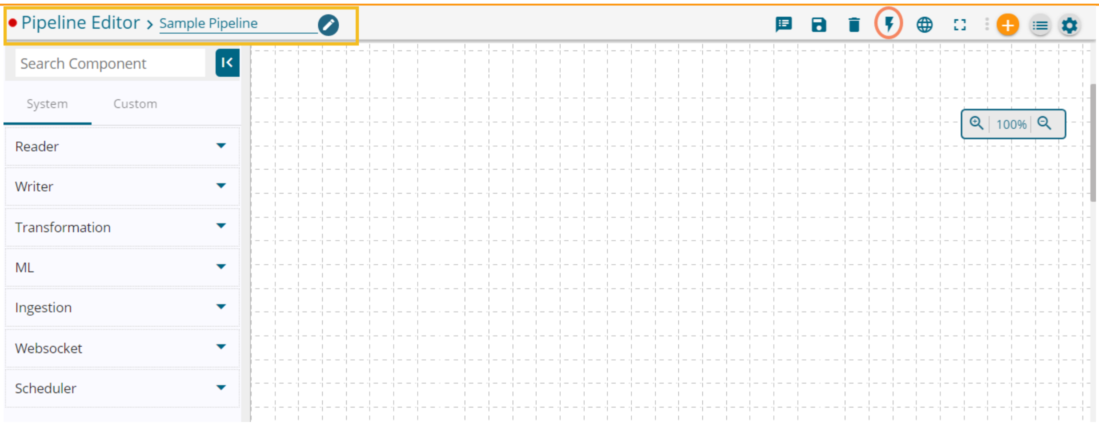
i) Private: This tab contains a list of all the created events by the user for the current pipeline.

ii) Shared: This tab contains a list of all the shared events from another pipeline.

6.3.1. Event Creation

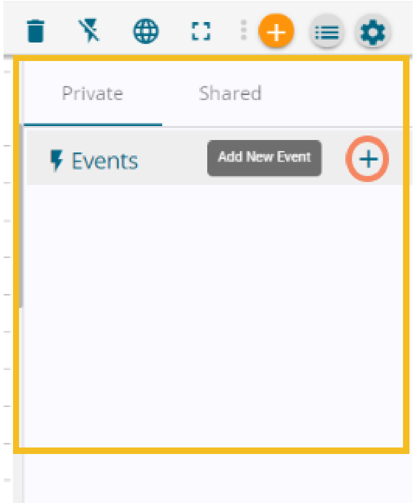
i) Navigate to the Pipeline Editor page.

ii) Click the '**Toggle Event Panel**' icon from the header.

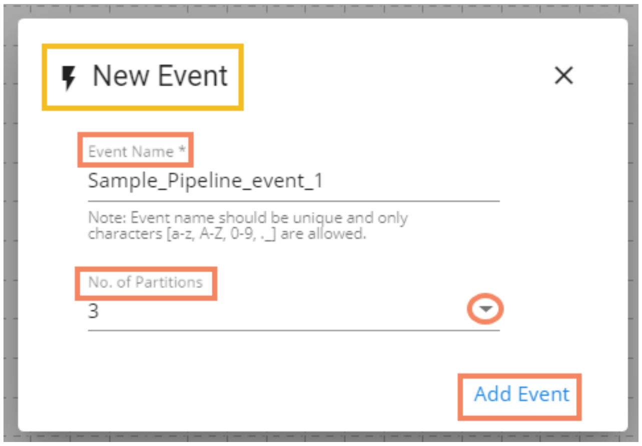


iii) The Events Panel appears, and the Toggle Event Panel icon gets changed as ⚡ suggesting that the event panel is displayed.

iv) Click the ‘**Add New Event**’ icon from the Event Panel.



v) The New Event window opens.
   a. Provide a name for the new Event.
   b. Click the ‘**Add Event**’ option.



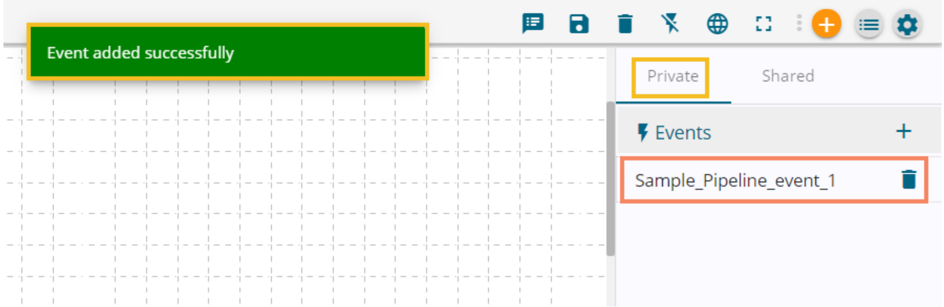
New Event

Event Name *
Sample_Pipeline_event_1

Note: Event name should be unique and only characters [a-z, A-Z, 0-9, ._] are allowed.

No. of Partitions
3

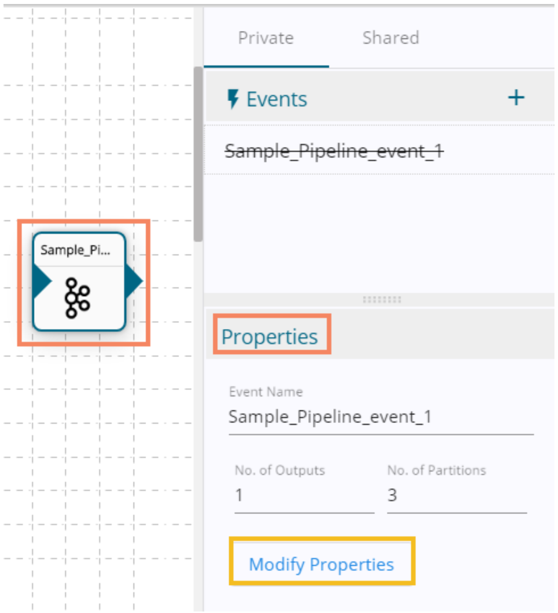
vi) The newly created Event lists under the Private tab in the Events panel.



Event added successfully

6.3.2. Updating an Event

i) Click on the newly created Event to open the Properties pallet for the event.
ii) Drag the newly created event to display the Properties tab.
iii) Click the ‘**Modify Properties**’ option.

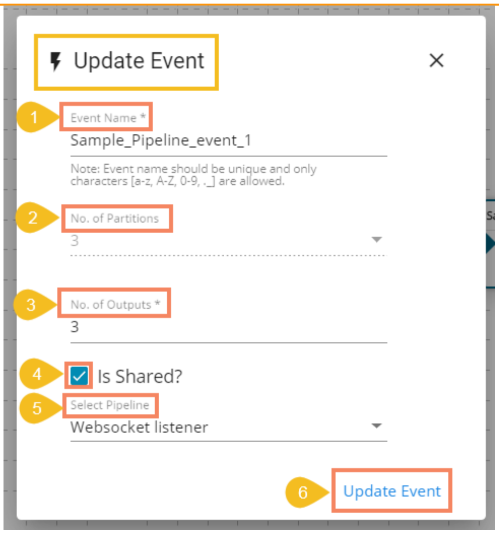


Event Name: Sample_Pipeline_event_1

No. of Outputs: 1

No. of Partitions: 3

iv) The Update Event window opens.
v) The user can configure the following information:
  a. Event Name: Modify the event name
  b. No. of Partitions: this field appears displaying default no. of partition that is 3. The user cannot change the number.
  c. No. of Outputs: Set the number of outputs (the maximum allowed number of output is 3)
  d. Is Shared ?: Enable this option to share the created event.
  e. Select Pipeline: Select one pipeline or multiple pipelines using the drop-down list.
vi) Click the ‘**Update Event**’ option to update the recent changes.



Update Event

1 Event Name *

Sample_Pipeline_event_1

Note: Event name should be unique and only characters [a-z, A-Z, 0-9, ._] are allowed.

2 No. of Partitions

3

3 No. of Outputs *

3

4 ☑ Is Shared?

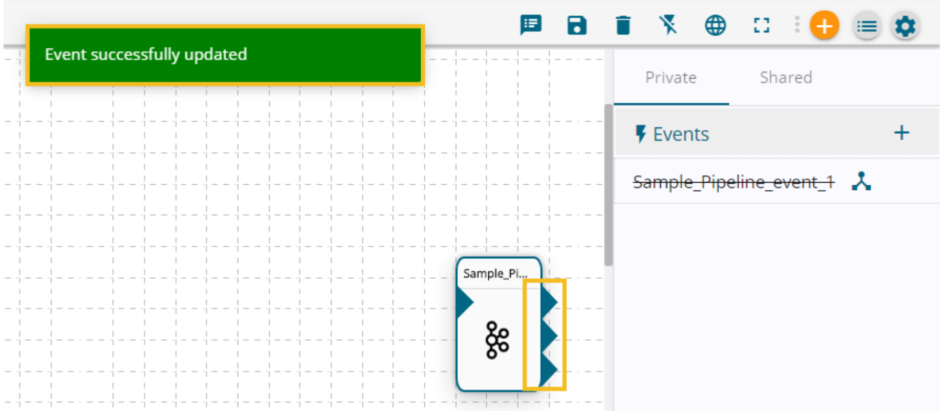
5 Select Pipeline

Websocket listener

6 Update Event

vii) A success message appears to confirm the updates.

viii) The event gets the same number of the output nodes as mentioned in the Update Event window.

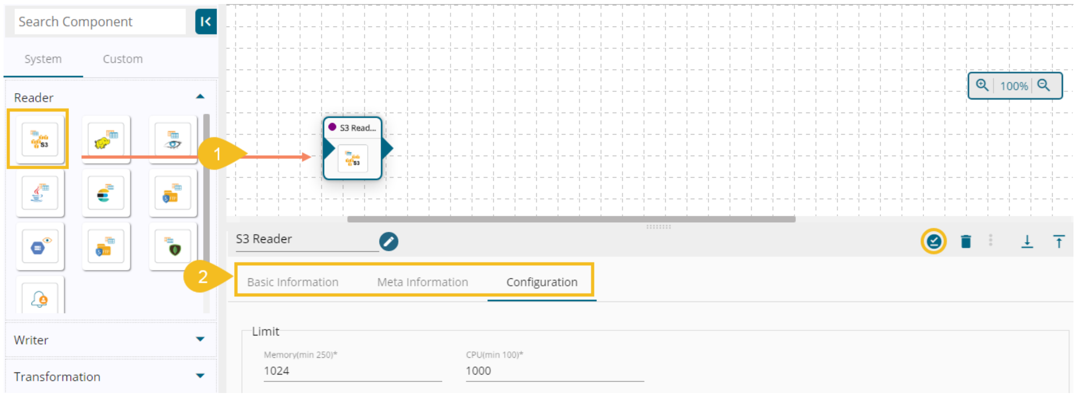


Event successfully updated

6.3.3. Connecting Event to a Component

i) Drag a (reader) component to the canvas.

ii) Configure the parameters of the dragged reader.



S3 Reader

Basic Information | Meta Information | Configuration

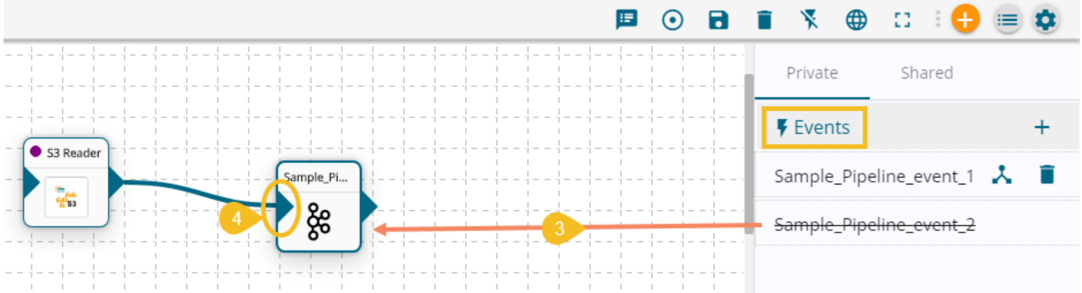
Limit

Memory(min 250)*: 1024

CPU(min 100)*: 1000

iii) Drag the Event from the Events Panel.

iv) Connect the dragged reader component as the input connection to the dragged Event.



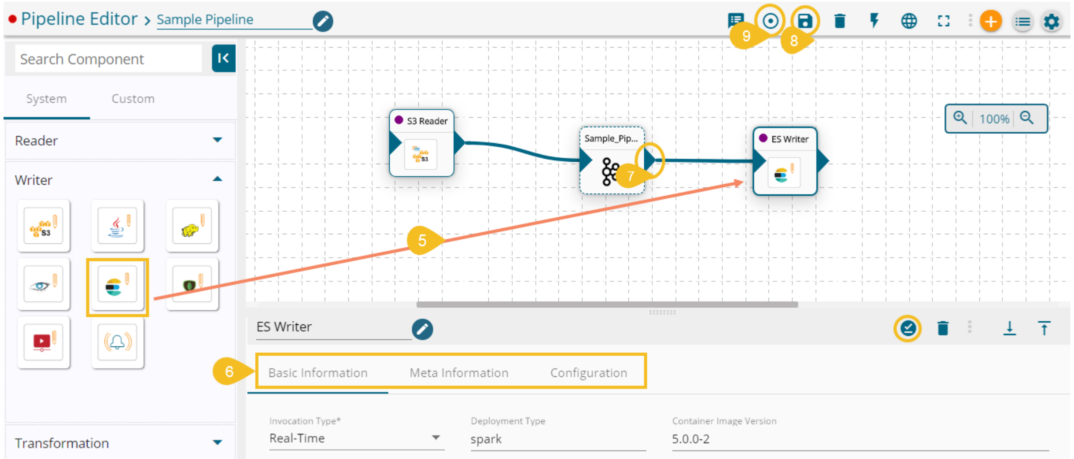
v) Drag a Writer component to the Editor canvass from the Component Pallet.

vi) Configure the writer component.

vii) Connect the Writer component as an output connection to the dragged Event to complete the data flow.

viii) Click the ‘**Update Pipeline**’ icon to save the pipeline workflow.

ix) Click the ‘**Activate Pipeline**’ icon to activate the Pipeline.



Invocation Type* Real-Time

Deployment Type spark

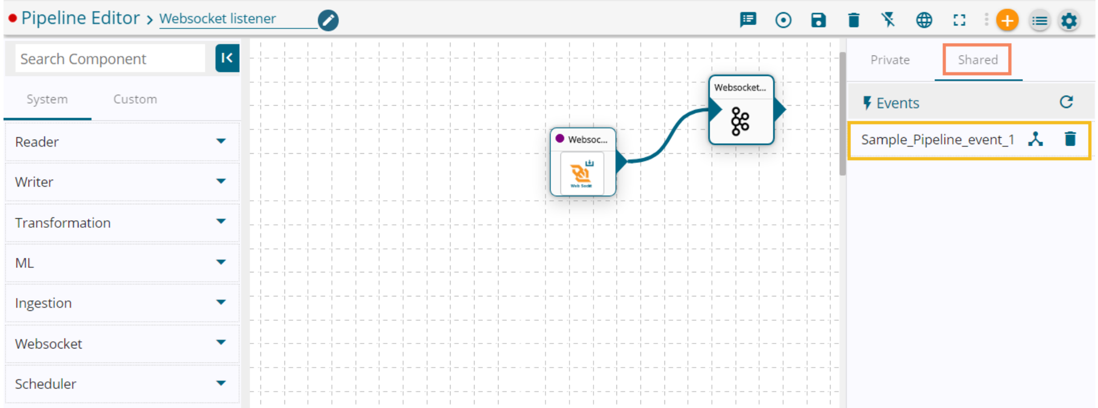
Container Image Version 5.0.0-2

Note: The user gets notification messages on the successful update and activation of the Data Pipeline.

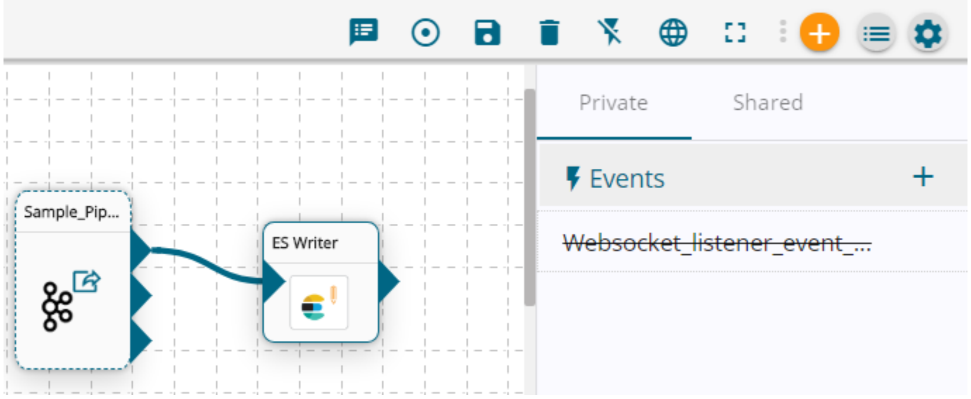
6.3.4. Shared Event

i) Open the pipeline where the same event is shared.

ii) The shared event gets added to the ‘**Shared**’ section of the Event.



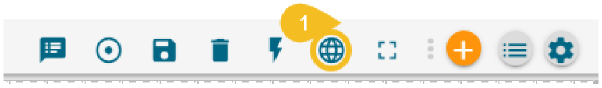
iii) The user can use the Shared Event in the new pipeline with a writer as displayed in the following image:



6.4. Global Variable

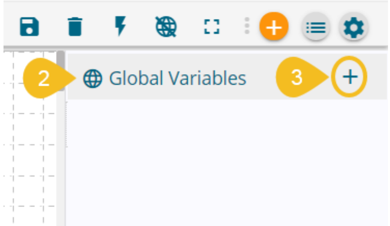
The user can create a Global Variable by following the below given steps:

i) Click the Global Variable icon from the User Interface.



ii) A Global Variables panel opens.

iii) Click the '**Add New Variable**' from the Global Variables panel.

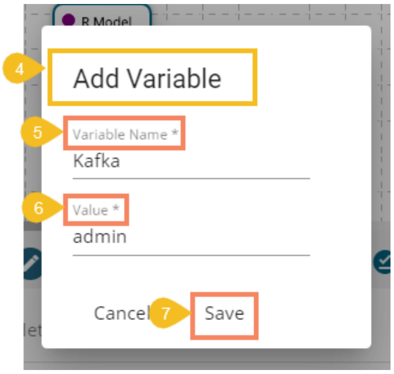


iv) The Add Variable window opens.

v) Insert Variable name.

vi) Provide Value for the Variable.

vii) Click the '**Save**' option.



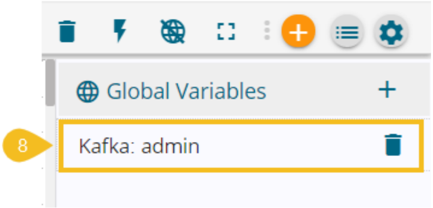
Add Variable

Variable Name *
Kafka

Value *
admin

Cancel Save

viii) The Global Variable gets created and added to the panel.

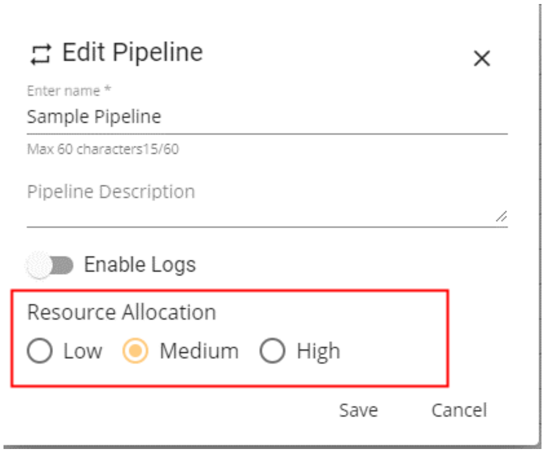


Kafka: admin

6.5. Resource Allocation

The following is used to deploy the pipeline with high, medium, or low-end configurations according to the velocity and volume of data that the pipeline must handle.

All the components saved in the pipeline are then allocated resources based on the selected Resource Allocation option depending on the component type (Spark and Docker).



Enter name *
Sample Pipeline
Max 60 characters15/60

Pipeline Description

Enable Logs

Resource Allocation
○ Low ● Medium ○ High

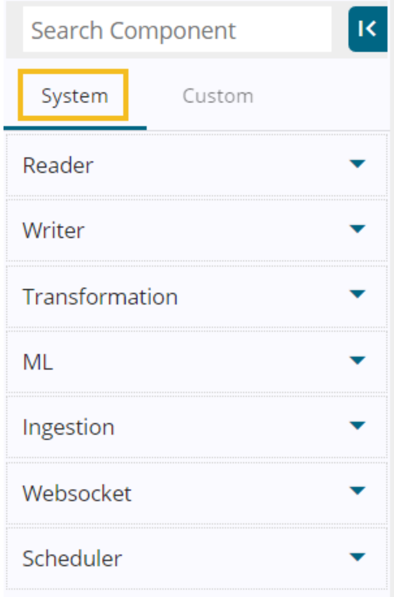
7. Component Pallet

The user interface of the Data Pipeline has a component pallet on the left side on the page. The Component Pallet lists System and Custom components of the Data Pipeline.

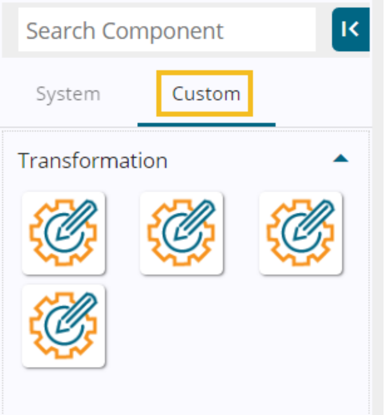
7.1. Sorting and Grouping Components

The Component Pallet displays a list of all the pipeline components. The Components are majorly divided into two groups.

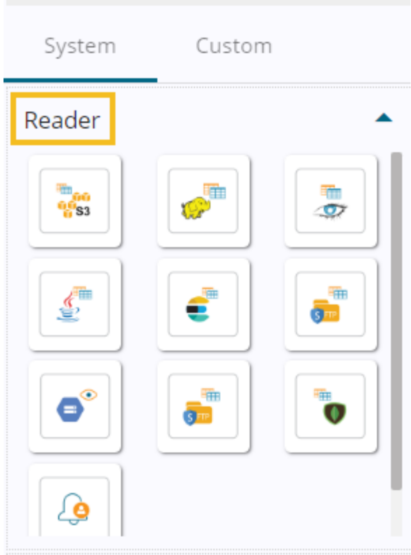
i) System-The pre-designed pipeline components are listed under the '**System**' tab.



ii) The custom-The Custom tab lists all the customized pipeline components created by the user.



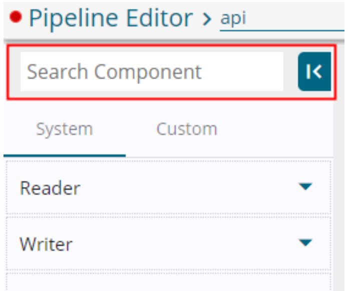
The Component gets grouped based on the Component type. E.g., All the reader components are provided under the 'Reader' menu tab of the Component Pallet.



7.2. Searching a Component

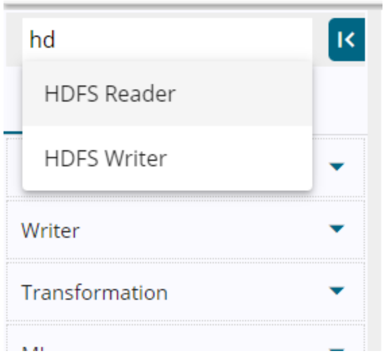
A search bar is provided to search across the 50+ components. It helps to find a specific component by inserting the name in the Search Bar.

i) Navigate to the Search bar provided in the Component Pallet.



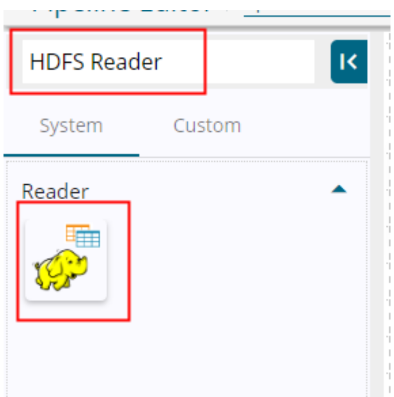
ii) Type in the given search bar.

iii) The user gets prompt suggestions for the searched components. E.g., While searching 'hd', it lists HDFS Reader and HDFS Writer components.



iv) Select an option from the prompted choices.

v) The searched component appears under the 'System' tab. (E.g., HDFS Reader component as displayed in the below image)



8. Component Properties

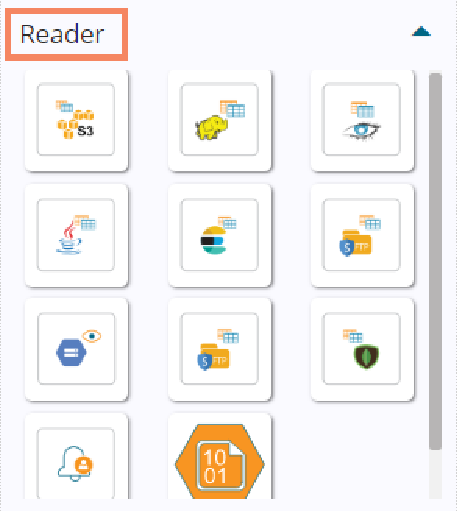
Each component has properties to configure. The user can use a specific component into the Pipeline Workflow only after configuring the required parameters for that component.

This section aims at describing all the component parameters as per the given groups under the Component Pallet.

8.1. Reader

Data Reader component helps to read data from different sources and ingest that data into the Pipeline for further processing.

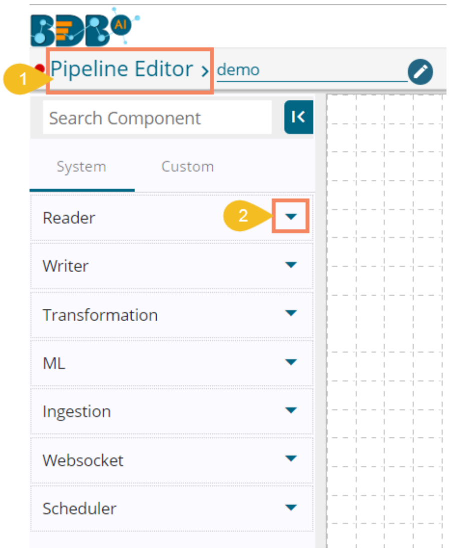
The Data Pipeline contains the following Data Readers under the Component Pallet:



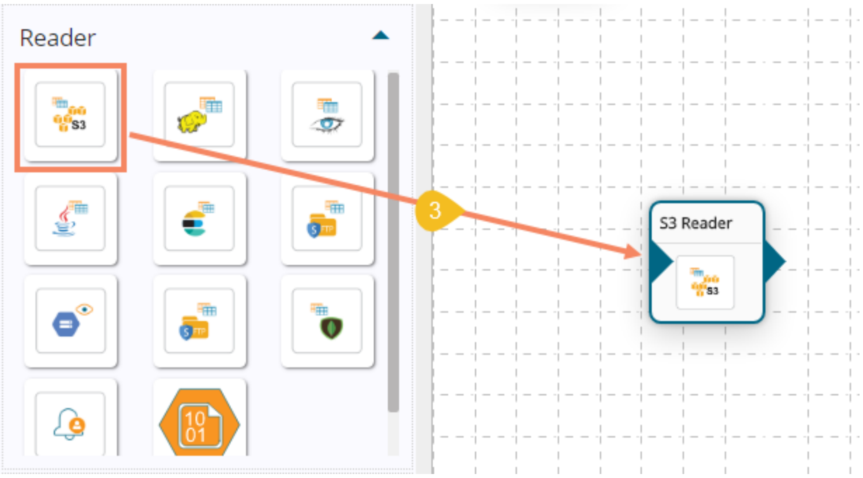
8.1.1. S3 Reader

i) Navigate to the Data Pipeline Editor.

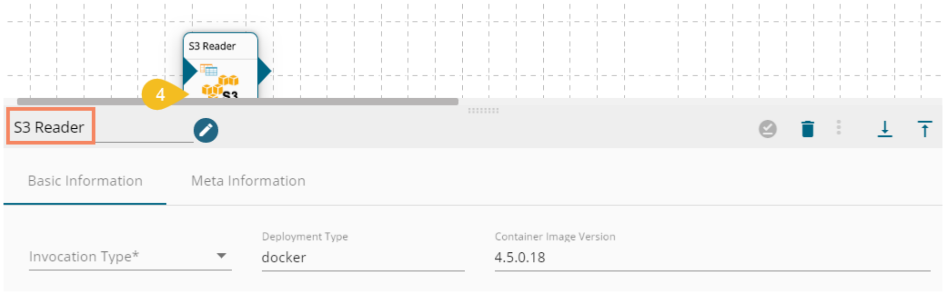
ii) Expand the Reader section provided under the Component Pallet.



iii) Drag and drop the S3 Reader component to the Workflow Editor.



iv) Click on the dragged S3 Reader to get the component properties tabs.



Deployment Type: docker

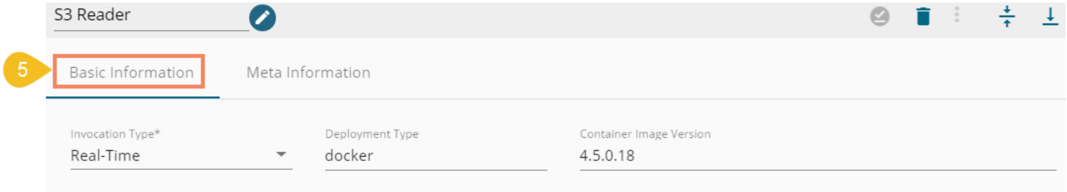
Container Image Version: 4.5.0.18

v) **Basic Information**: It is the default tab to open for the component while configuring it.

a. Invocation Type: Select an invocation mode out of ‘Real-Time’ or ‘Batch’ using the drop-down menu.

b. Deployment Type: It displays the deployment type for the reader component. This field comes pre-selected.

c. Container Image Version: It displays the image version for the docker container. This field comes pre-selected.



Invocation Type*: Real-Time

Deployment Type: docker

Container Image Version: 4.5.0.18

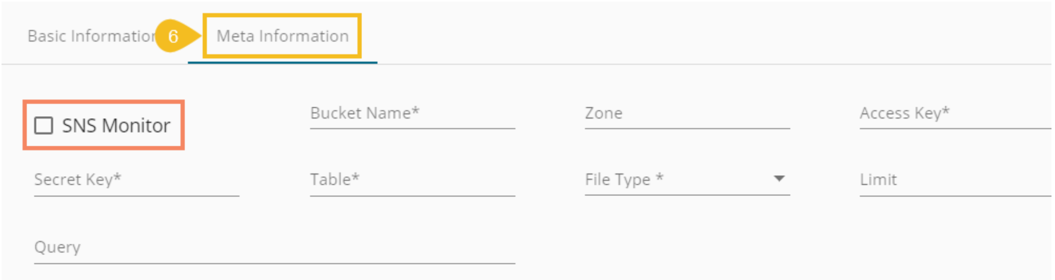
vi) Open the ‘**Meta Information**’ tab and fill all the connection-specific details for the S3 Reader.

- Configuration fields when SNS Monitor is disabled:
  i) Bucket Name(*): Folder Name
  ii) Zone(*): S3 Zone location
  iii) Access Key(*): Access key shared by AWS to login
  iv) Secret Key(*): Secret key shared by AWS to login
  v) Table(*): Mention the Table or object name which is to be read

vi) File Type(*): Select a file type from the drop-down menu (CSV, JSON, PARQUET, AVRO are the supported file types)

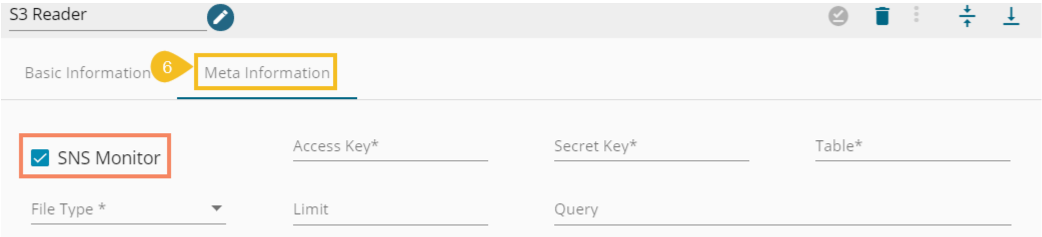
vii) Limit: Set limit for the number of records.

viii) Query: Insert an SQL query (it supports query containing a join statement as well)



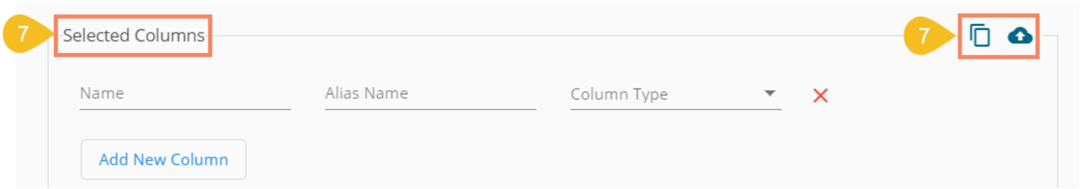
- Configuration fields when SNS Monitor is enabled:

1. Access Key(*): Access key shared by AWS to login
2. Secret Key(*): Secret key shared by AWS to login
3. Table(*): Mention the Table or object name which has to be read
4. File Type(*): Select a file type from the drop-down menu (CSV, JSON, PARQUET, AVRO are the supported file types)
5. Limit: Set limit for the number of records
6. Query: Insert an SQL query (it supports query containing a join statement as well)

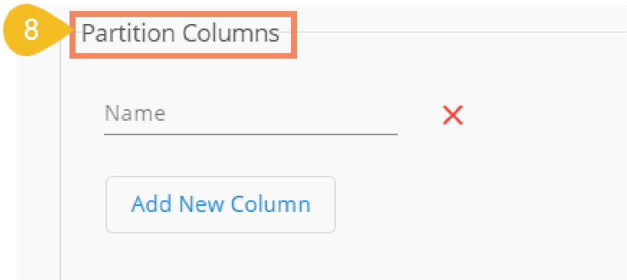


vii) Selected Columns: There is also a section for the selected columns in the Meta Information tab if the user can select some specific columns from the table to read data instead of selecting a complete table so this can be achieved by using the 'Selected Columns' section. Select the columns which you want to read and if you want to change the name of the column, then put that name in alias name section otherwise keep alias name same as of column name and then select a Column Type from the drop-down menu.
or
Use '**Download Data**' and '**Upload File**' options to select the desired columns.

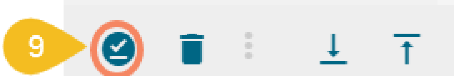
1. Upload File: The user can upload the existing system files (CSV, JSON) using the '**Upload File**' ☁ icon (file size must be less than 2 MB).
2. Download Data (Schema): Users can download the schema structure in JSON format by using the '**Download Data**' ⧉ icon.



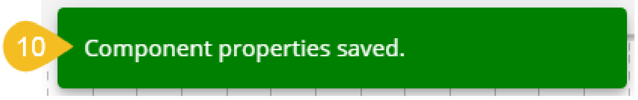
viii) **Partition Columns**: Provide a unique Key column name to partition data in Spark.



ix) Click the '**Save Component in Storage**' icon after doing all the configurations to save the reader component.

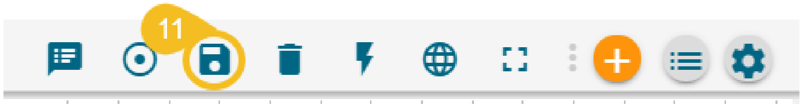


x) A notification message appears to inform about the component configuration success.

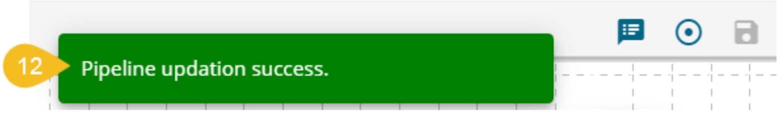


Component properties saved.

xi) Click the '**Update Pipeline**' icon.

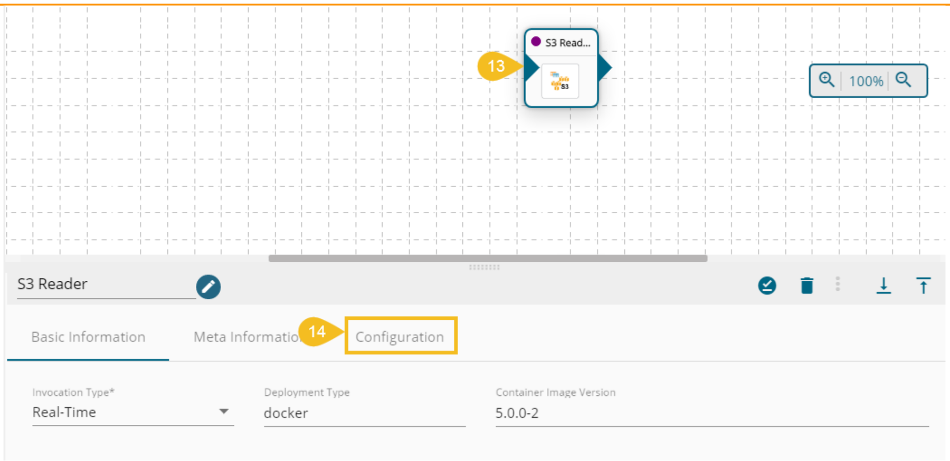


xii) A notification message appears to inform that the pipeline is successfully updated.



Pipeline updation success.

xiii) Open the S3 Reader component by a click on it.
xiv) The S3 Reader gets the '**Configuration**' properties tab.



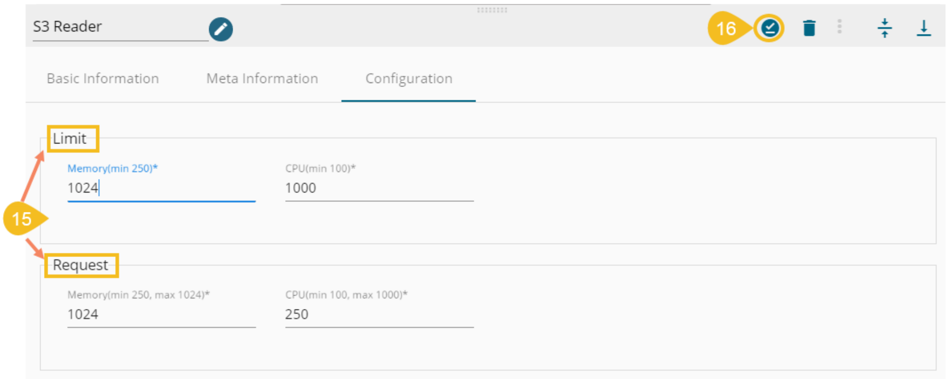
Invocation Type*: Real-Time

Deployment Type: docker

Container Image Version: 5.0.0-2

xv) Modify the values for the Limit or Request section from the Configuration tab.

xvi) Click the ‘**Save Component in Storage**’ ✓ icon.



Limit

Memory(min 250)*: 1024

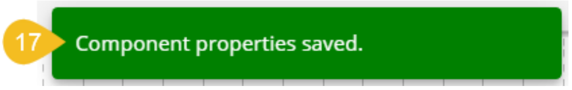
CPU(min 100)*: 1000

Request

Memory(min 250, max 1024)*: 1024

CPU(min 100, max 1000)*: 250

xvii) A message appears to notify about the component properties.

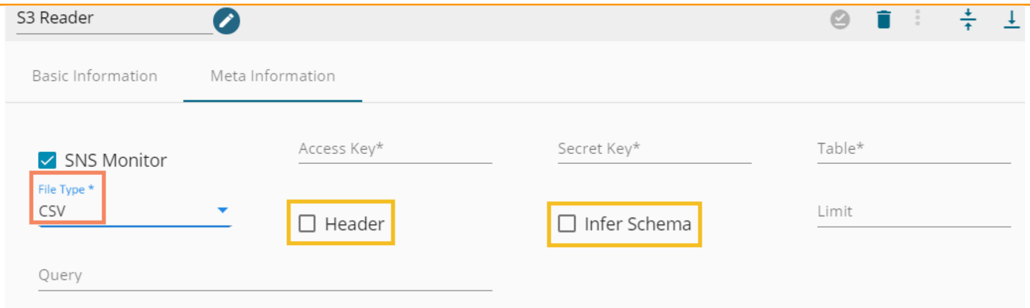


Component properties saved.

xviii) The component properties get configured for the S3 Reader and it is ready to be used in the Pipeline Workflow.

Note:

a. (*) symbol indicates that the field is mandatory.
b. Either table or query must be specified for the data readers except for SFTP Reader.
c. Selected Columns- There should not be a data type mismatch in the Column Type for all the Reader components.
d. The Meta Information fields may vary based on the selected File Type. All the possibilities are mentioned below:
    i. CSV: ‘**Header**’ and ‘**Infer Schema**’ fields get displayed with CSV as the selected File Type.



S3 Reader

Basic Information | Meta Information

☑ SNS Monitor

File Type *: CSV

Access Key*

Secret Key*

Table*

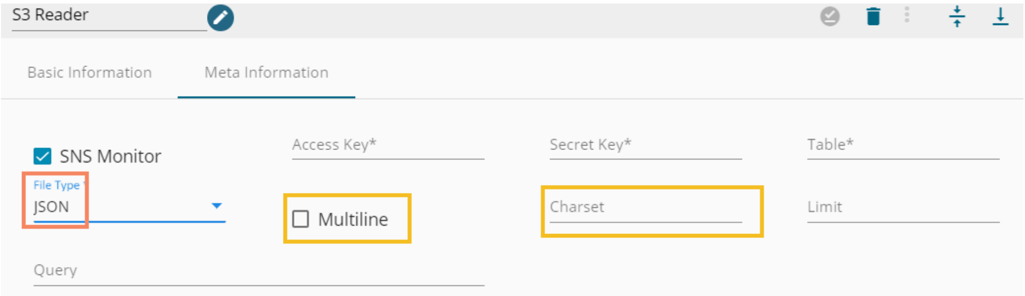
☐ Header

☐ Infer Schema

Limit

Query

ii. JSON: ‘**Multiline**’ and ‘**Charset**’ fields get displayed with JSON as the selected File Type.



S3 Reader

Basic Information | Meta Information

☑ SNS Monitor

File Type *: JSON

Access Key*

Secret Key*

Table*

☐ Multiline

Charset

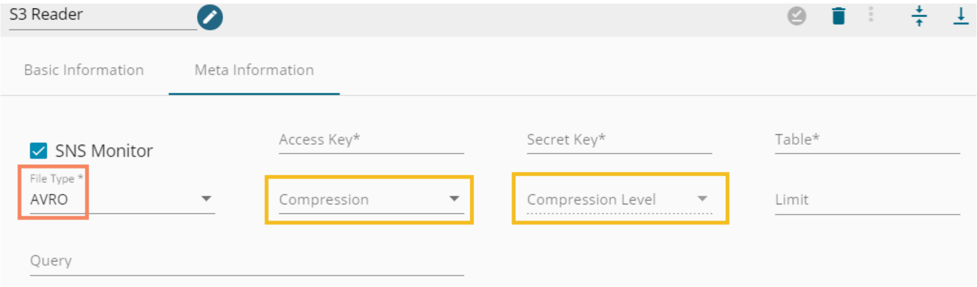
Limit

Query

iii. PARQUET: No extra field gets displayed with PARQUET as the selected File Type.

iv. AVRO: This File Type provides two drop-down menus.

1. Compression: Select an option out of ‘**Deflate**’ and ‘**Snappy**’ option.
2. Compression Level: This field appears for the Deflate compression option. It provides 0 to 9 levels via a drop-down menu.



S3 Reader

Basic Information | Meta Information

☑ SNS Monitor

File Type *: AVRO

Access Key*

Secret Key*

Table*

Compression

Compression Level

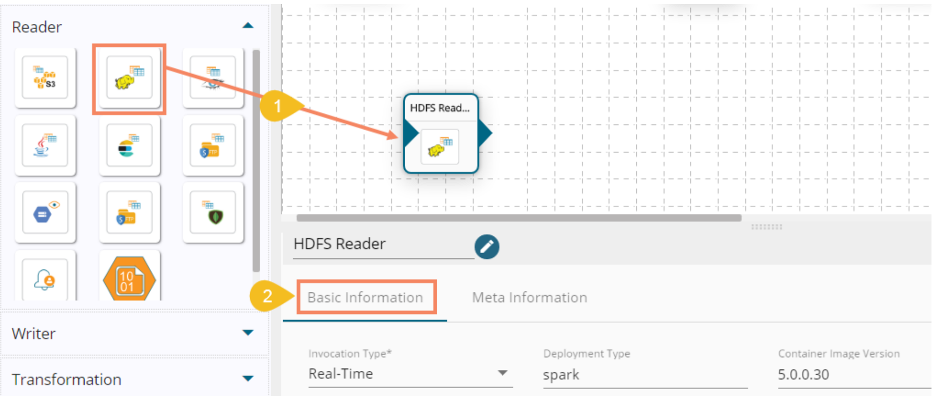
Limit

Query

8.1.2. HDFS Reader

i) Drag & Drop the HDFS Reader component on the Workflow Editor.

ii) Click on the dragged reader component to open the component properties tabs below.



HDFS Reader

Basic Information | Meta Information

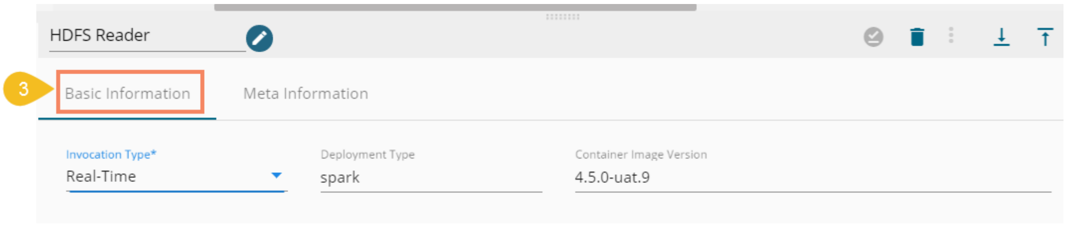
Invocation Type*: Real-Time

Deployment Type: spark

Container Image Version: 5.0.0.30

iii) Basic Information: It is the default tab to open for the component while configuring it.

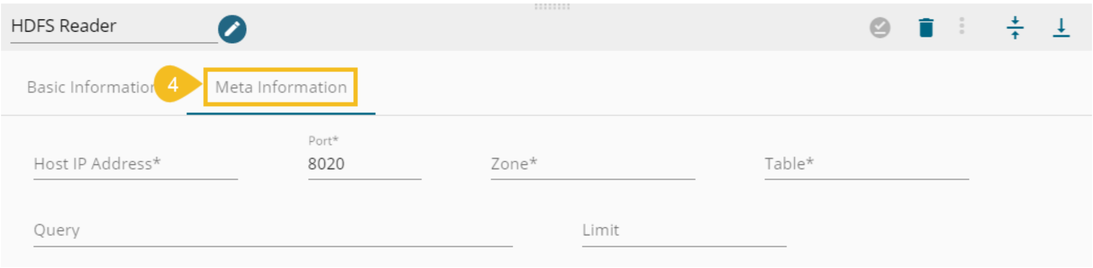
a. Select an Invocation type from the drop-down menu to confirm the running mode of the reader component. Select '**Real-Time**' or '**Batch**' from the drop-down menu.
b. Deployment Type: It displays the deployment type for the component. This field comes pre-selected.
c. Container Image Version: It displays the image version for the docker container. This field comes pre-selected.



Invocation Type*: Real-Time
Deployment Type: spark
Container Image Version: 4.5.0-uat.9

iv) Open the '**Meta Information**' tab and fill all the connection-specific details for the HDFS Reader.

1. Host IP Address(*): Hadoop IP address of the host
2. Port(*): Port number
3. Zone(*): HDFS Zone location
4. Table(*): Table or object name which must be read
5. Query: Insert an SQL query (it takes query containing a Join statement as well)
6. Limit: Set limit for the number of records.



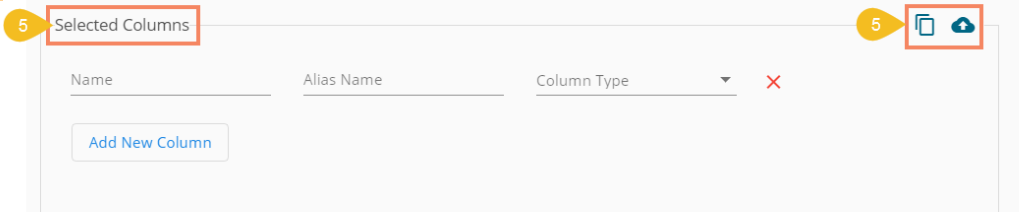
Port*: 8020

v) **Selected Columns**: The users can select some specific columns from the table to read data instead of selecting a complete table; this can be achieved via the **'Selected Columns**' section. Select the columns which you want to read and if you want to change the name of the column, then put that name in alias name section otherwise keep alias name same as of column name and then select a Column Type from the drop-down menu.

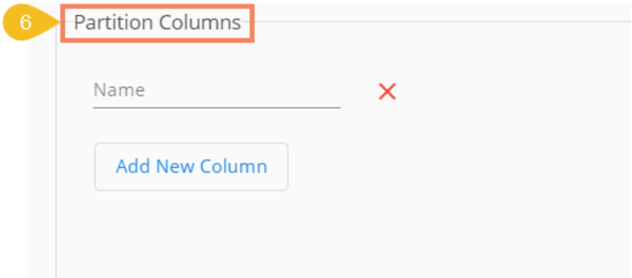
or

Use '**Download Data**' and '**Upload File**' options to select the desired columns.

1. Upload File: The user can upload the existing system files (CSV, JSON) using the '**Upload File**' ☁ icon (file size must be less than 2 MB).
2. Download Data (Schema): Users can download the schema structure in JSON format by using the '**Download Data**' ⧉ icon.

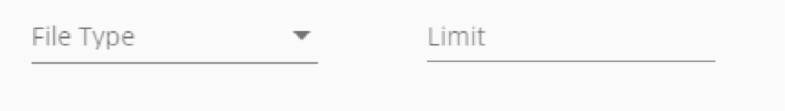


vi) Partition Column- Provide a unique Key column name to partition data in Spark.



i. File Type

The users get the File Type and Limit fields at the end. The File Type is a mandatory field for the HDFS reader.

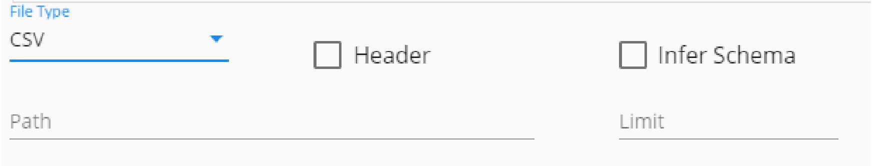


Currently, we support CSV, JSON, PARQUET, AVRO file types. The configuration fields may vary based on the selection of the file type.

1. CSV

   Configure the following fields when the selected file type is CSV:

   a. Header- Enable or Disable the Header option to confirm whether the stored data has header or not
   b. Infer Schema- Enable or disable the Infer Schema option to confirm whether the schema is provided or not
   c. Path- Provide a specific path for the file
   d. Limit- Set limit for the number of rows



File Type: CSV

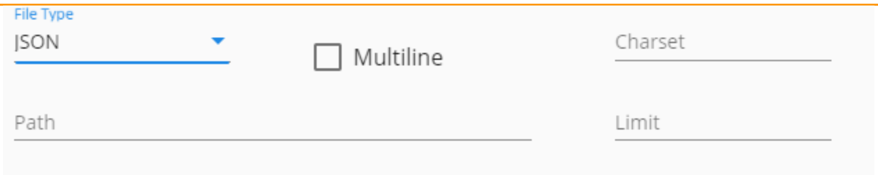
□ Header

□ Infer Schema

2. JSON

   Configure the following fields when the selected file type is JSON:

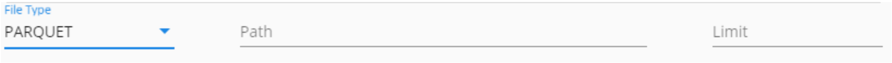
   a. Multiline-Enable this option, if the stored JSON has line breaks in-between
   b. Charset-Character set encoding, E.g., UTF-8
   c. Path-provide a specific path for the file
   d. Limit-set limit for the number of rows


File Type
JSON
□ Multiline
Charset
Path
Limit

3. PARQUET

   Configure the following fields when the selected file type is PARQUET:

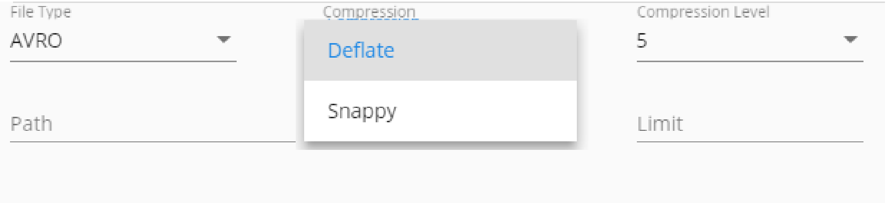
   a. Path-provide a specific path for the file
   b. Limit- set limit for the number of rows


File Type
PARQUET
Path
Limit

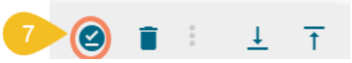
4. AVRO

   Configure the following fields when the selected file type is AVRO:

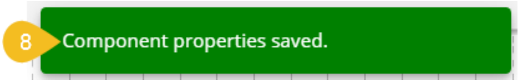
   a. Compression: Select a compression option from the drop-down menu out of ‘**Deflate**’ and ‘**Snappy**’ options.
   b. Compression Level- If the selected Compression option is ‘**Deflate**’ then select the compression level from 0-9 (where 0 and 9 are included).
   c. Path-provide the specific path for the file.
   d. Limit- set limit for the number of rows.


File Type
AVRO
Compression
Deflate
Snappy
Compression Level
5
Path
Limit

vii) After doing all the configurations click the ‘**Save Component in Storage**’ icon provides in the reader configuration panel to save the component.

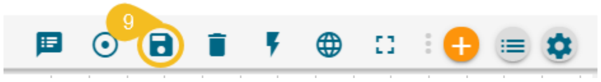


viii) A notification message appears to inform about the component configuration success.

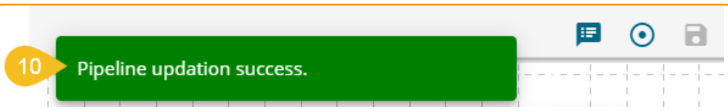


Component properties saved.

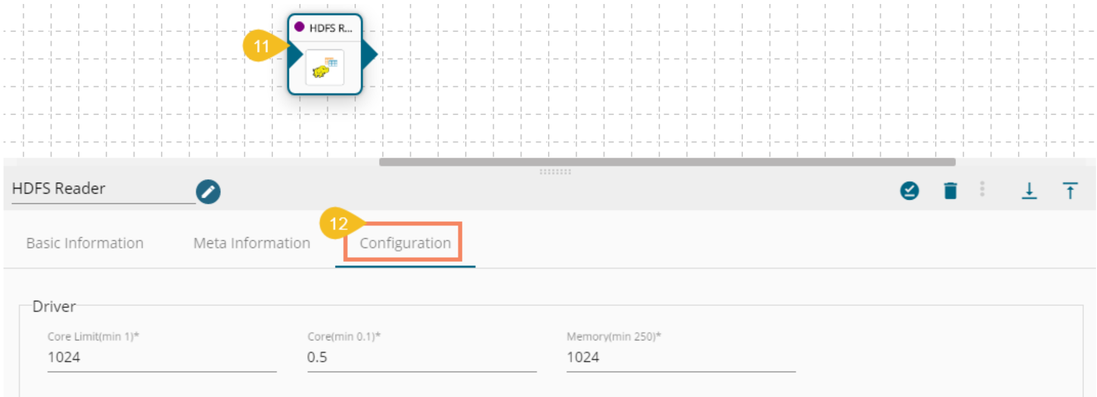
ix) Click the ‘**Update Pipeline**’ icon.



x) A notification message appears to inform that the pipeline is successfully updated.


10 Pipeline updation success.

xi) Open the HDFS Reader component by a click on it.
xii) The HDFS Reader gets the ‘**Configuration**’ properties tab.



HDFS Reader

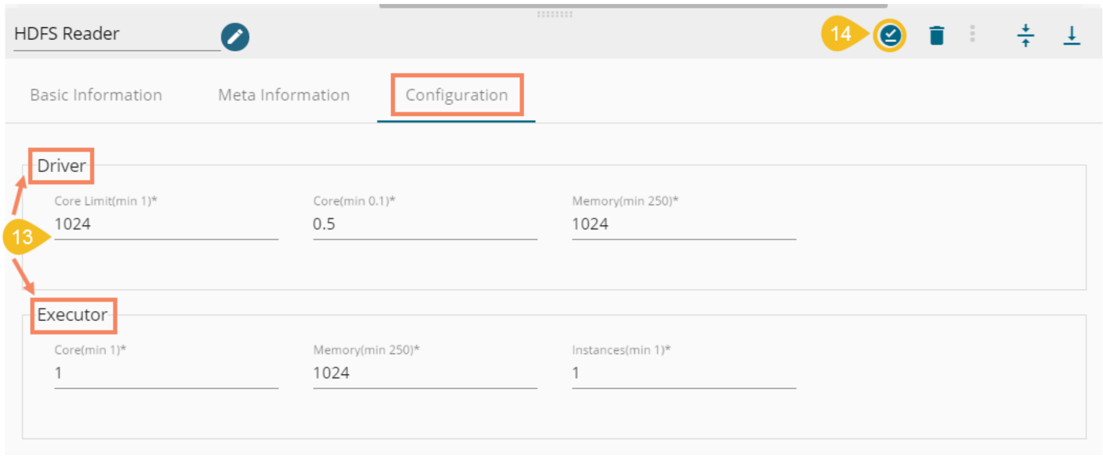
Basic Information | Meta Information | Configuration

Driver

Core Limit(min 1)*: 1024
Core(min 0.1)*: 0.5
Memory(min 250)*: 1024

xiii) Modify the values for the Driver or Executor section from the Configuration tab.

xiv) Click the ‘**Save Component in Storage**’ ✓ icon.



HDFS Reader

Basic Information | Meta Information | Configuration

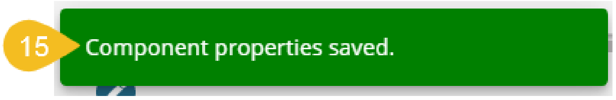
Driver

Core Limit(min 1)*: 1024
Core(min 0.1)*: 0.5
Memory(min 250)*: 1024

Executor

Core(min 1)*: 1
Memory(min 250)*: 1024
Instances(min 1)*: 1

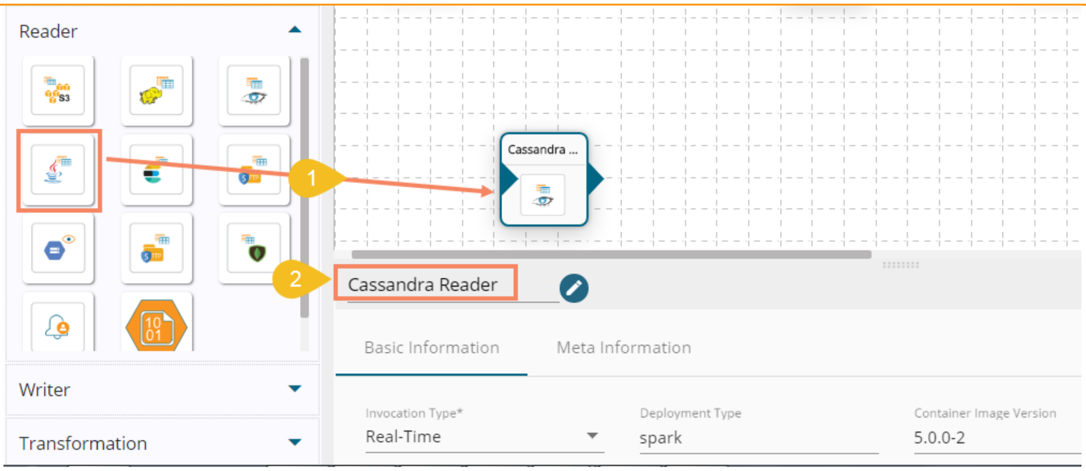
xv) A message appears to notify about the component properties.


15 Component properties saved.

xvi) The Properties get configured for the HDFS Reader and it is ready to be used in the Pipeline Workflow.

8.1.3. Cassandra Reader

i) Drag & Drop the Cassandra Reader component on the Workflow Editor.
ii) Click on the dragged reader component to open the component properties tabs below.

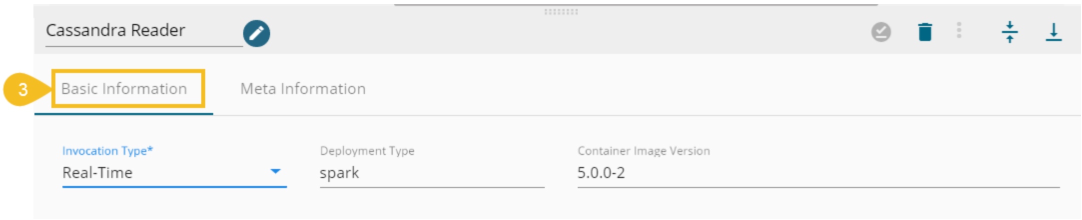


iii) Basic Information: It is the default tab to open for the Cassandra Reader while configuring the component.

a. Select an Invocation type from the drop-down menu to confirm the running mode of the reader component. Select 'Real-Time' or 'Batch' from the drop-down menu.

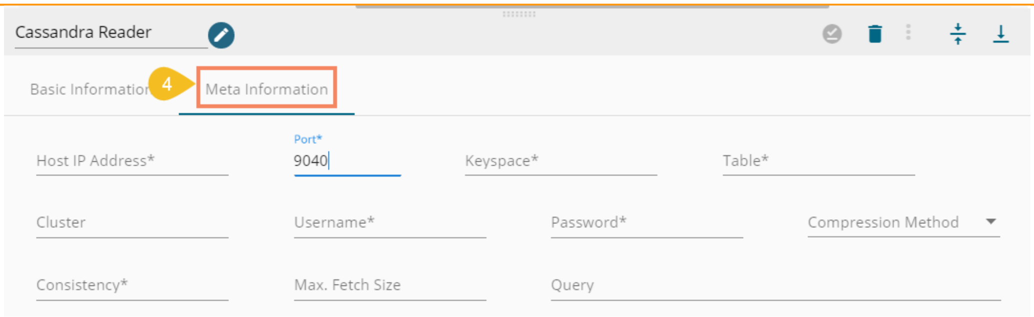
b. Deployment Type: It displays the deployment type for the component. This field comes pre-selected.

c. Container Image Version: It displays the image version for the docker container. This field comes pre-selected.



iv) Open the '**Meta Information**' tab and fill all the connection-specific details for the Cassandra Reader.

1. Host IP Address (*): IP Address
2. Port(*): Host server port number
3. Keyspace(*): Key for the table
4. Table(*): Table Name to read data
5. Cluster: Name of the cluster
6. Username(*): Provide username for login
7. Password(*): provide a valid password for login
8. Compression Method: Select a compression method from the drop-down
9. Consistency: the minimum number of Cassandra nodes that must acknowledge the operation
10. Max. Fetch Size: Max number of records per batch
11. Query: Insert an SQL query (it takes query containing a Join statement as well)


Cassandra Reader

Basic Information 4 Meta Information

Host IP Address*

Port* 9040

Keyspace*

Table*

Cluster

Username*

Password*

Compression Method

Consistency*

Max. Fetch Size

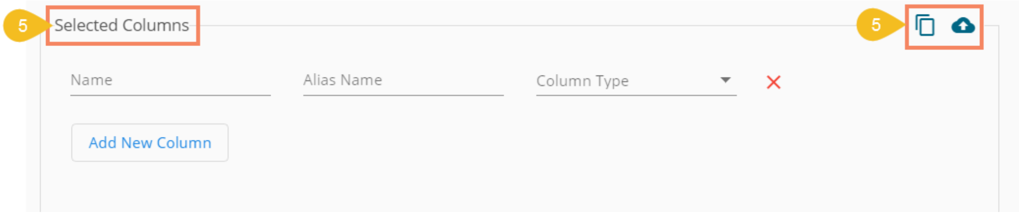
Query

v) Selected Columns: The users can select some specific columns from the table to read data instead of selecting a complete table; this can be achieved via the '**Selected Columns**' section. Select the columns which you want to read and if you want to change the name of the column, then put that name in alias name section otherwise keep alias name same as of column name and then select a Column Type from the drop-down menu.

or

Use '**Download Data**' and '**Upload File**' options to select the desired columns.

1. Upload File: The user can upload the existing system files (CSV, JSON) using the '**Upload File**' ☁ icon (file size must be less than 2 MB).
2. Download Data (Schema): Users can download the schema structure in JSON format by using the 'Download Data' ⧉ icon.


5 Selected Columns

5

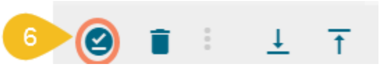
Name

Alias Name

Column Type

Add New Column

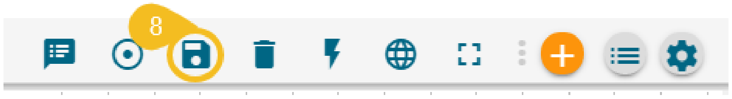
vi) After doing all the configurations click the '**Save Component in Storage**' ✓ icon provides in the reader configuration panel to save the component.



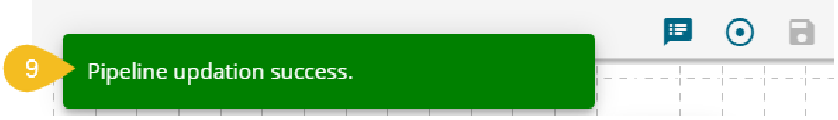
vii) A notification message appears to inform about the component configuration success.

7 Component properties saved.

viii) Click the '**Update Pipeline**' 💾 icon.



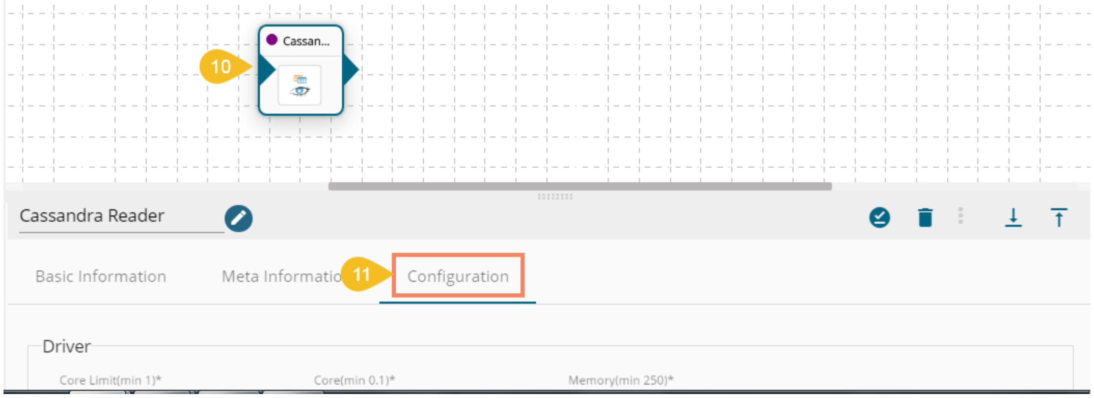
ix) A notification message appears to inform that the pipeline is successfully updated.



Pipeline updation success.

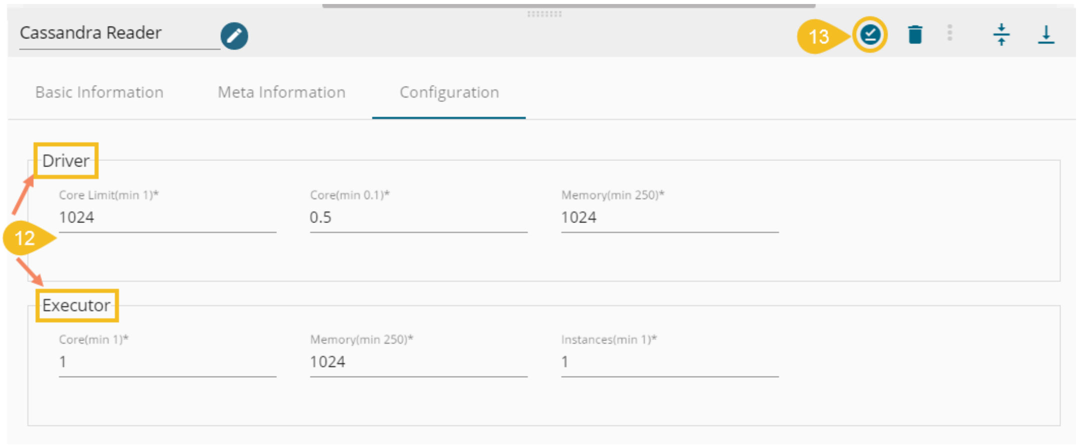
x) Open the Cassandra Reader component by a click on it.

xi) The Cassandra Reader gets the '**Configuration**' properties tab.



xii) Modify the values for the Limit or Request section from the Configuration tab.

xiii) Click the '**Save Component in Storage**' ✓ icon.



Driver

Core Limit(min 1)*: 1024

Core(min 0.1)*: 0.5

Memory(min 250)*: 1024

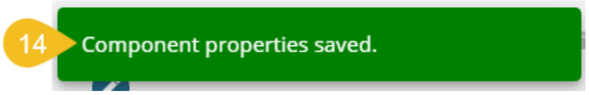
Executor

Core(min 1)*: 1

Memory(min 250)*: 1024

Instances(min 1)*: 1

xiv) A message appears to notify about the component properties.



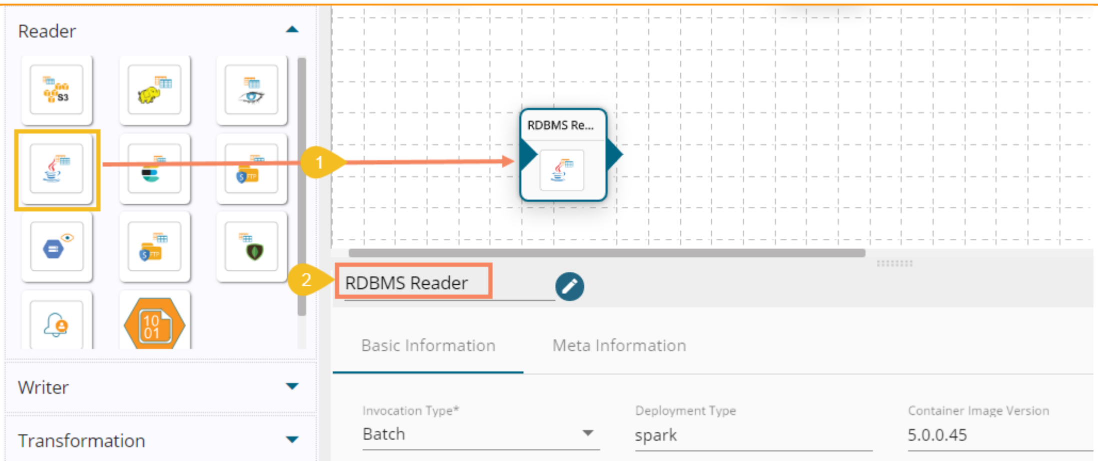
Component properties saved.

xv) The Properties get configured for the Cassandra Reader and it is ready to be used in the Pipeline Workflow.

8.1.4. RDBMS Reader

i) Drag & Drop the JDBC Reader component on the Workflow Editor.

ii) Click on the dragged reader component to open the component properties tabs below.



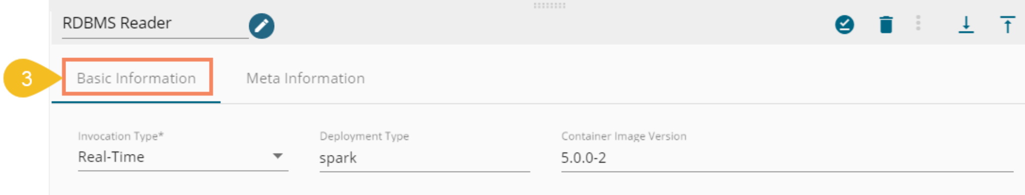
RDBMS Reader

Basic Information | Meta Information

Invocation Type*: Batch
Deployment Type: spark
Container Image Version: 5.0.0.45

iii) Basic Information: It is the default tab to open for the RDBMS Reader while configuring the component.

   d. Select an Invocation type from the drop-down menu to confirm the running mode of the reader component. Select 'Real-Time' or 'Batch' from the drop-down menu.
   e. Deployment Type: It displays the deployment type for the component. This field comes pre-selected.
   f. Container Image Version: It displays the image version for the docker container. This field comes pre-selected.



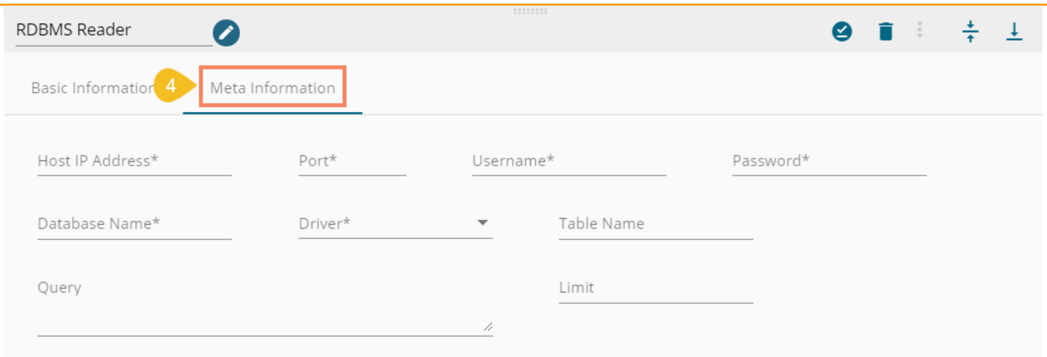
RDBMS Reader

Basic Information | Meta Information

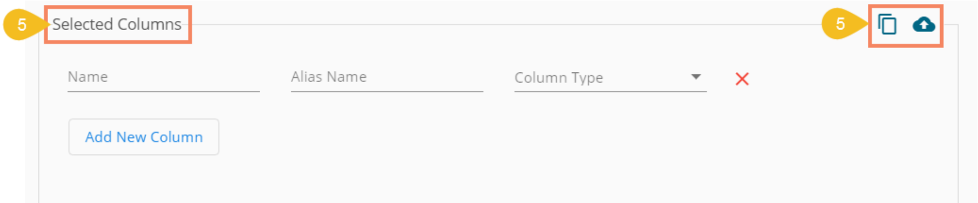
Invocation Type*: Real-Time
Deployment Type: spark
Container Image Version: 5.0.0-2

iv) Open the '**Meta Information**' tab and fill all the connection-specific details for the RDBMS Reader.

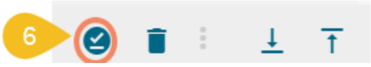
   1. Host IP Address (*): IP Address
   2. Port(*): Host server port number
   3. Username(*): Username for login
   4. Password(*): Password for login
   5. Database Name(*): Provide the Database name
   6. Driver(*): Select Database type (MYSQL, MSSQL, Oracle, Postgres)
   7. Table Name: Provide the table name to read data
   8. Query: Insert an SQL query (it takes query containing a Join statement as well)
   9. Limit: Set limit for the number of records



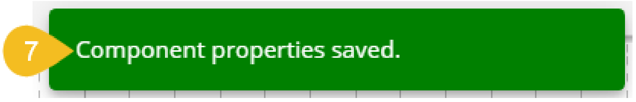
v) Selected Columns: The users can select some specific columns from the table to read data instead of selecting a complete table; this can be achieved via the '**Selected Columns**' section. Select the columns which you want to read and if you want to change the name of the column, then put that name in alias name section otherwise keep alias name same as of column name and then select a Column Type from the drop-down menu.
or
Use '**Download Data**' and '**Upload File**' options to select the desired columns.
1. Upload File: The user can upload the existing system files (CSV, JSON) using the '**Upload File'** icon (file size must be less than 2 MB).
2. Download Data (Schema): Users can download the schema structure in JSON format by using the 'Download Data' icon.



vi) After doing all the configurations click the '**Save Component in Storage**' icon provides in the reader configuration panel to save the component.

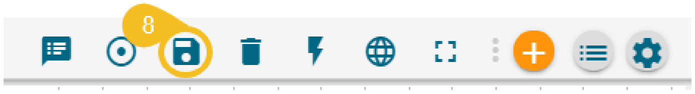


vii) A notification message appears to inform about the component configuration success.

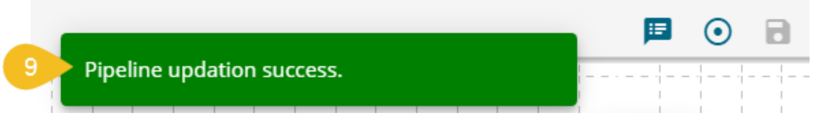


Component properties saved.

viii) Click the 'Update Pipeline' icon.



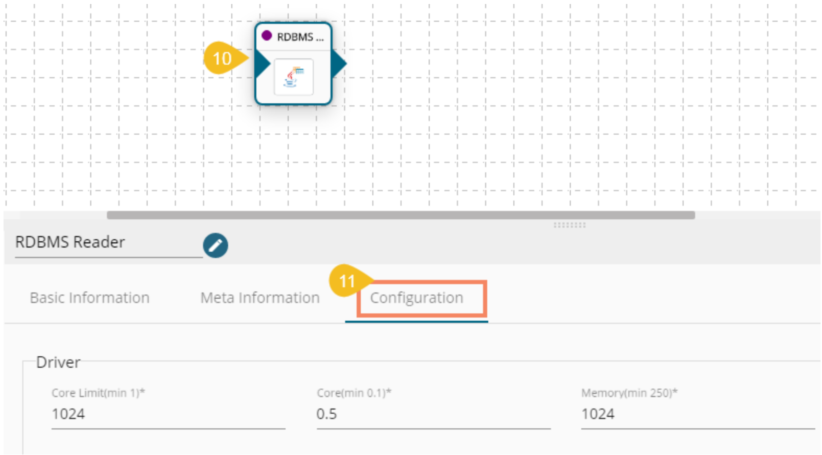
ix) A notification message appears to inform that the pipeline is successfully updated.



Pipeline updation success.

x) Open the RDBMS Reader component by a click on it.

xi) The RDBMS Reader gets the 'Configuration' properties tab.

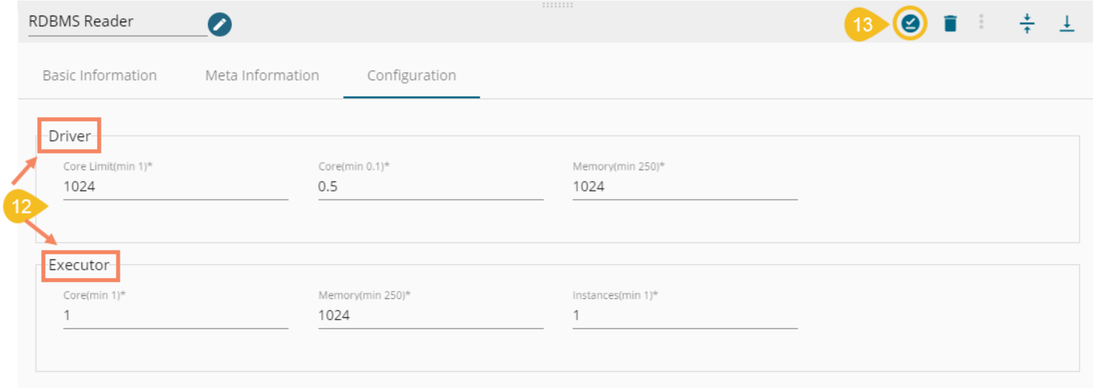


Driver

Core Limit(min 1)*: 1024
Core(min 0.1)*: 0.5
Memory(min 250)*: 1024

xii) Modify the values for the Driver or Executor section from the Configuration tab.

xiii) Click the '**Save Component in Storage**' ✓ icon.



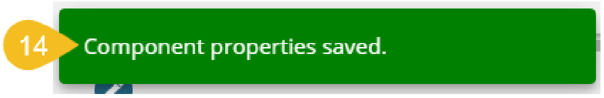
Driver

Core Limit(min 1)*: 1024
Core(min 0.1)*: 0.5
Memory(min 250)*: 1024

Executor

Core(min 1)*: 1
Memory(min 250)*: 1024
Instances(min 1)*: 1

xiv) A message appears to notify about the component properties.



Component properties saved.

xv) The RDBMS Reader component configuration gets completed and it is ready to be used in the Pipeline Workflow.

8.1.5. ES Reader

i) Drag & drop the ES Reader component on the Workflow Editor.

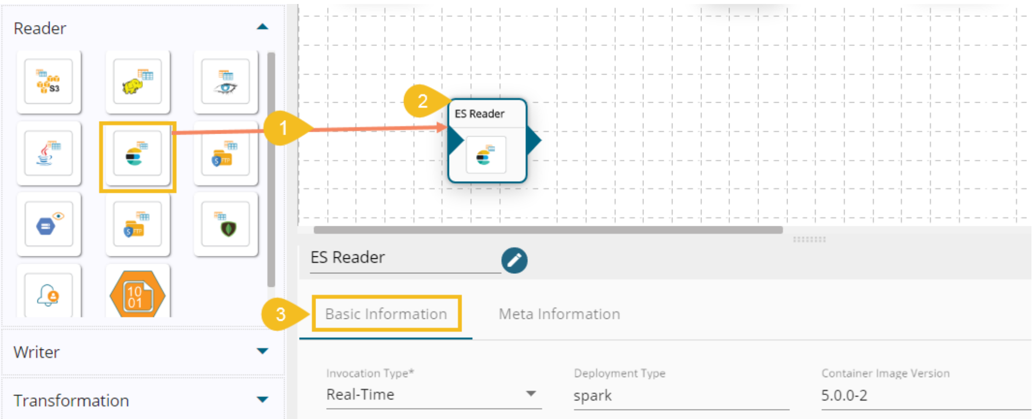
ii) Click on the dragged reader component to open the component properties tabs below.

iii) Basic Information: It is the default tab to open for the ES Writer while configuring the component.

a. Select an Invocation type from the drop-down menu to confirm the running mode of the reader component. Select 'Real-Time' or 'Batch' from the drop-down menu.

b. Deployment Type: It displays the deployment type for the component. This field comes pre-selected.

c. Container Image Version: It displays the image version for the docker container. This field comes pre-selected.



ES Reader

Basic Information | Meta Information

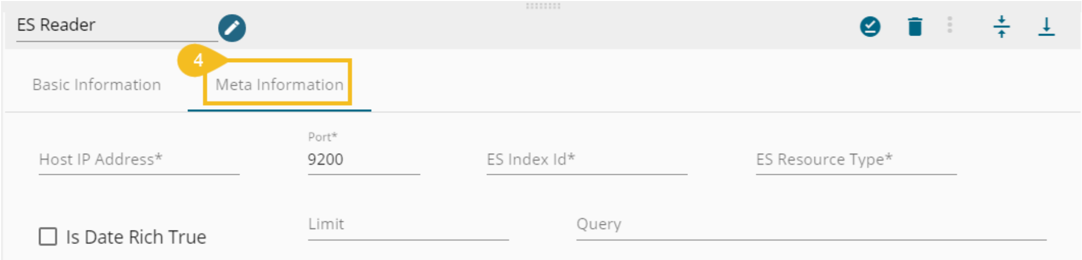
Invocation Type*: Real-Time

Deployment Type: spark

Container Image Version: 5.0.0-2

iv) Open the 'Meta Information' tab and fill all the connection-specific details of ES Reader.

1. Host IP Address (*): Database Host Address
2. Port(*): Database Host port
3. ES Index Id(*): Id of the elastic search index
4. ES Resource Type: Type of the elastic search index (can be given same as Id)
5. Is Data-Rich True: Required while reading data from indexes having dates
6. Limit: Set limit for the number of records
7. Query: Provide a valid query to fetch data from the dataset.



ES Reader

Basic Information | Meta Information

Host IP Address*

Port*: 9200

ES Index Id*

ES Resource Type*

□ Is Date Rich True

Limit

Query

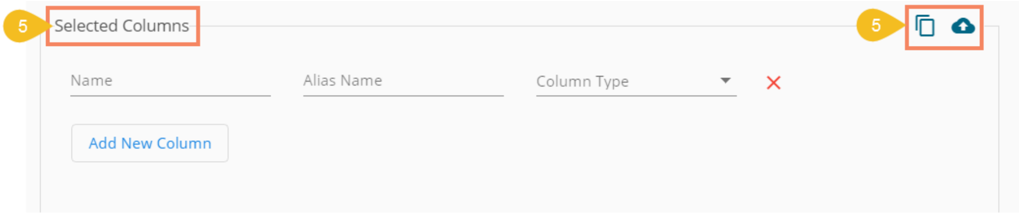
v) Selected Columns: The users can select some specific columns from the table to read data instead of selecting a complete table; this can be achieved via the 'Selected Columns' section. Select the columns which you want to read and if you want to change the name of the column, then put that name in alias name section otherwise keep alias name same as of column name and then select a Column Type from the drop-down menu.

or

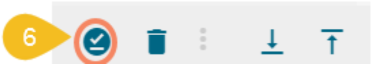
Use 'Download Data' and 'Upload File' options to select the desired columns.

1. Upload File: The user can upload the existing system files (CSV, JSON) using the 'Upload File' ☁ icon (file size must be less than 2 MB).

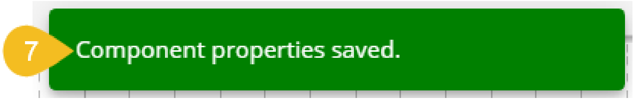
2. Download Data (Schema): Users can download the schema structure in JSON format by using the 'Download Data' ⧉ icon.



vi) Click the '**Save Component in Storage**' ✓ icon provides to save the component properties.

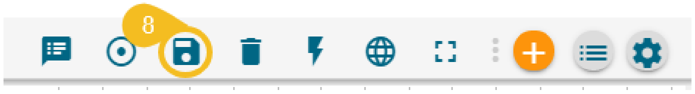


vii) A notification message appears to inform about the component configuration success.

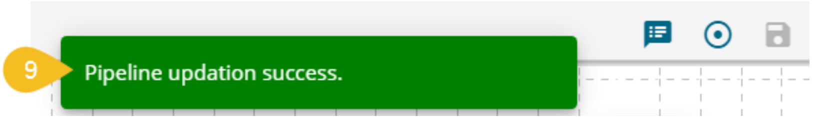


Component properties saved.

viii) Click the '**Update Pipeline**' 💾 icon.



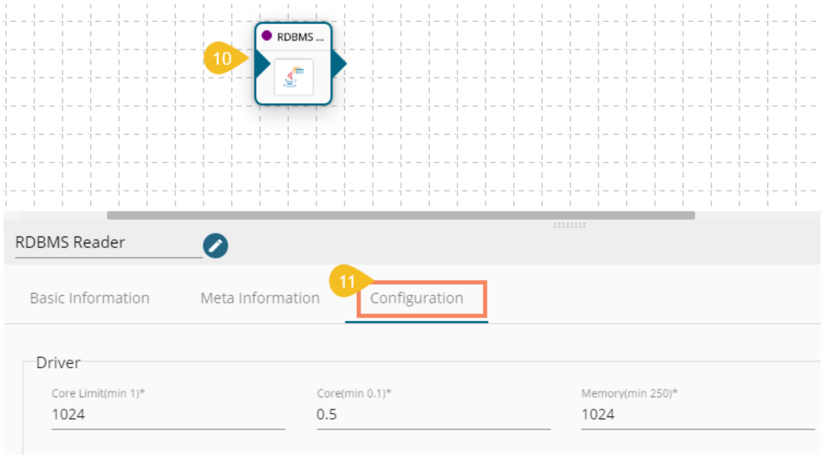
ix) A notification message appears to inform that the pipeline is successfully updated.



Pipeline updation success.

x) Open the ES Reader component by a click on it.

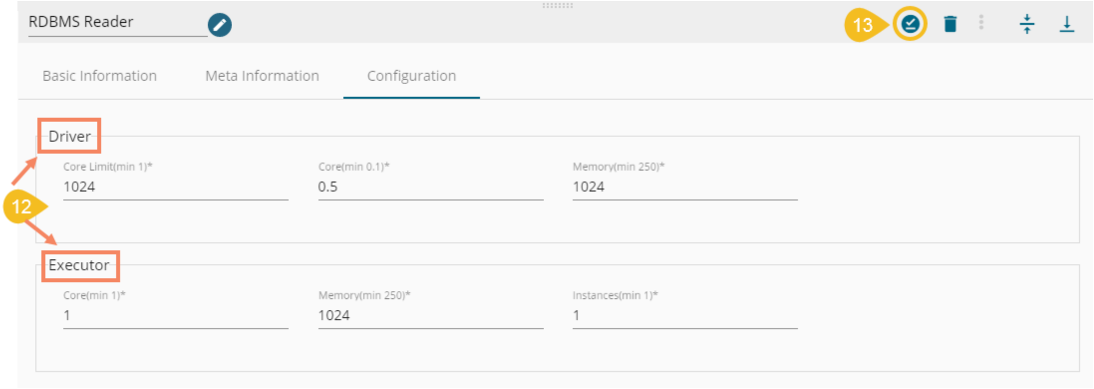
xi) The ES Reader gets the '**Configuration**' properties tab.



Core Limit(min 1)* 1024
Core(min 0.1)* 0.5
Memory(min 250)* 1024

xii) Modify the values for the Driver or Executor section from the Configuration tab.

xiii) Click the '**Save Component in Storage**' ✓ icon.



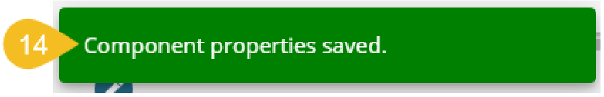
Driver

Core Limit(min 1)*	Core(min 0.1)*	Memory(min 250)*
1024	0.5	1024

Executor

Core(min 1)*	Memory(min 250)*	Instances(min 1)*
1	1024	1

xiv) A message appears to notify about the component properties.



Component properties saved.

xv) The ES Reader component configuration gets completed and it is ready to be used in the Pipeline Workflow.

8.1.6. SFTP Reader

i) Drag & Drop SFTP Reader component on the Workflow Editor.

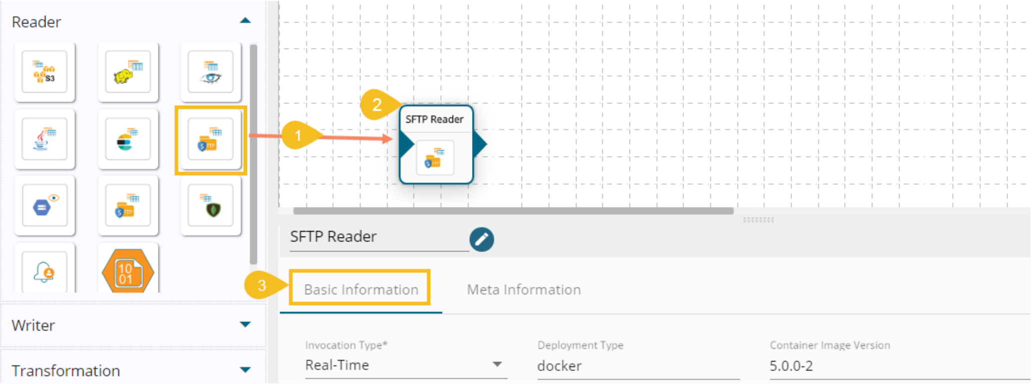
ii) Click on the dragged reader component to open the component properties tabs below.

iii) Basic Information: It is the default tab to open for the SFTP Reader while configuring the component.

a. Select an Invocation type from the drop-down menu to confirm the running mode of the reader component. Select 'Real-Time' or 'Batch' from the drop-down menu.

b. Deployment Type: It displays the deployment type for the component. This field comes pre-selected.

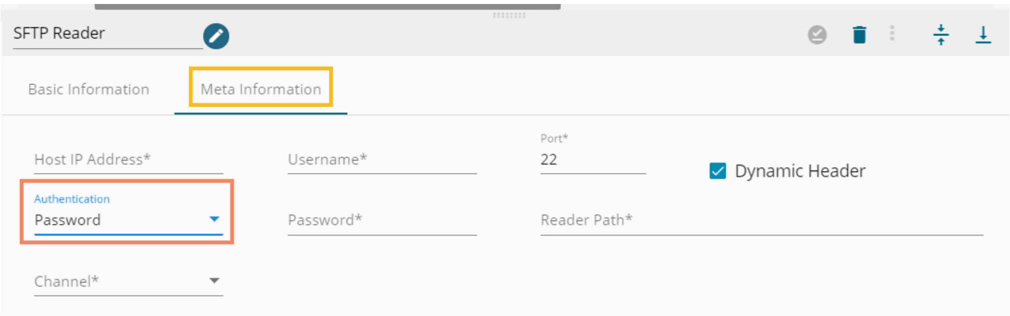
c. Container Image Version: It displays the image version for the docker container. This field comes pre-selected.



Invocation Type*	Deployment Type	Container Image Version
Real-Time	docker	5.0.0-2

iv) Open the '**Meta Information**' tab and fill all the connection-specific details for the SFTP Reader.

1. Host IP Address (*): IP address of SFTP location.
2. Username(*): Username of SFTP location.
3. Port(*): Port of SFTP server (default:22).
4. Dynamic Header: Enable this option by putting a checkmark in the box.
5. Authentication(*): Select an authentication option for login to the SFTP server using the drop-down menu. The field gets displayed based on the Authentication option.
    a. Password: configure the following fields that appear after selecting the 'Password' option as authentication.
        i. Password: Provide a valid password in the 'Password' fields to log in.

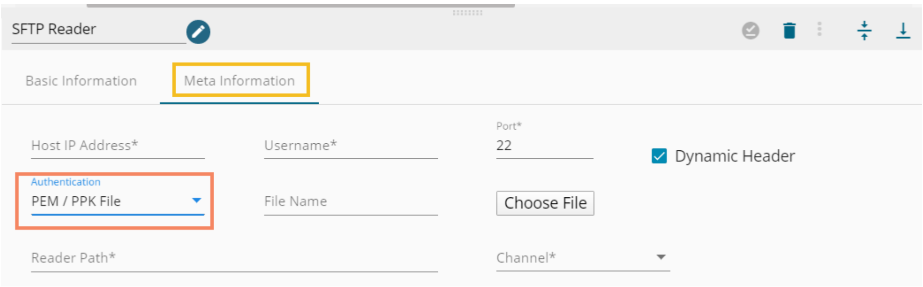


Port*: 22

☑ Dynamic Header

Authentication: Password

    b. PEM/PPK (file): Select PEM/PPK file database host port for login to SFTP server
        i. File Name: Provide a file name
        ii. Choose File: select a PEM/PPK file using this option

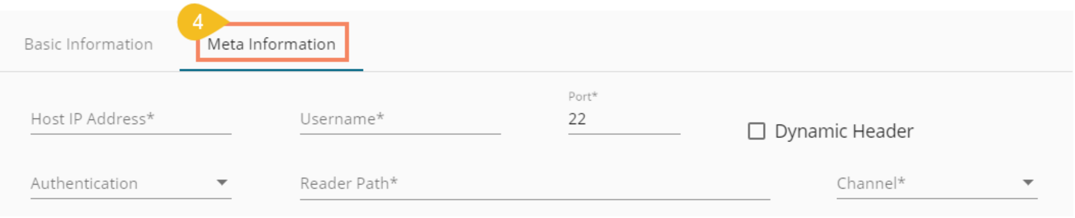


Port*: 22

☑ Dynamic Header

Authentication: PEM / PPK File

6. Reader Path(*) – complete path (absolute path) for the folder (from where the new files need to be read)
7. Channel(*) – the supported channel is SFTP.



Port*: 22

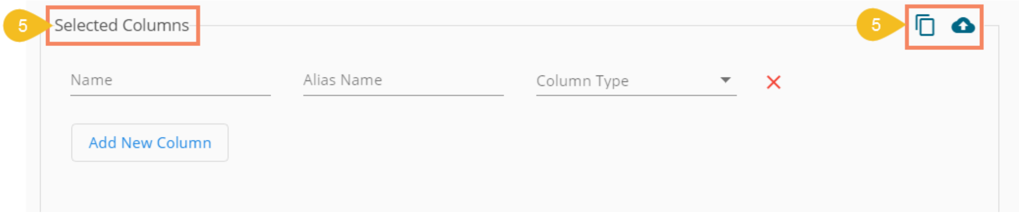
☐ Dynamic Header

v) Selected Columns: The users can select some specific columns from the table to read data instead of selecting a complete table; this can be achieved via the ‘Selected Columns’ section. Select the columns which you want to read and if you want to change the name of the column, then put that name in alias name section otherwise keep alias name same as of column name and then select a Column Type from the drop-down menu.

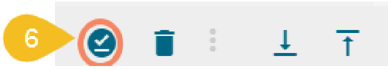
or

Use ‘**Download Data**’ and ‘**Upload File**’ options to select the desired columns.

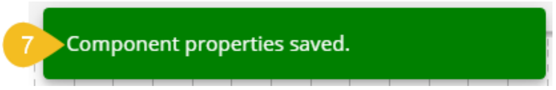
1. Upload File: The user can upload the existing system files (CSV, JSON) using the ‘Upload File’ icon (file size must be less than 2 MB).
2. Download Data (Schema): Users can download the schema structure in JSON format by using the ‘Download Data’ icon.



vi) After doing all the configurations click the ‘**Save Component in Storage’** icon provides in the reader configuration panel to save the component.

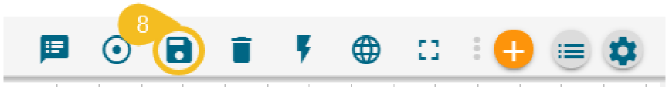


vii) A notification message appears to inform about the component configuration success.

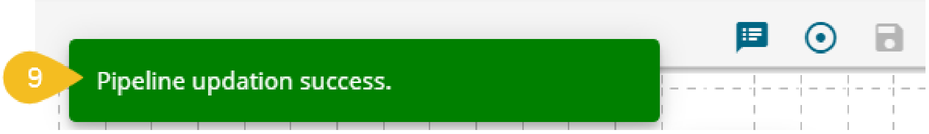


Component properties saved.

viii) Click the ‘**Update Pipeline’** icon.



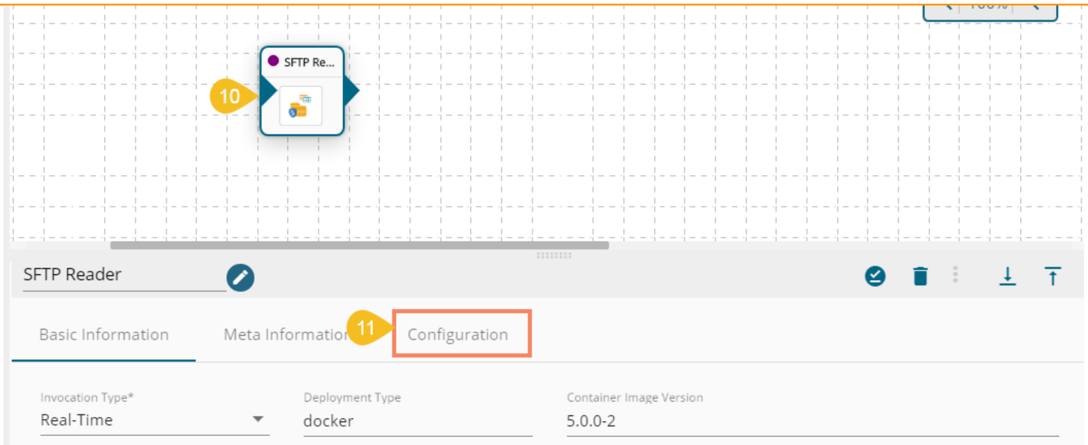
ix) A notification message appears to inform that the pipeline is successfully updated.



Pipeline updation success.

x) Open the SFTP Reader component by a click on it.

xi) The SFTP Reader gets the ‘Configuration’ properties tab.



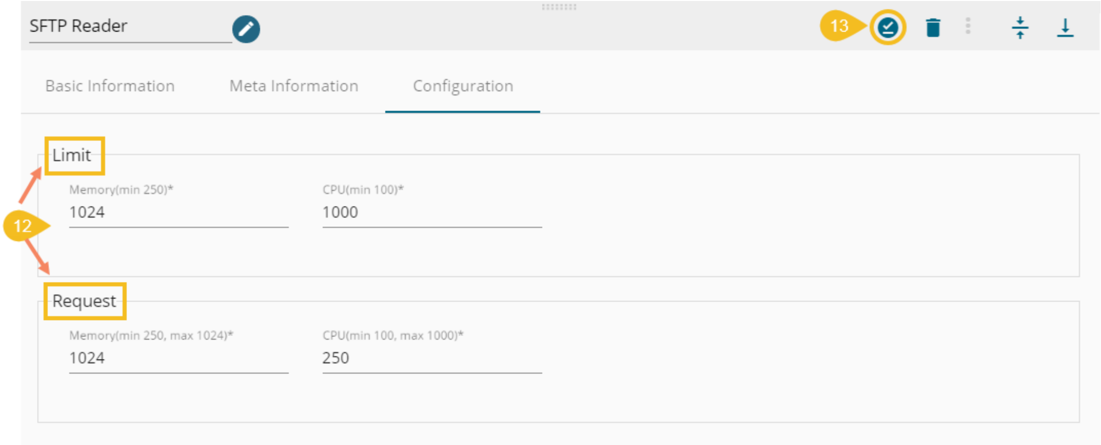
Invocation Type*: Real-Time

Deployment Type: docker

Container Image Version: 5.0.0-2

xii) Modify the values for the Limit or Request section from the Configuration tab.

xiii) Click the ‘Save Component in Storage’ ✓ icon.



Limit

Memory(min 250)*: 1024

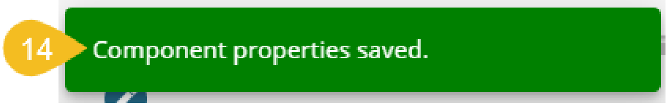
CPU(min 100)*: 1000

Request

Memory(min 250, max 1024)*: 1024

CPU(min 100, max 1000)*: 250

xiv) A message appears to notify about the component properties.



Component properties saved.

xv) The SFTP Reader component configuration gets completed and it is ready to be used in the Pipeline Workflow.

8.1.7. GCS Reader

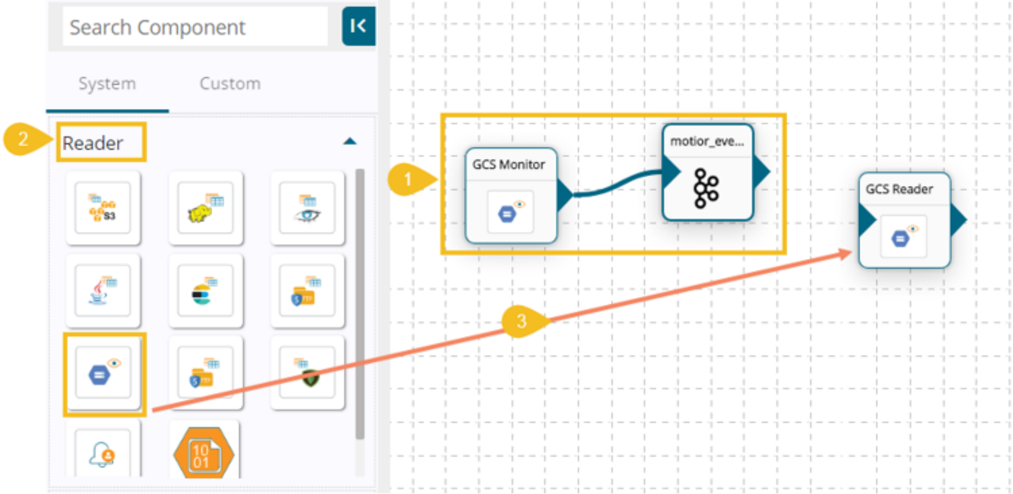
GCS Reader pulls data from the GCS Monitor, so the first step is to implement GCS Monitor.

Note: The users can refer to the GCS Monitor section of this document for the details.

i) Navigate to the Pipeline Workflow Editor page for an existing pipeline workflow with GCS Monitor and Event component.

ii) Open the Reader section of the Component Pallet.

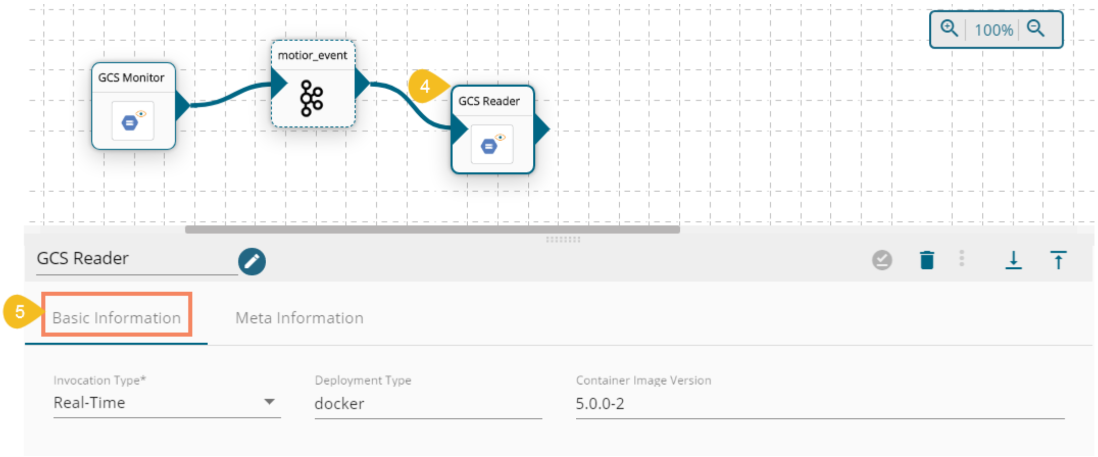
iii) Drag the GCS Reader to the Workflow Editor.



iv) Click on the dragged GCS Reader component to get component properties tabs below.

v) Basic Information: It is the default tab to open for the GCS Reader while configuring the component.

   a. Select an Invocation type from the drop-down menu to confirm the running mode of the reader component. Select 'Real-Time' or 'Batch' from the drop-down menu.
   b. Deployment Type: It displays the deployment type for the component. This field comes pre-selected.
   c. Container Image Version: It displays the image version for the docker container. This field comes pre-selected.



GCS Reader

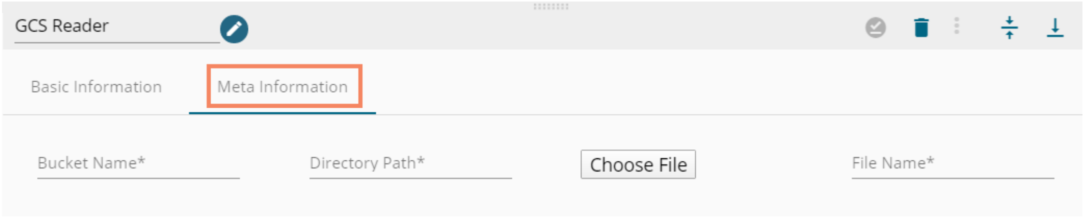
Invocation Type* Real-Time

Deployment Type docker

Container Image Version 5.0.0-2

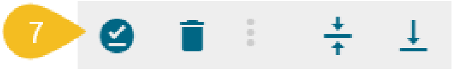
vi) Provide the following Meta Information for the GCS Reader component:

   a. Bucket Name: Destination bucket name (Copy Bucket Name of the GCS Monitor)
   b. Directory Path: Destination copy folder path (Copy Directory Path of the GCS Monitor)
   c. Choose File: Upload the same Service account keys using the JSON file
   d. File Name: It displays the file name of the uploaded file

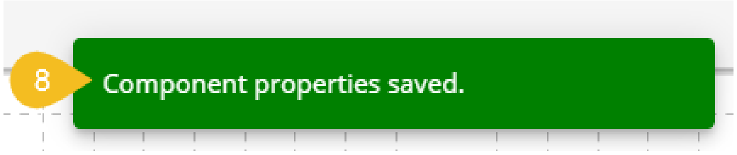


Note: Provide the same destination configuration information that was used in the GCS Monitor.

vii) After doing all the configurations click the ‘**Save Component in Storage**’ icon provides in the reader configuration panel to save the component.

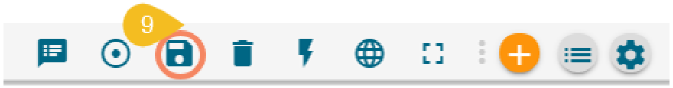


viii) A message appears to confirm the component properties update.

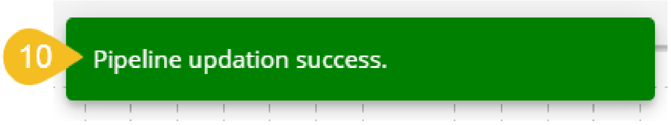


Component properties saved.

ix) Click the ‘Update Pipeline’ icon.



x) A message appears to notify about the success of the pipeline update.

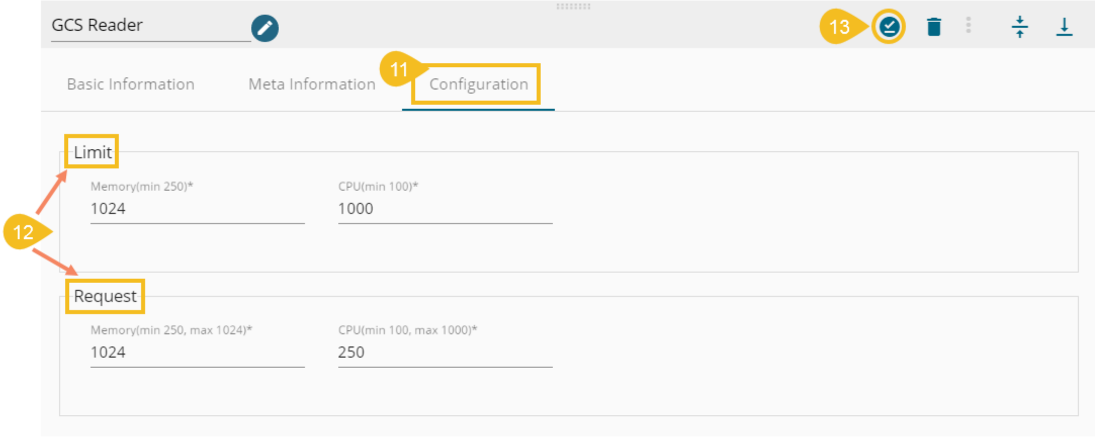


Pipeline updation success.

xi) The ‘**Configuration**’ tab gets accessible after updating the pipeline.

xii) Modify the values for the Limit or Request section from the Configuration tab.

xiii) Click the ‘**Save Component in Storage**’ icon.



GCS Reader

Limit

Memory(min 250)* 1024

CPU(min 100)* 1000

Request

Memory(min 250, max 1024)* 1024

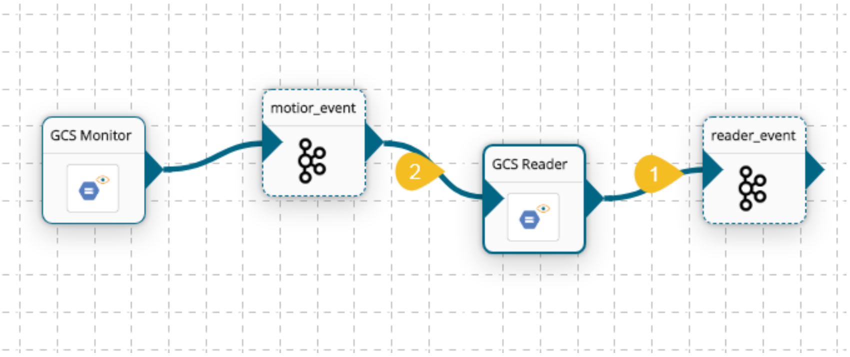
CPU(min 100, max 1000)* 250

xiv) The GCS Reader component configuration gets completed and it is ready to be used in the Pipeline Workflow.

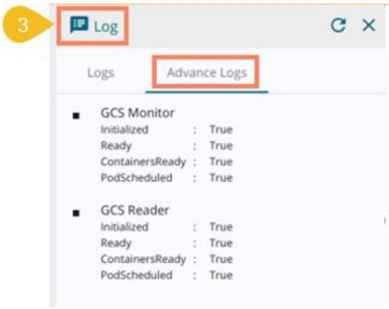
GCS Reader in a Workflow

i) Create a new Event to store all the read data and connect it with the dragged GCS Reader component.

ii) Connect the GCS Reader to the existing GCS Monitor Pipeline to create the below given workflow.



iii) Open the Advanced Log Tab to check the component status.

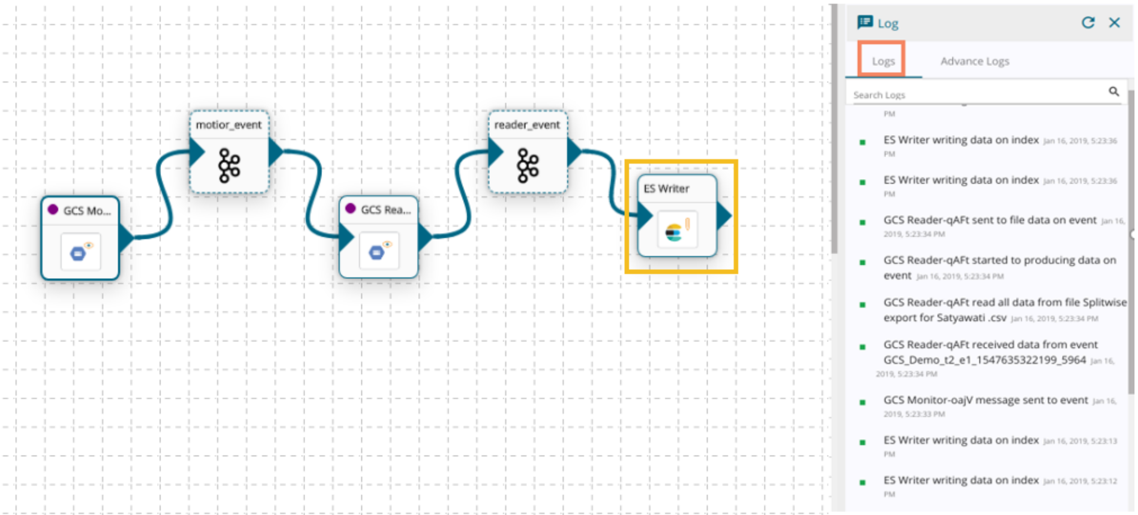


```
■ GCS Monitor
  Initialized     :  True
  Ready           :  True
  ContainersReady :  True
  PodScheduled    :  True

■ GCS Reader
  Initialized     :  True
  Ready           :  True
  ContainersReady :  True
  PodScheduled    :  True
```

iv) Upload a CSV or JSON file on monitoring location and check the Logs section displaying the ongoing operations.

v) Connect a Writer component to store the processed data (E.g., the ES Writer component is used in this pipeline workflow).



```
PM
■ ES Writer writing data on index Jan 16, 2019, 5:23:36 PM
■ ES Writer writing data on index Jan 16, 2019, 5:23:36 PM
■ GCS Reader-qAFt sent to file data on event Jan 16, 2019, 5:23:34 PM
■ GCS Reader-qAFt started to producing data on event Jan 16, 2019, 5:23:34 PM
■ GCS Reader-qAFt read all data from file Splitwise export for Satyawati .csv Jan 16, 2019, 5:23:34 PM
■ GCS Reader-qAFt received data from event GCS_Demo_t2_e1_1547635322199_5964 Jan 16, 2019, 5:23:34 PM
■ GCS Monitor-oajV message sent to event Jan 16, 2019, 5:23:33 PM
■ ES Writer writing data on index Jan 16, 2019, 5:23:13 PM
■ ES Writer writing data on index Jan 16, 2019, 5:23:12 PM
```

Note: GCS Reader component gets a message from an Event component that was initially sent by the GCS Monitor to the Event, so the users must configure the GCS Monitor component before using the GCS Reader component.

8.1.8. SFTP Stream Reader

i) Drag & Drop the SFTP Reader component on the Workflow Editor.

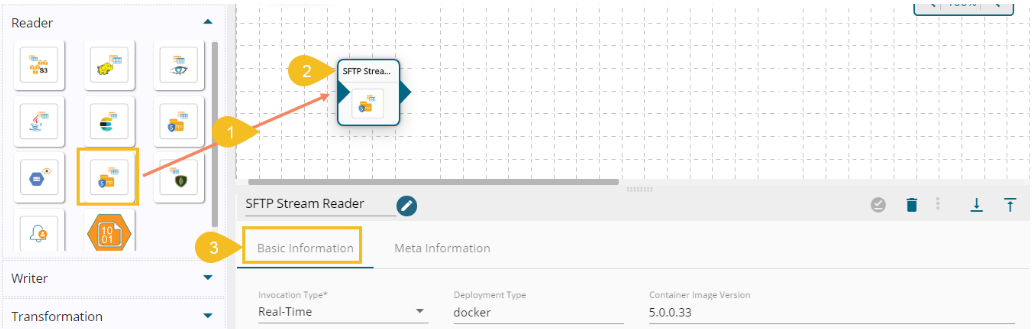
ii) Click on the dragged reader component to open the component properties tabs below.

iii) Basic Information: It is the default tab to open for the SFTP Stream Reader while configuring the component.

a. Select an Invocation type from the drop-down menu to confirm the running mode of the reader component. Select ‘Real-Time’ or ‘Batch’ from the drop-down menu.

b. Deployment Type: It displays the deployment type for the component. This field comes pre-selected.

c. Container Image Version: It displays the image version for the docker container. This field comes pre-selected.



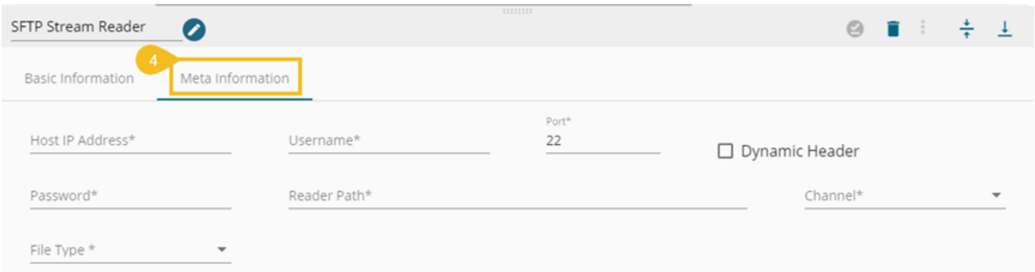
Invocation Type* Real-Time

Deployment Type docker

Container Image Version 5.0.0.33

iv) Open the ‘**Meta Information**’ tab and provide the connection-specific information.

1. Host IP Address (*): IP address of SFTP location.
2. Username(*): Username of SFTP location.
3. Port(*): Port of SFTP server (default:22).
4. Dynamic Header: Enable this option by putting a checkmark in the box.
5. Password(*): Provide a valid password in the ‘Password’ field to login.
6. Reader Path(*): complete path (absolute path) for the folder (from where the new files need to be read)
7. Channel(*): the supported channel is SFTP.
8. File Type(*): Select a file from the drop-down menu.



Host IP Address*

Username*

Port* 22

☐ Dynamic Header

Password*

Reader Path*

Channel*

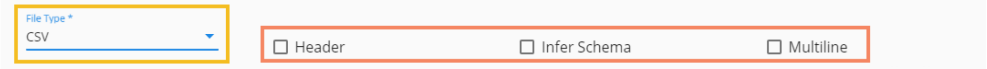
File Type *

Note: The user needs to configure some extra fields based on the selected File Type as displayed below.

a. The following fields appear if the selected file type is CSV

1. Header: Put a checkmark in the box to get Column names/ headers from the CSV file.
2. Infer Schema: Put a checkmark in the box to get inferring datatypes.

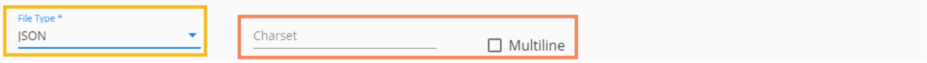
3. Multiline: Enable this option to get a column with multiple lines in a single cell of the data. E.g., the address column can have multiple lines in a single cell.


File Type *
CSV

☐ Header ☐ Infer Schema ☐ Multiline

b. The following fields appear if the selected file type is JSON

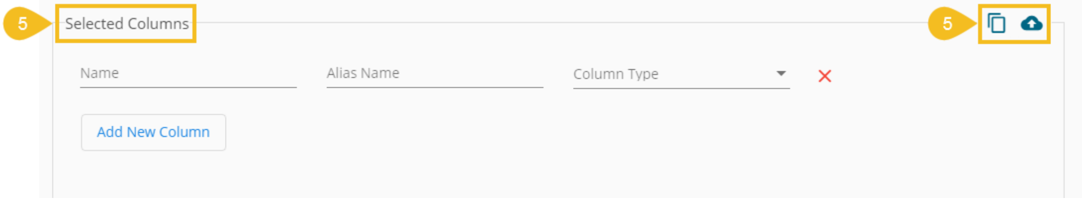
1. Charset-Character set encoding, E.g., UTF-8
2. Multiline: Enable this option to get a column with multiple lines in a single cell of the data.


File Type *
JSON

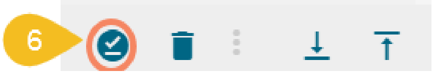
Charset ☐ Multiline

v) Selected Columns: The users can select some specific columns from the table to read data instead of selecting a complete table; this can be achieved via the '**Selected Columns**' section. Select the columns which you want to read and if you want to change the name of the column, then put that name in alias name section otherwise keep alias name same as of column name and then select a Column Type from the drop-down menu.
or
Use '**Download Data**' and '**Upload File**' options to select the desired columns.

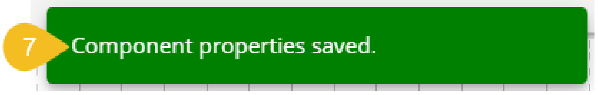
3. Upload File: The user can upload the existing system files (CSV, JSON) using the 'Upload File' icon (file size must be less than 2 MB).
4. Download Data (Schema): Users can download the schema structure in JSON format by using the 'Download Data' icon.



vi) After doing all the configurations click the '**Save Component in Storage**' icon provides in the reader configuration panel to save the component.

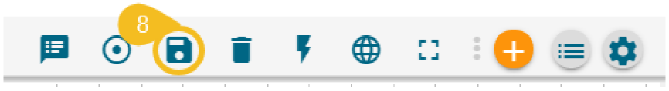


vii) A notification message appears to inform about the component configuration success.

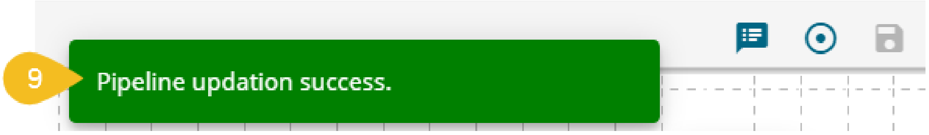


Component properties saved.

viii) Click the '**Update Pipeline**' icon.



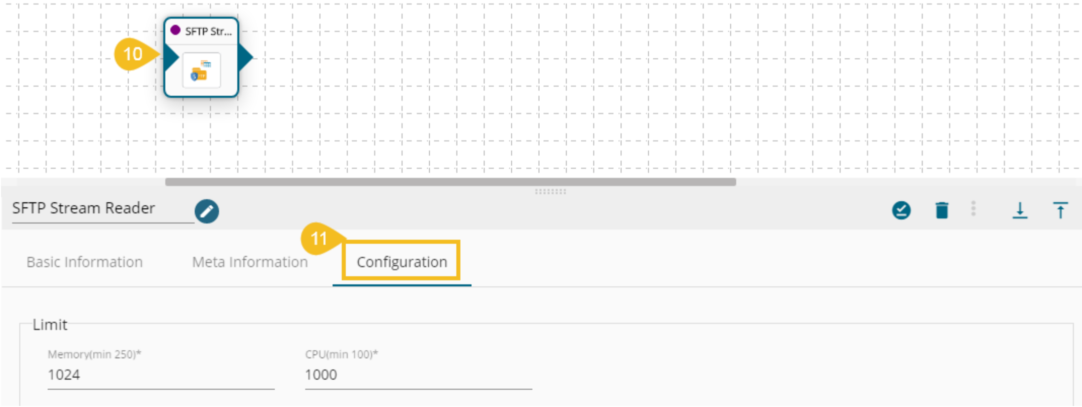
ix) A notification message appears to inform that the pipeline is successfully updated.



Pipeline updation success.

x) Open the SFTP Stream Reader component by a click on it.

xi) The SFTP Stream Reader gets the '**Configuration**' properties tab.



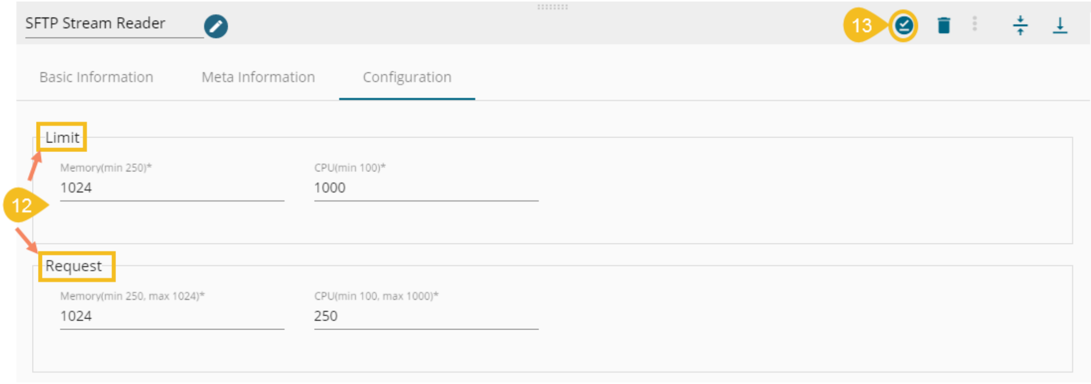
Limit

Memory(min 250)*: 1024

CPU(min 100)*: 1000

xii) Modify the values for the Limit or Request section from the Configuration tab.

xiii) Click the '**Save Component in Storage**' ✓ icon.



Limit

Memory(min 250)*: 1024

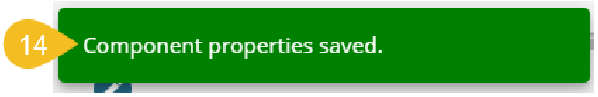
CPU(min 100)*: 1000

Request

Memory(min 250, max 1024)*: 1024

CPU(min 100, max 1000)*: 250

xiv) A message appears to notify about the component properties.



Component properties saved.

xv) The SFTP Stream Reader component configuration gets completed and it is ready to be used in the Pipeline Workflow.

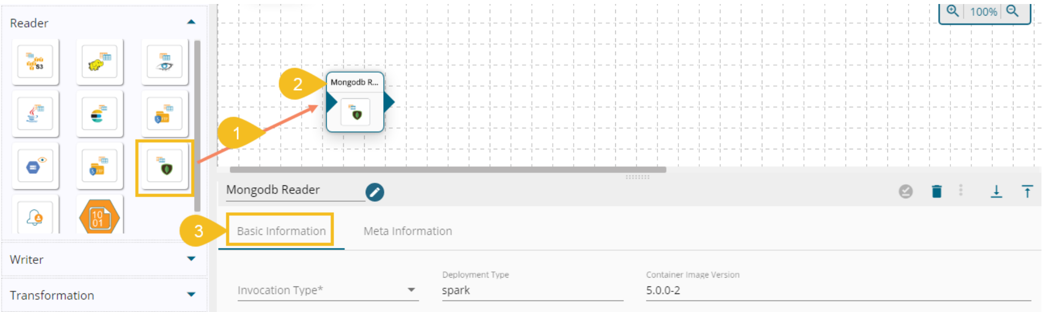
8.1.9. MongoDB Reader

i) Drag & Drop the Mongo DB Reader on the Workflow Editor.

ii) Click on the dragged reader component to open the component properties tabs below.

iii) Basic Information: It is the default tab to open for the Mongo DB reader while configuring the component.

a. Select an Invocation type from the drop-down menu to confirm the running mode of the reader component. Select ‘Real-Time’ or ‘Batch’ from the drop-down menu.
b. Deployment Type: It displays the deployment type for the component. This field comes pre-selected.
c. Container Image Version: It displays the image version for the docker container. This field comes pre-selected.

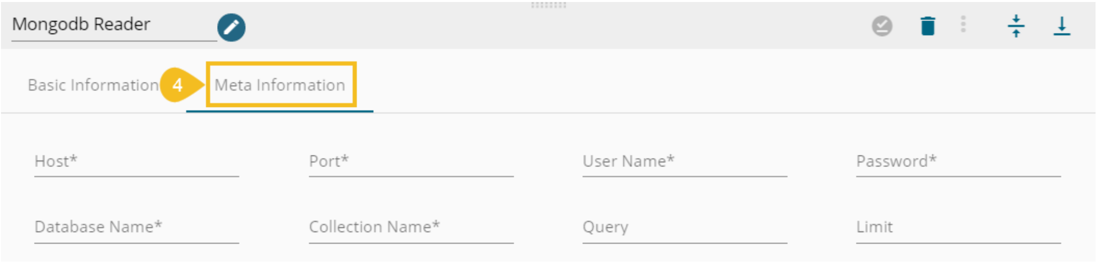


Deployment Type: spark

Container Image Version: 5.0.0-2

iv) Open the ‘**Meta Information**’ tab and fill all the connection-specific details of Mogo DB.

1. Host IP Address(*): Hadoop IP address of the host.
2. Port(*): Port number.
3. Username(*): Provide username.
4. Password(*): Provide a valid password to access the Mongo DB.
5. Database Name (*): Provide the name of the database from where you wish to read data.
6. Collection Name(*): Provide the name of the collection.
7. Query: Insert an SQL query (it takes a query containing a Join statement as well).
8. Limit: Set limit for the number of records.



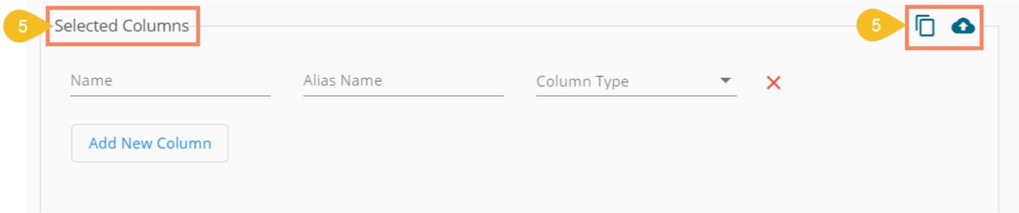
v) Selected Columns: The users can select some specific columns from the table to read data instead of selecting a complete table; this can be achieved via the ‘Selected Columns’ section. Select the columns which you want to read and if you want to change the name of the column, then put that name in alias name section otherwise keep alias name same as of column name and then select a Column Type from the drop-down menu.

or

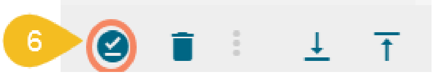
Use ‘**Download Data**’ and ‘**Upload File**’ options to select the desired columns.

1. Upload File: The user can upload the existing system files (CSV, JSON) using the ‘**Upload File**’ ☁ icon (file size must be less than 2 MB).

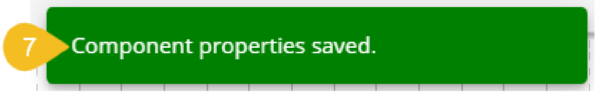
2. Download Data (Schema): Users can download the schema structure in JSON format by using the '**Download Data**' icon.



vi) After doing all the configurations click the '**Save Component in Storage**' icon provides in the reader configuration panel to save the component.

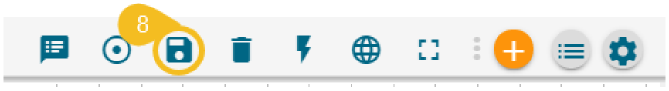


vii) A notification message appears to inform about the component configuration success.

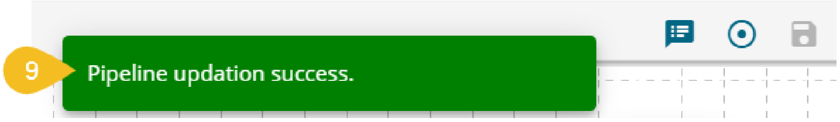


Component properties saved.

viii) Click the '**Update Pipeline**' icon.



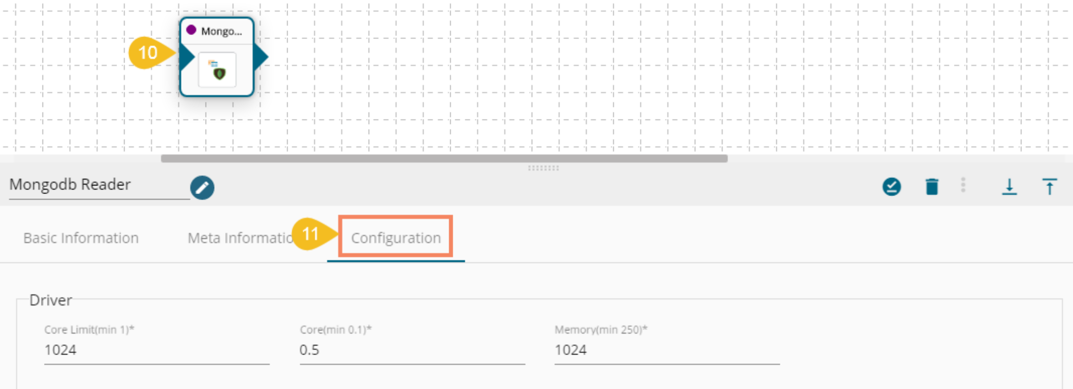
ix) A notification message appears to inform that the pipeline is successfully updated.



Pipeline updation success.

x) Open the Mongo DB Reader component by a click on it.

xi) The Mongo DB Reader gets the '**Configuration**' properties tab.



Mongodb Reader

Driver

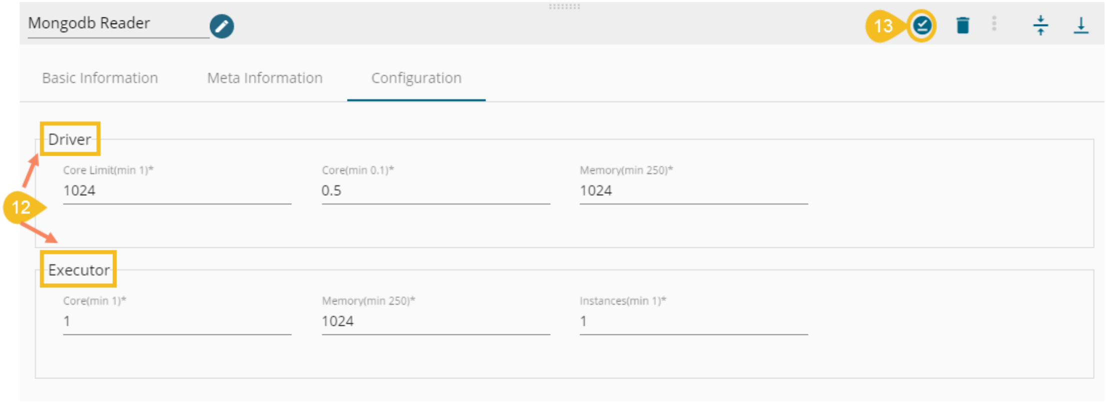
Core Limit(min 1)*: 1024

Core(min 0.1)*: 0.5

Memory(min 250)*: 1024

xii) Modify the values for the Limit or Request section from the Configuration tab.

xiii) Click the '**Save Component in Storage**' ✓ icon.



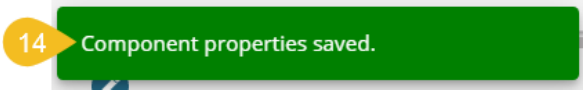
Driver

Core Limit(min 1)*	Core(min 0.1)*	Memory(min 250)*
1024	0.5	1024

Executor

Core(min 1)*	Memory(min 250)*	Instances(min 1)*
1	1024	1

xiv) A message appears to notify about the component properties.



Component properties saved.

xv) The Mongo DB Reader component configuration gets completed and it is ready to be used in the Pipeline Workflow.

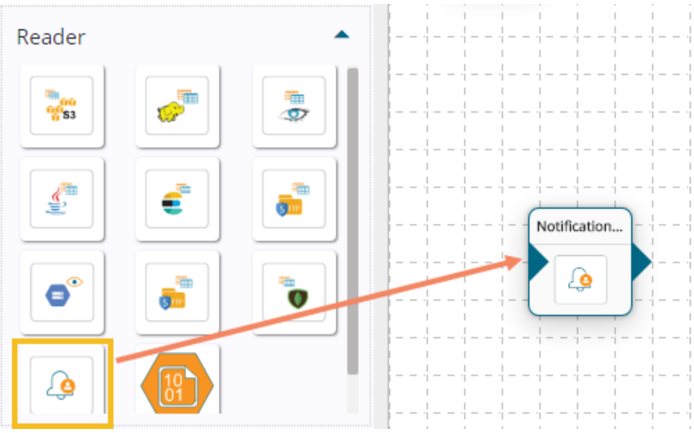
8.1.10. Notification Subscriber

Notification Subscriber Component as the name suggests is meant to subscribe to the messages published as a notification. This can be very helpful in acting in scenarios where the immediate reaction is required in some conditions.

Example: A pipeline is monitoring an IoT sensor that sends the temperature of a machine. If the temperature reaches a certain threshold; the process engineer is to be notified immediately to maintain the same to stabilize the machine. Here, the user can subscribe to the same SNS and further use it in a pipeline for sending this directly to the respective dashboard.

The steps to configure the Notification Subscriber component is described in this section.

i) Navigate to the Pipeline Editor.
ii) Drag the Notification Subscriber from the Reader section provided under the Components Pallet and drop to the Pipeline Workspace.



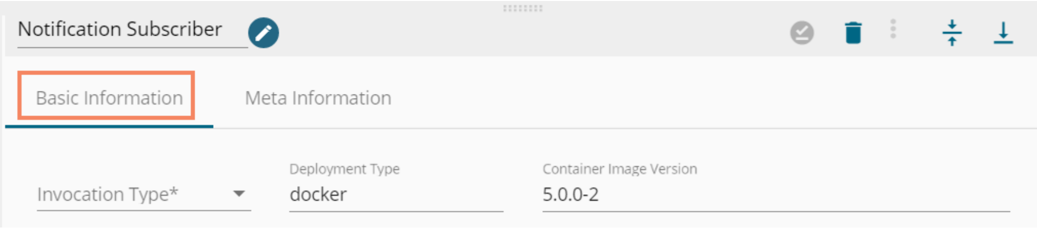
iii) Click the dragged Notification Subscriber component to get the Configuration fields:

iv) The Basic Information tab opens by default.

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

b. Deployment Type: It comes preselected based on the component.

c. Container Image Version: It comes preselected based on the component.



Deployment Type: docker

Container Image Version: 5.0.0-2

v) Open the '**Meta Information**' tab and provide the connection-specific fields.

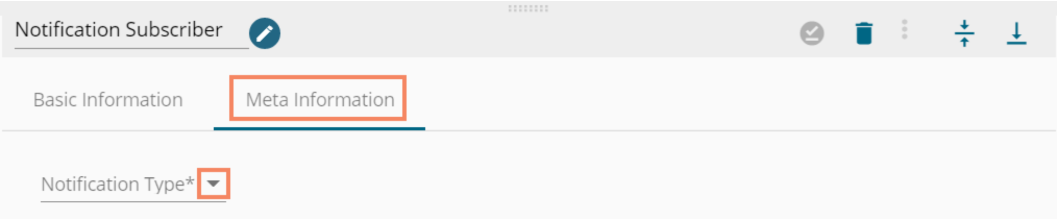
a. Notification Type: Select a notification type using the drop-down menu. The provided options are:

i. SNS

ii. Google Pub/Sub

b. Based on the selected Notification Type the remaining configuration fields get displayed.

c. The user needs to complete the configuration process for the selected notification type and click the '**Save Component in Storage**' ✓ icon to complete the configuration process for the Notification Subscriber.



8.1.10.1. SNS as Notification Type

i) Navigate to the Meta Information tab provided for the Notification Publisher.

ii) Select the 'SNS' as the Notification Type option using the drop-down menu.

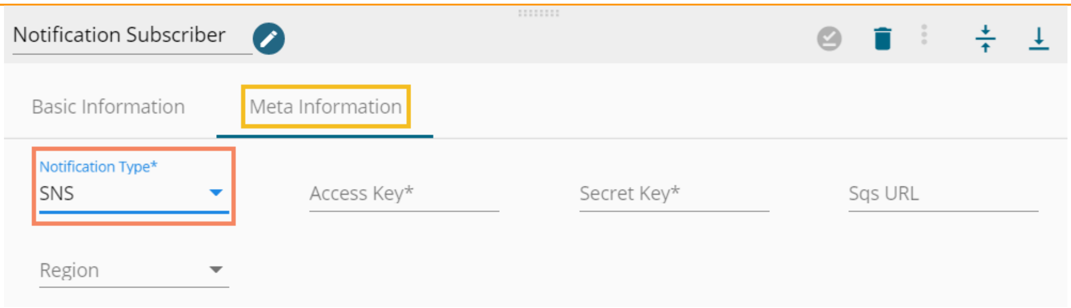
iii) The following configuration fields appear:

a. Access Key: Access keys consist of two parts: an access key ID (for example, AKIAIOSFODNN7EXAMPLE) can be retrieved from AWS key management.

b. Secret Key: A secret access key (for example, wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY) can also be retrieved from the AWS key management.

c. SQS URL: The field gets filled after the topic creation. Navigate to Topics and then click the desired topic.

d. Region: Select an option from the drop-down menu where the topic has been created.

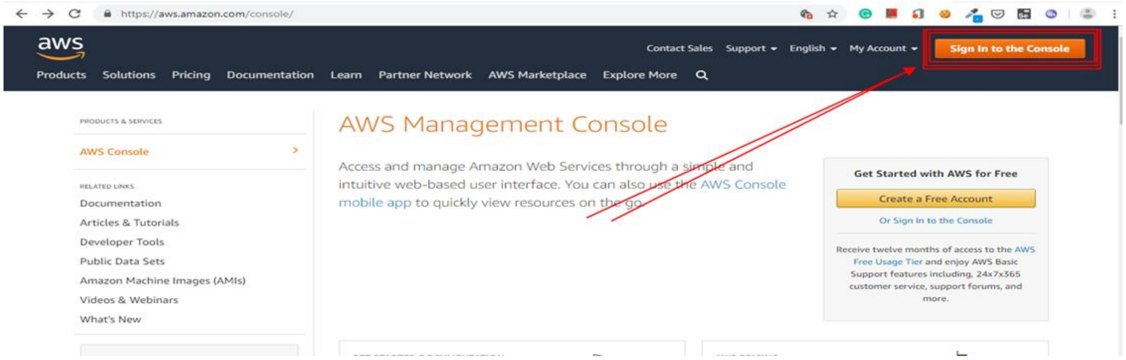


Notification Type*
SNS

iv) The user needs to follow some more steps to create and subscribe to a topic. The below given description explains those steps:

a. Navigate to the URL: https://aws.amazon.com/console/

b. Click the ‘Sign In to the Console’ option.



AWS Management Console

Access and manage Amazon Web Services through a simple and intuitive web-based user interface. You can also use the AWS Console mobile app to quickly view resources on the go.

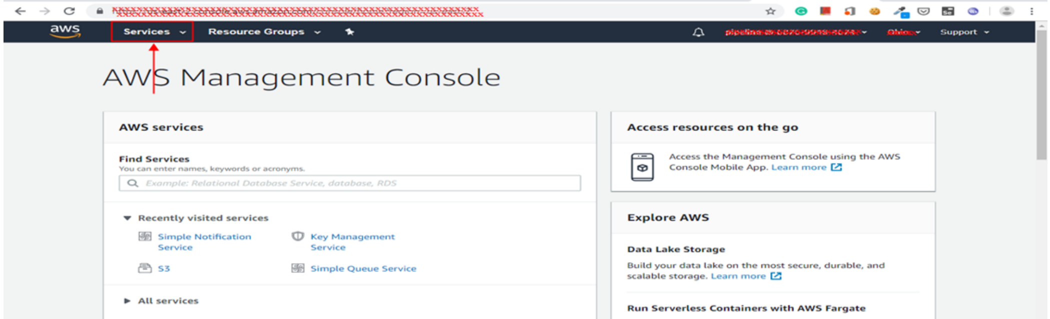
Receive twelve months of access to the AWS Free Usage Tier and enjoy AWS Basic Support features including, 24x7x365 customer service, support forums, and more.

c. Sign in with your credentials.

d. The AWS Management Console opens.

e. Click the ‘Services’ option from the header drop-down.

f. Search SQS service using the ‘Find Services’ space.



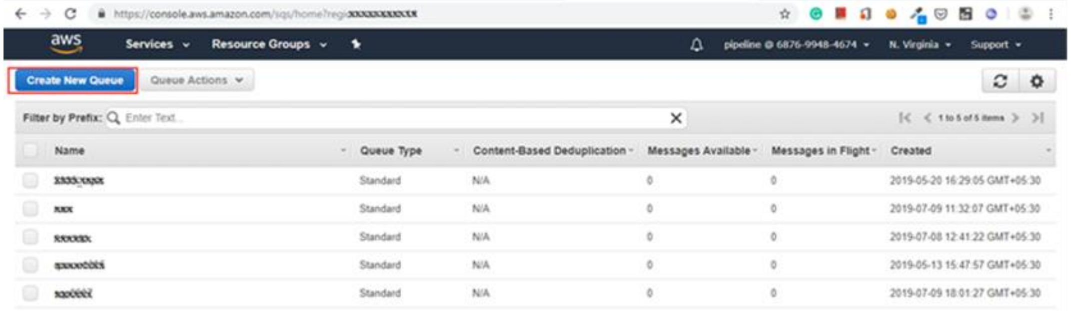
[redacted]
aws
Services
Resource Groups
[redacted]
[redacted]
Support
AWS Management Console
AWS services
Find Services
You can enter names, keywords or acronyms.
Example: Relational Database Service, database, RDS
Recently visited services
Simple Notification Service
Key Management Service
S3
Simple Queue Service
All services
Access resources on the go
Explore AWS
Data Lake Storage
Run Serverless Containers with AWS Fargate

Access the Management Console using the AWS Console Mobile App. Learn more

Build your data lake on the most secure, durable, and scalable storage. Learn more

g. The Amazon SNS page opens.

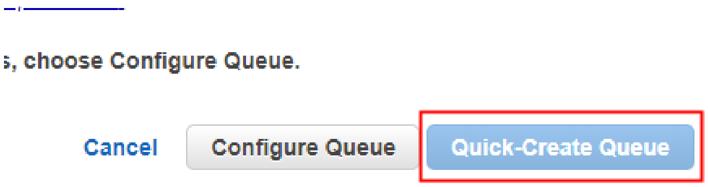
h. Click the ‘Create New Queue’ option.



https://console.aws.amazon.com/sqs/home?regi[redacted]

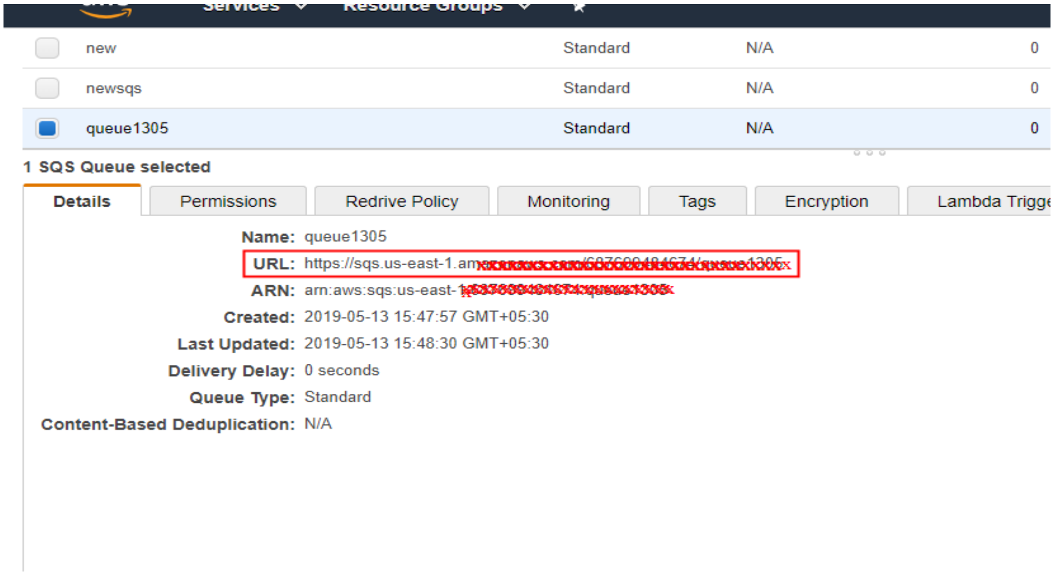
Name	Queue Type	Content-Based Deduplication	Messages Available	Messages in Flight	Created
[redacted]	Standard	N/A	0	0	2019-05-20 16:29:05 GMT+05:30
[redacted]	Standard	N/A	0	0	2019-07-09 11:32:07 GMT+05:30
[redacted]	Standard	N/A	0	0	2019-07-08 12:41:22 GMT+05:30
[redacted]	Standard	N/A	0	0	2019-05-13 15:47:57 GMT+05:30
[redacted]	Standard	N/A	0	0	2019-07-09 18:01:27 GMT+05:30

i. Click the 'Quick-Create Queue' option after giving SQS name.



j. Click on the created SQS to see the details.

k. Copy the SQS url from there.



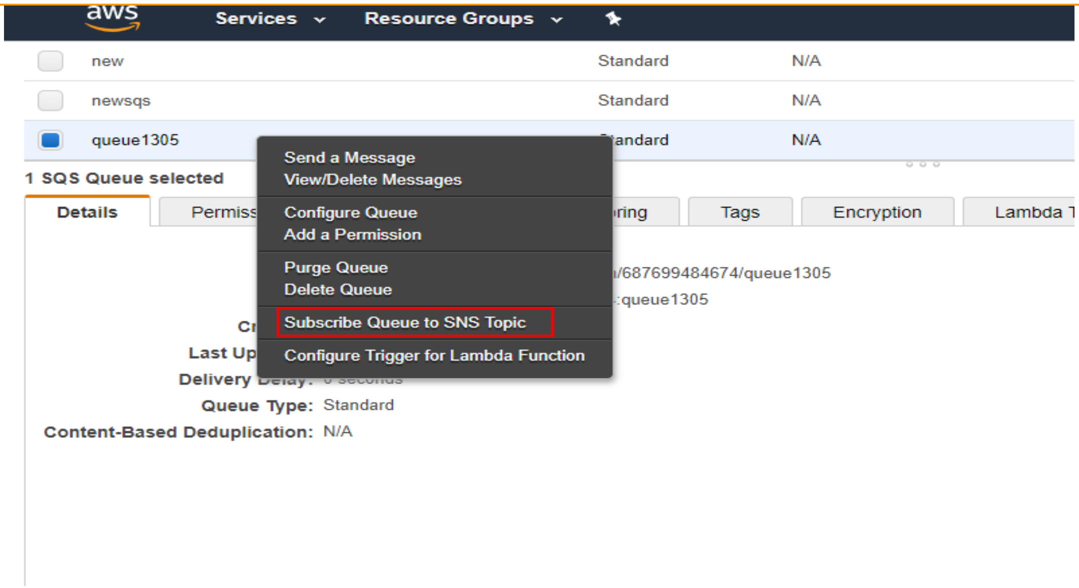
☐ new	Standard	N/A	0
☐ newsqs	Standard	N/A	0
☑ queue1305	Standard	N/A	0

1 SQS Queue selected

Name: queue1305
URL: https://sqs.us-east-1.am[redacted]
ARN: arn:aws:sqs:us-east-1[redacted]
Created: 2019-05-13 15:47:57 GMT+05:30
Last Updated: 2019-05-13 15:48:30 GMT+05:30
Delivery Delay: 0 seconds
Queue Type: Standard
Content-Based Deduplication: N/A

l. Use right click on the created SQS.

m. Select the 'Subscribe Queue to SNS Topic' to subscribe to the desired SNS Topic.



new	Standard	N/A
newsqs	Standard	N/A
queue1305	andard	N/A

1 SQS Queue selected

/687699484674/queue1305
:queue1305

Queue Type: Standard
Content-Based Deduplication: N/A

n. The ‘Subscribe to a Topic’ window opens.

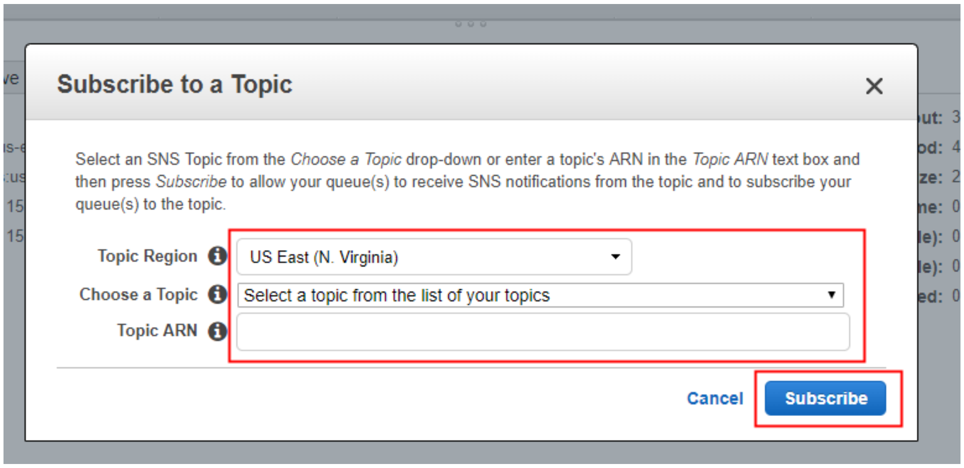
o. Provide the following information:

   i. Topic Region

   ii. Choose a Topic

   iii. Topic ARN

p. Click the ‘Subscribe’ option.



Select an SNS Topic from the *Choose a Topic* drop-down or enter a topic's ARN in the *Topic ARN* text box and then press *Subscribe* to allow your queue(s) to receive SNS notifications from the topic and to subscribe your queue(s) to the topic.

Topic Region: US East (N. Virginia)

Choose a Topic: Select a topic from the list of your topics

Topic ARN:

Note:

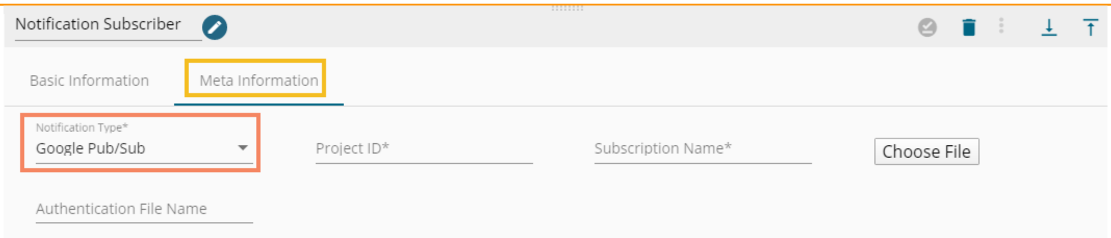
a. Choose the Topic from where your messages are being published.

b. SNS Region: The same can be seen in the SNS ARN URL. The user needs to use the same.

8.1.10.2. Google Pub/Sub as the selected Notification Type

i) Navigate to the Meta Information tab provided for the Notification Publisher.

ii) Select the ‘Google Pub/Sub’ as the Notification Type option using the drop-down menu.


Notification Subscriber

Basic Information | Meta Information

Notification Type*: Google Pub/Sub

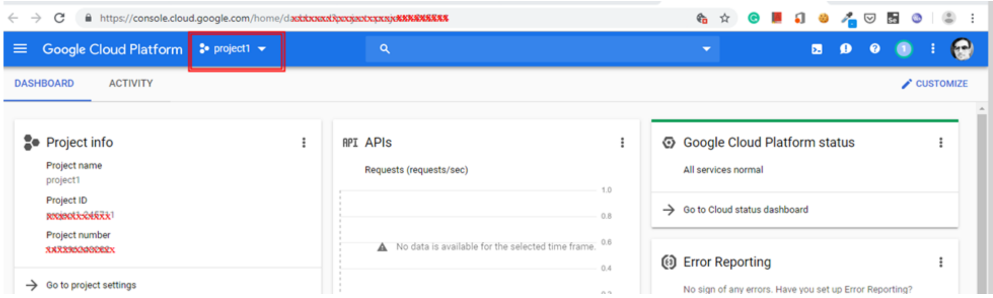
Project ID*

Subscription Name*

Choose File

Authentication File Name

iii) The user needs to follow some more steps to get the Pub/Subtopic ID
iv) Navigate to the https://console.cloud.google.com/
v) Log in with your credentials.
vi) The Landing page opens.
vii) Click on the marked area.
viii) Create a Project or use the desired one.


https://console.cloud.google.com/home/d[redacted]

Google Cloud Platform · project1

DASHBOARD | ACTIVITY | CUSTOMIZE

Project info

Project name: project1

Project ID: [redacted]

Project number: [redacted]

Go to project settings

APIs

Requests (requests/sec)

No data is available for the selected time frame.

Google Cloud Platform status

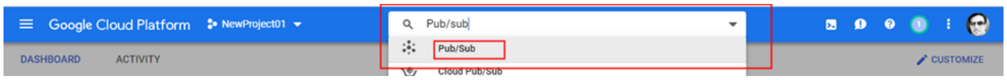
All services normal

Go to Cloud status dashboard

Error Reporting

No sign of any errors. Have you set up Error Reporting?

ix) Search for the Pub/Sub option in the Search bar.


Google Cloud Platform · NewProject01

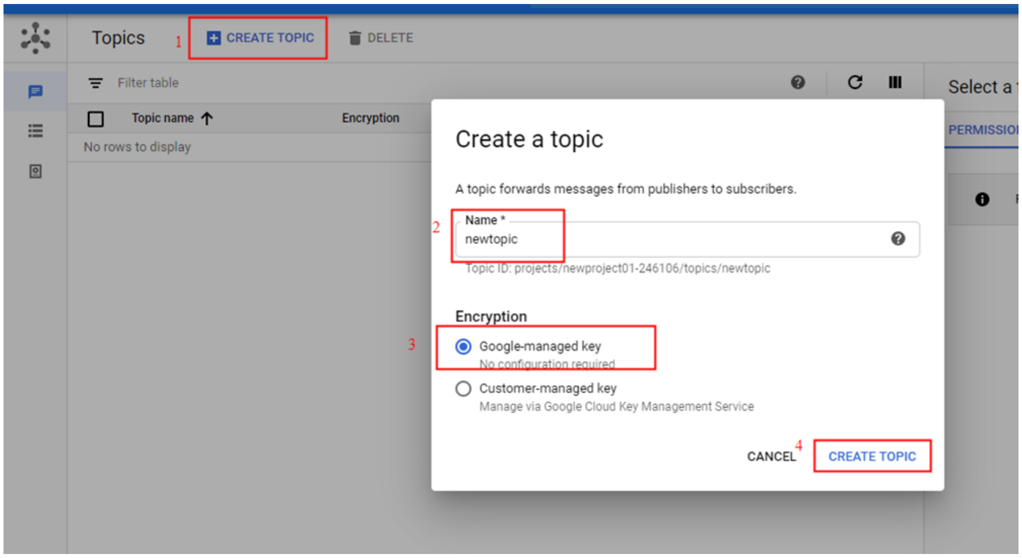
Pub/sub

Pub/Sub

Cloud Pub/Sub

DASHBOARD | ACTIVITY | CUSTOMIZE

x) Create a Topic using the following steps:
a. Click the ‘CREATE TOPIC’ option.
b. Provide a name for the new topic
c. Select the ‘Google-managed key’ as the Encryption option.
d. Click the ‘CREATE TOPIC’ option.


Topics | 1 CREATE TOPIC | DELETE

Filter table

Topic name | Encryption

No rows to display

Select a

PERMISSIO

Create a topic

A topic forwards messages from publishers to subscribers.

2 Name *: newtopic

Topic ID: projects/newproject01-246106/topics/newtopic

Encryption

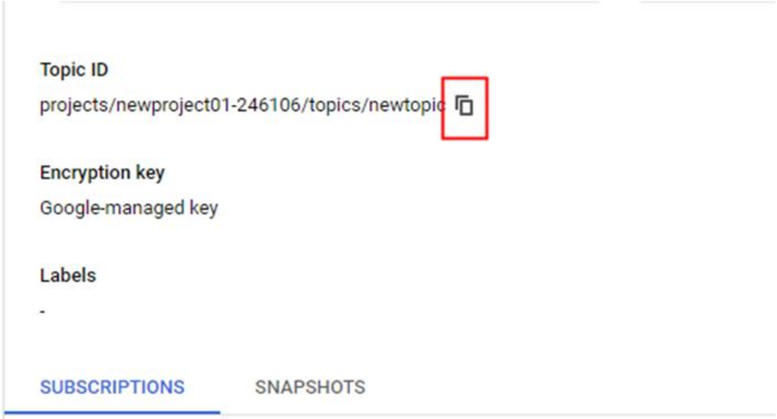
3 ● Google-managed key — No configuration required

○ Customer-managed key — Manage via Google Cloud Key Management Service

CANCEL | 4 CREATE TOPIC

xi) The user gets either of the following screens:

a. While using an existing Topic ID



Topic ID

projects/newproject01-246106/topics/newtopic

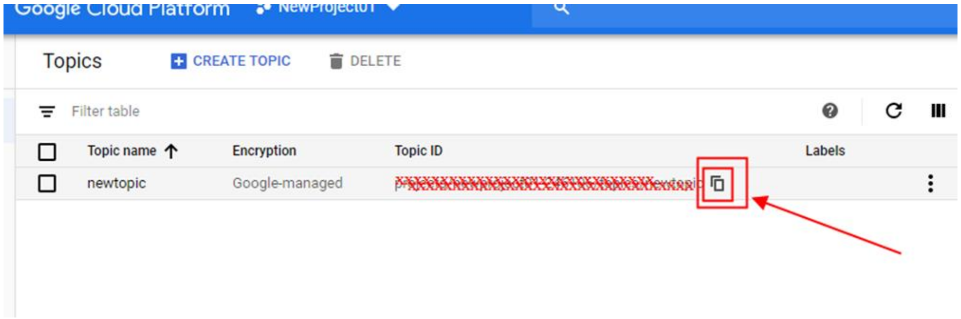
Encryption key

Google-managed key

Labels

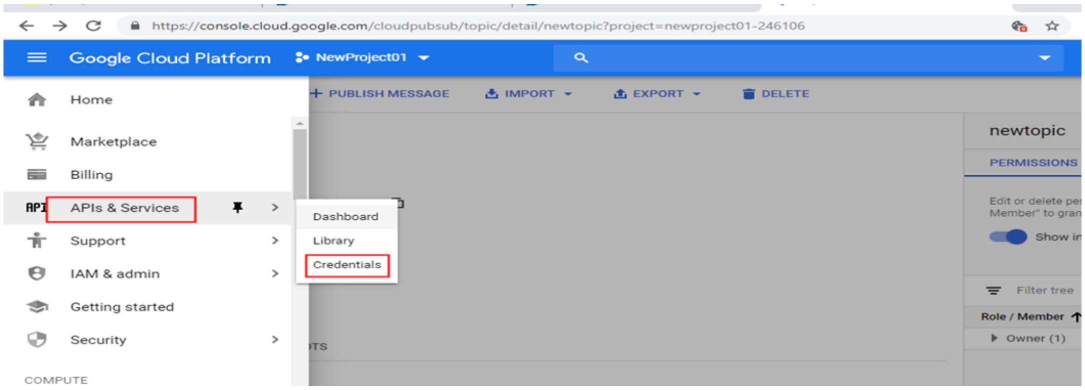
-

Alternatively, while using the newly created Topic ID



☐	Topic name ↑	Encryption	Topic ID	Labels
☐	newtopic	Google-managed	[redacted]	

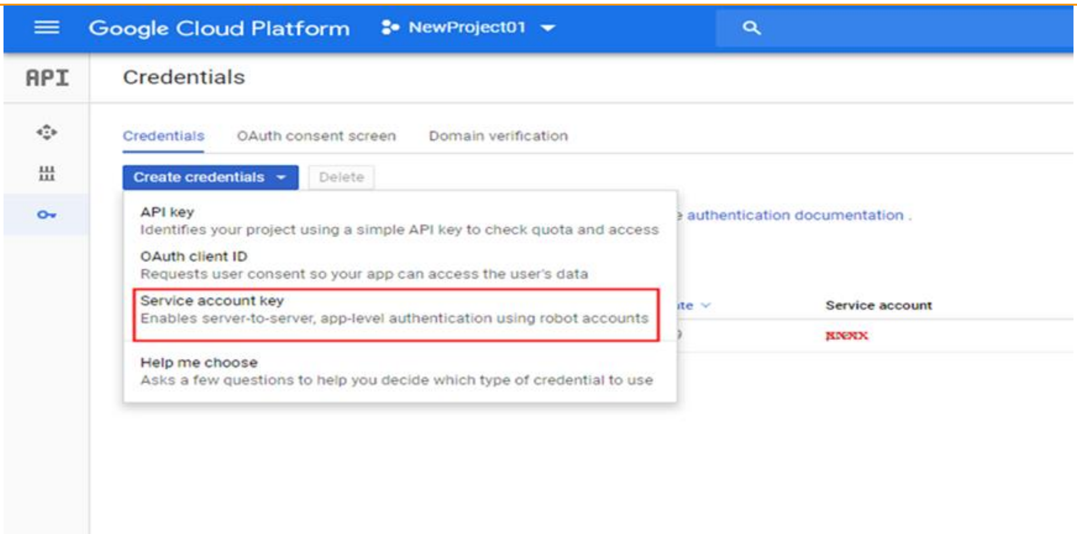
b. Create Auth-JSON for the current Project.



https://console.cloud.google.com/cloudpubsub/topic/detail/newtopic?project=newproject01-246106

c. Create a Service account key by following the given steps:

i. Click the ‘**Service account key**’ option from the Credentials tab.



API key
Identifies your project using a simple API key to check quota and access

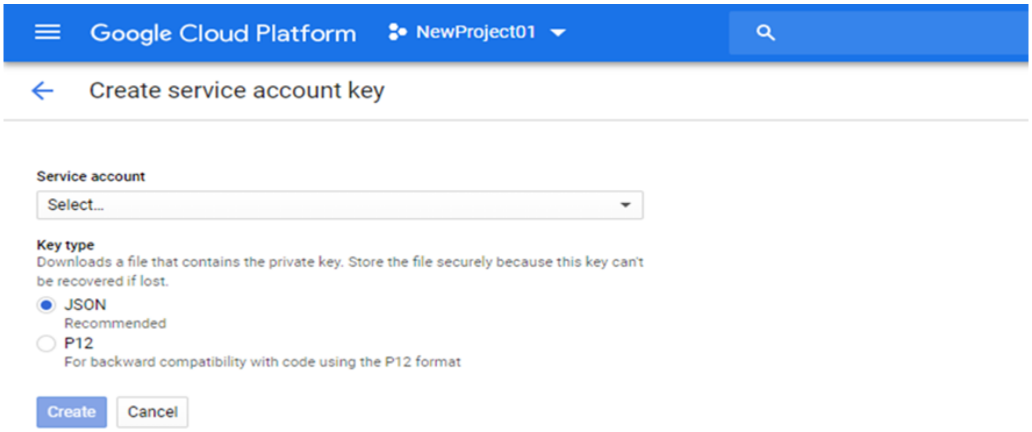
OAuth client ID
Requests user consent so your app can access the user's data

Service account key
Enables server-to-server, app-level authentication using robot accounts

Help me choose
Asks a few questions to help you decide which type of credential to use

ii. Select a Service account using the drop-down menu.
iii. Download the JSON file using the radio button.
iv. Click the ‘Create’ option.

Use the copied Topic ID or the service account as metadata input.



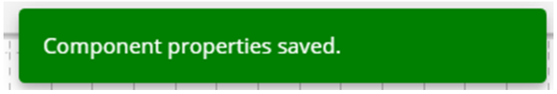
Service account
Select...

Key type
Downloads a file that contains the private key. Store the file securely because this key can't be recovered if lost.

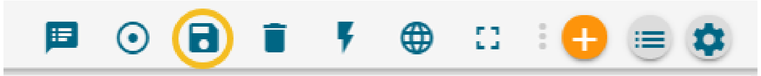
● JSON
Recommended
○ P12
For backward compatibility with code using the P12 format

.

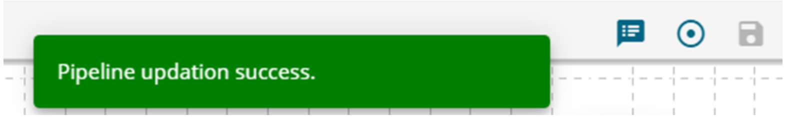
xii) Click the ‘**Save Component in Storage**’ icon.
xiii) A notification message appears to inform about the component configuration success.


Component properties saved.

xiv) Click the ‘Update Pipeline’ icon.

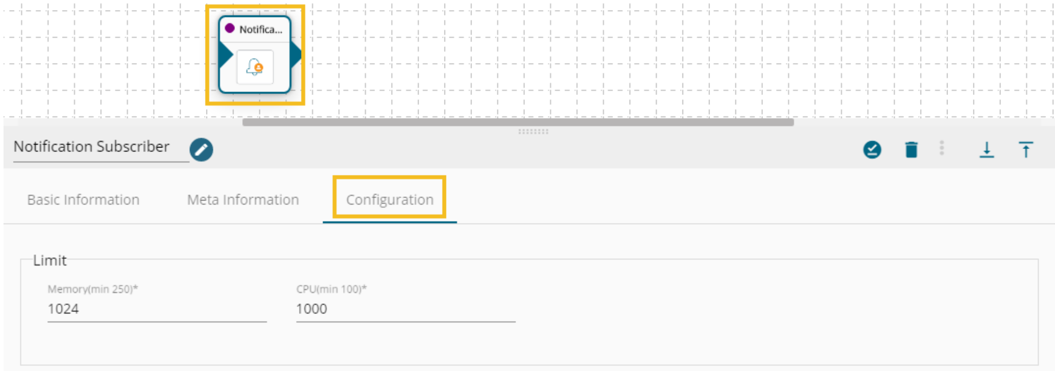


xv) A notification message appears to inform that the pipeline is successfully updated.


Pipeline updation success.

xvi) Open the SFTP Reader component by a click on it.

xvii) The Notification Subscriber gets the 'Configuration' properties tab.



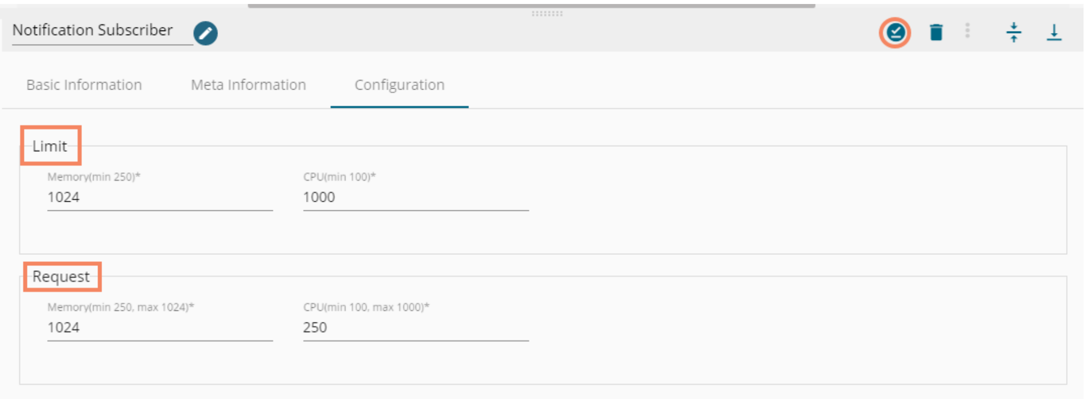
Limit

Memory(min 250)*: 1024

CPU(min 100)*: 1000

xviii) Modify the values for the Limit or Request section from the Configuration tab.

xix) Click the 'Save Component in Storage' ✓ icon.



Limit

Memory(min 250)*: 1024

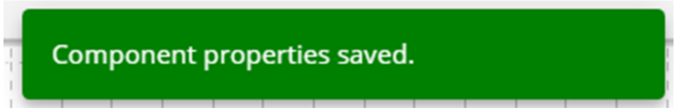
CPU(min 100)*: 1000

Request

Memory(min 250, max 1024)*: 1024

CPU(min 100, max 1000)*: 250

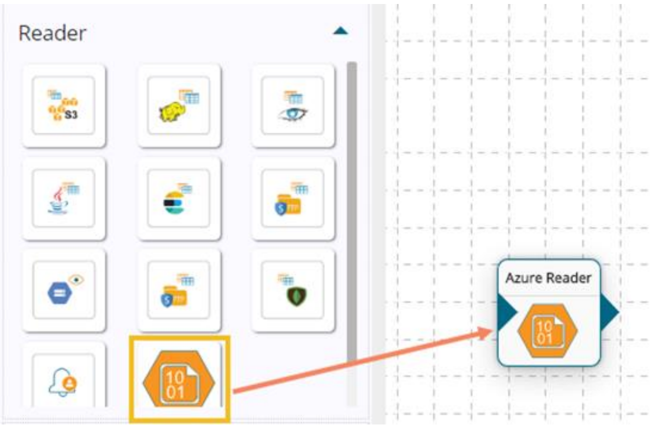
xx) A message appears to notify about the component properties.


Component properties saved.

xxi) The Notification Subscriber component configuration gets completed and it is ready to be used in the Pipeline Workflow.

8.1.11. Azure Reader

i) Drag and drop the Azure Reader component to the Workflow Editor.



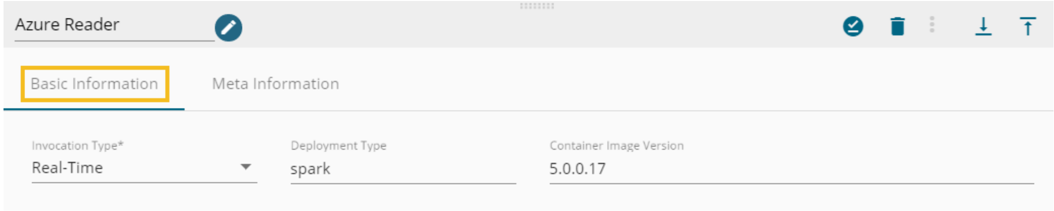
ii) Click on the dragged Azure Reader to get the component properties tabs.

iii) **Basic Information**: It is the default tab to open for the component while configuring the component.

a. Invocation Type: Select an invocation mode out of '**Real-Time**' or '**Batch**' using the drop-down menu.

b. Deployment Type: It displays the deployment type for the reader component. This field comes pre-selected.

c. Container Image Version: It displays the image version for the docker container. This field comes pre-selected.


Azure Reader

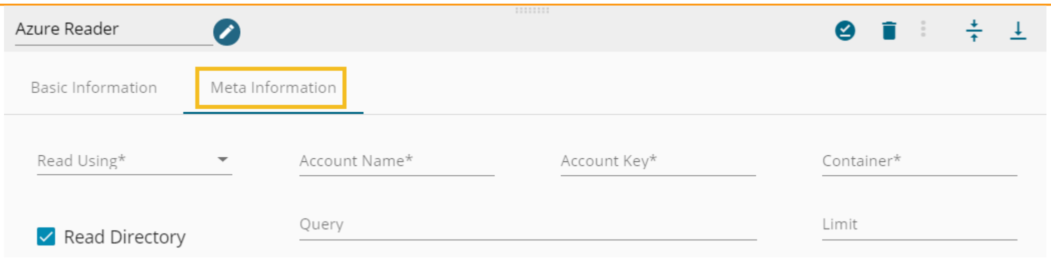
Invocation Type* Real-Time

Deployment Type spark

Container Image Version 5.0.0.17

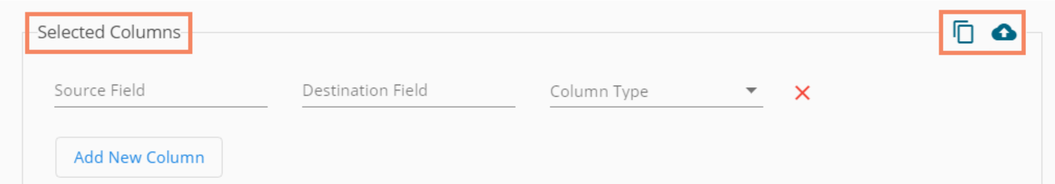
iv) Open the '**Meta Information**' tab and fill all the connection-specific details for the Azure Reader.

1. Read Using(*): Select an option from the drop-down menu to connect to data. The supported options are '**Connection String**' and '**Secret Key**'.
2. Account Name(*): Name of the Azure account.
3. Account Key(*): Provide Account Key to login.
4. Container(*): Container details.
5. Read Directory: Enable this option by putting a checkmark in the box to be read from the Directory.
6. Blob Name: Provide a blob name. The Blob field appears if the '**Read Directory**' option is not enabled.
7. Query: Insert relevant query.
8. Limit: Set limit for the number of records.



v) Selected Columns: There is also a section for the selected columns in the Meta Information tab if the user can select some specific columns from the table to read data instead of selecting a complete table so this can be achieved by using the ‘Selected Columns’ section. Select the columns which you want to read and if you want to change the name of the column, then put that name in alias name section otherwise keep alias name same as of column name and then select a Column Type from the drop-down menu.
Or
Use ‘Download Data’ and ‘Upload File’ options to select the desired columns.

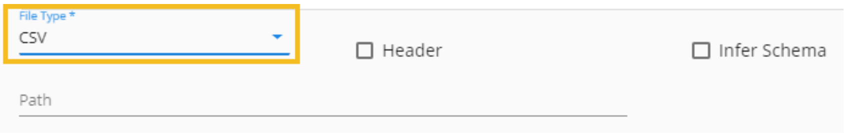
1. Upload File: The user can upload the existing system files (CSV, JSON) using the ‘Upload File’ icon (file size must be less than 2 MB).
2. Download Data (Schema): Users can download the schema structure in JSON format by using the ‘**Download Data**’ icon.



vi) File Type(*): Select a file type from the drop-down menu (CSV, JSON, PARQUET, AVRO are the supported file types).
The users get the File Type and Limit fields at the end. The File Type is a mandatory field for the Azure reader.

Currently, it supports CSV, JSON, PARQUET, AVRO file types. The configuration fields may vary based on the selection of the file type.

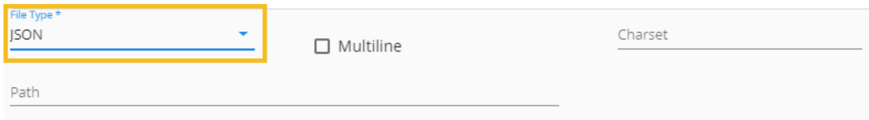
1. CSV
   Configure the following fields when the selected file type is CSV:
   a. Header- Enable or Disable the Header option to confirm whether the stored data has header or not
   b. Infer Schema- Enable or disable the Infer Schema option to confirm whether the schema is provided or not
   c. Path- Provide a specific path for the file


File Type *
CSV

2. JSON

   Configure the following fields when the selected file type is JSON:

   a. Multiline-Enable this option, if the stored JSON has line breaks in-between
   b. Charset-Character set encoding, E.g., UTF-8
   c. Path-provide a specific path for the file.



File Type *: JSON

□ Multiline

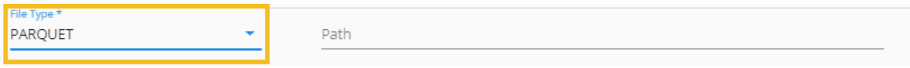
Charset

Path

3. PARQUET

   Configure the following fields when the selected file type is PARQUET:

   a. Path-provide a specific path for the file.



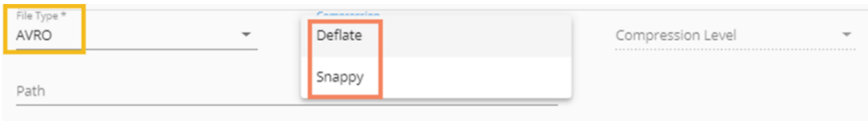
File Type *: PARQUET

Path

4. AVRO

   Configure the following fields when the selected file type is AVRO:

   a. Compression: Select a compression option from the drop-down menu out of 'Deflate' and 'Snappy' options.
   b. Compression Level- If the selected Compression option is 'Deflate' then select the compression level from 0-9 (where 0 and 9 are included).
   c. Path-provide the specific path for the file.
   d. Limit- set limit for the number of rows.



File Type *: AVRO

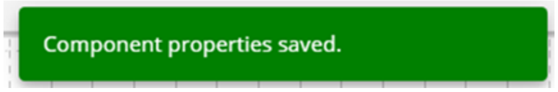
Compression: Deflate, Snappy

Compression Level

Path

vii) Click the '**Save Component in Storage**' icon after doing all the configurations to save the reader component.

viii) A notification message appears to inform you that the component properties have been saved.



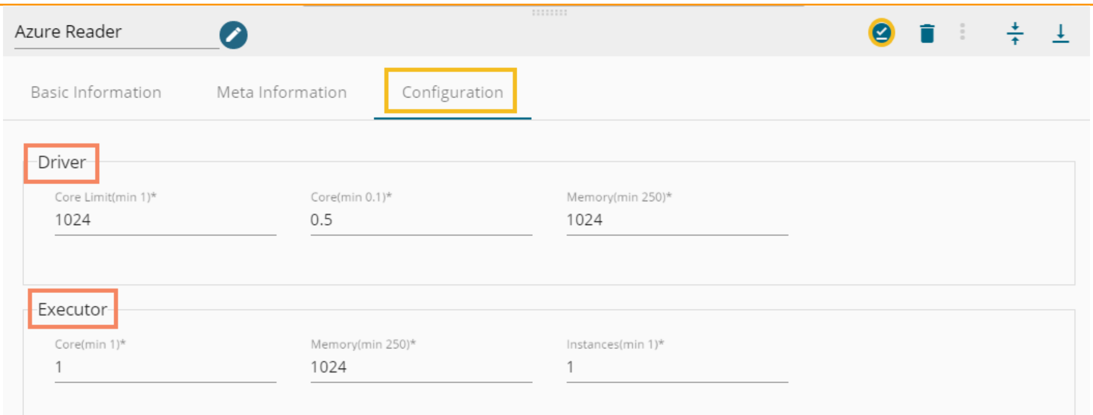
Component properties saved.

ix) Click the 'Update Pipeline' icon. Another notification message appears to inform that the pipeline is updated.

x) Open the Azure Reader component again to see the '**Configuration**' properties tab below.

xi) Modify the values for the Limit or Request section from the Configuration tab.

xii) Click the '**Save Component in Storage**' icon.


Azure Reader

Basic Information | Meta Information | Configuration

Driver

Core Limit(min 1)*: 1024

Core(min 0.1)*: 0.5

Memory(min 250)*: 1024

Executor

Core(min 1)*: 1

Memory(min 250)*: 1024

Instances(min 1)*: 1

xiii) A message appears to notify about the component properties.

xiv) A notification message appears, and the Azure Reader component gets ready to be used in a pipeline workflow.

Note:

a. (*) symbol indicates that the field is mandatory.

b. Either table or query must be specified for the data readers except for SFTP Reader.

c. Selected Columns- There should not be a data type mismatch in the Column Type for all the Reader components.

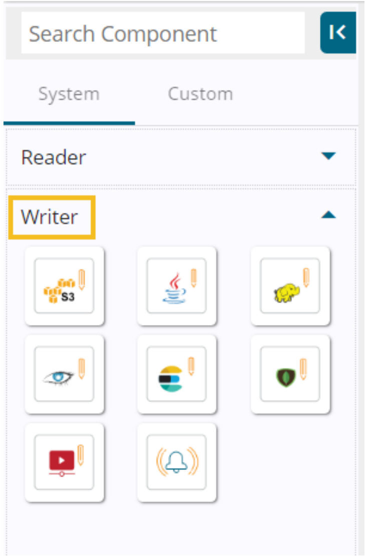
8.2. Writer

The Data Pipeline provides Writer components to write input data to the required place and create a pipeline workflow.

i) Navigate to the Pipeline Workflow Editor.

ii) Expand the Writer section using the Components Pallet.

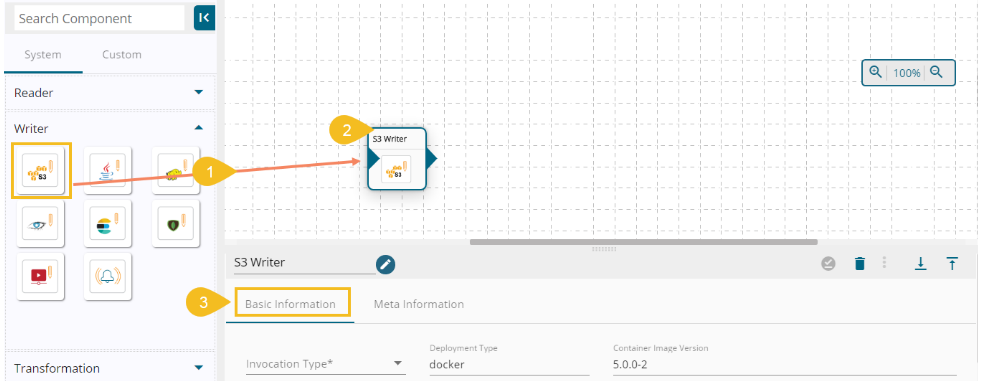
iii) The following Writer components are available under the System tab of the Component Pallet.



8.2.1. S3 Writer

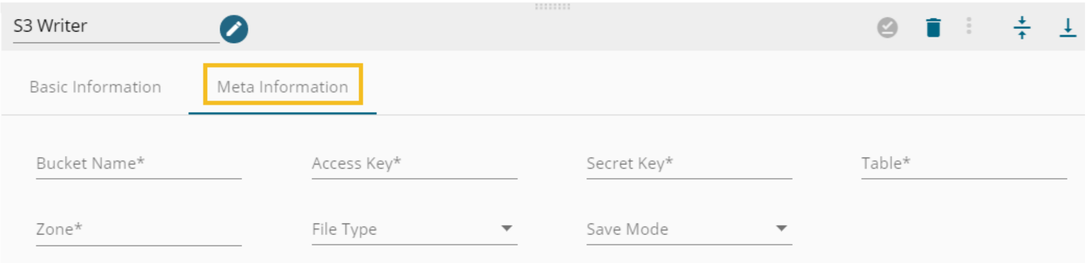
The S3 writer component writes data to the AWS S3 location.

i) Drag & Drop the '**S3 Writer**' component to the Workflow Editor.
ii) Click the dragged Writer component to get the Properties tabs below.
iii) Basic Information: It is the default tab to open for the S3 writer while configuring the component.
   a. Select an Invocation type from the drop-down menu to confirm the running mode of the reader component. Select '**Real-Time**' or '**Batch**' from the drop-down menu.
   b. Deployment Type: It displays the deployment type for the component. This field comes pre-selected.
   c. Container Image Version: It displays the image version for the docker container. This field comes pre-selected.

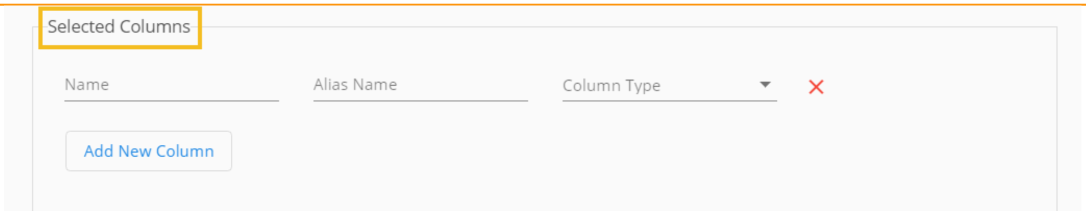


Deployment Type: docker
Container Image Version: 5.0.0-2

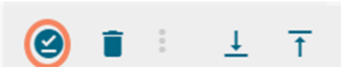
iv) Open the '**Meta Information**' tab and fill all the connection-specific details for the S3 Writer.
   a. Bucket Name(*): Provide Bucket name
   b. Access Key(*): Access key shared by AWS to login
   c. Secret Key(*): Secret key shared by AWS to login
   d. Table(*): Mention the Table or object name which is to be read
   e. Zone(*): Provide the zone information
   f. File Type: Select a file type from the drop-down menu (CSV, JSON, PARQUET, AVRO are the supported file types)
   g. Save Mode: supported save mode is '**Append**' at present.



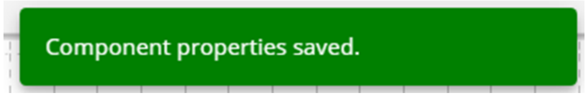
v) Selected Columns- Select the columns that you wish to store in the S3 location. Provide Column Name, Alias Name, Column Type for the selected column(s).
(Add multiple columns to the Selected Columns section using the '**Add New Column**' option.)



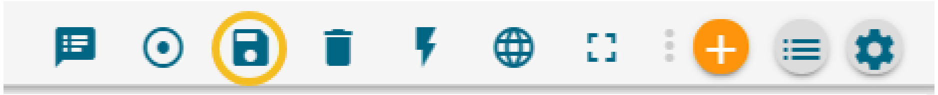
vi) Save the component properties.



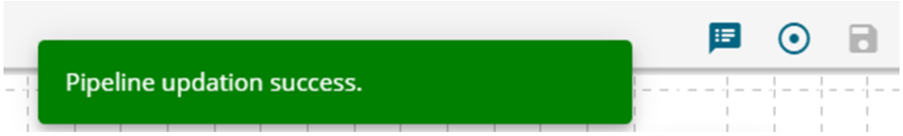
vii) A message appears to assure that the component properties have been saved.


Component properties saved.

viii) Click the ‘Update Pipeline’ icon.

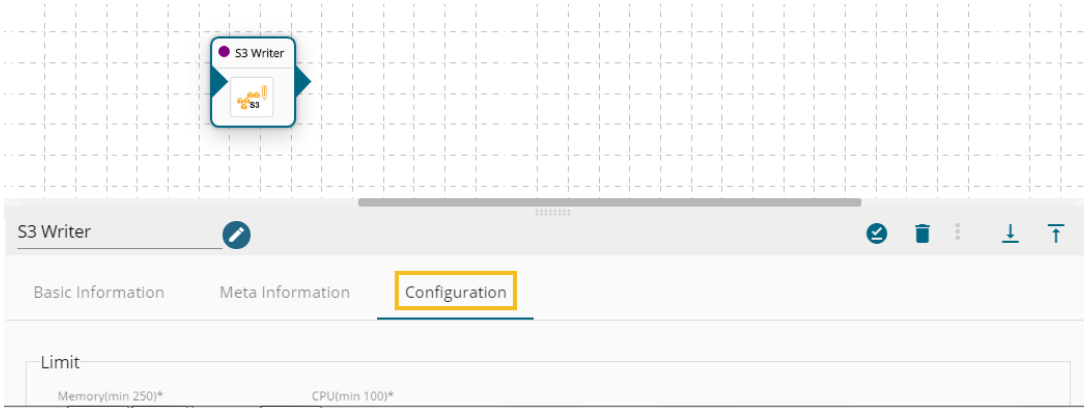


ix) A message appears to assure that the Pipeline has been updated.


Pipeline updation success.

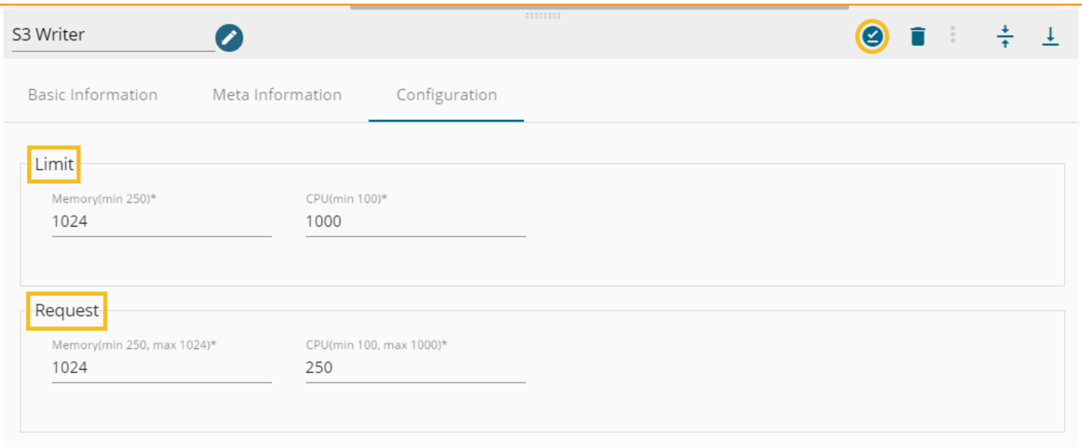
x) Open the S3 writer component.

xi) The Configuration tab appears for the dragged S3 writer component.



xii) Modify the Limit or Request values.

xiii) Click the ‘**Save Component in Storage**’ icon.


S3 Writer

Basic Information | Meta Information | Configuration

Limit

Memory(min 250)*: 1024

CPU(min 100)*: 1000

Request

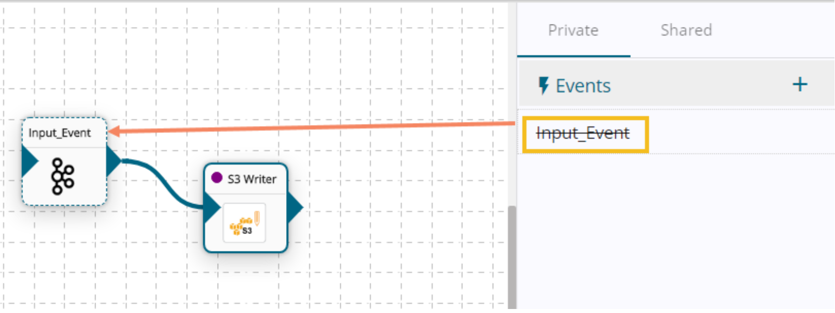
Memory(min 250, max 1024)*: 1024

CPU(min 100, max 1000)*: 250

The S3 writer configuration gets completed and now ready to be used in the pipeline workflow.

Note:

a. The S3 Writer requires data input from an Event and writes the data to the S3 location.
   i) Create one event and drag it to the workspace.
   ii) Connect the created input Event to the dragged S3 writer component.



b. The data in the input event can come from any Ingestion, Readers or shared events.

8.2.2. RDBMS Writer

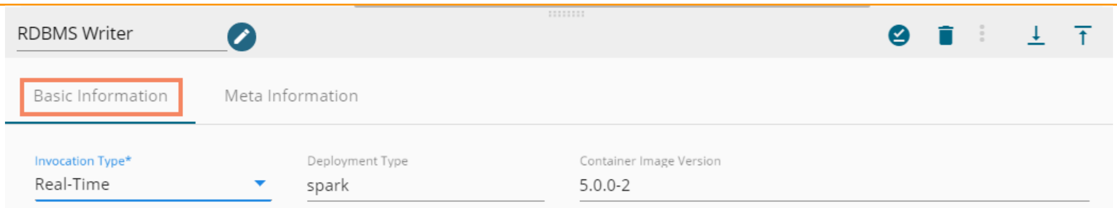
The RDBMS writer helps to write data to the supported relational databases.

i) Drag & drop the RDBMS Writer component to the Workflow Editor.

ii) Click the RDBMS writer to open component properties tabs below.

iii) Basic Information: It is the default tab to open for the RDBMS writer while configuring the component.

a. Select an Invocation type from the drop-down menu to confirm the running mode of the reader component. Select ‘Real-Time’ or ‘Batch’ from the drop-down menu.
b. Deployment Type: It displays the deployment type for the component. This field comes pre-selected.
c. Container Image Version: It displays the image version for the docker container. This field comes pre-selected.



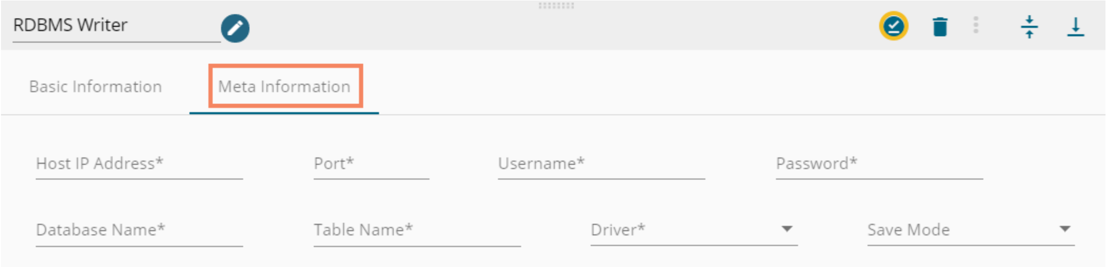
Invocation Type* Real-Time

Deployment Type spark

Container Image Version 5.0.0-2

iv) Open the '**Meta Information**' tab and fill all the connection-specific details for the RDBMS Writer.

a. Host
b. Port
c. Username
d. Password
e. Driver (MySQL/MSSQL/Oracle/Postgres)
f. Database Name
g. Table Name
h. Save Mode (Append/Overwrite/Upsert)

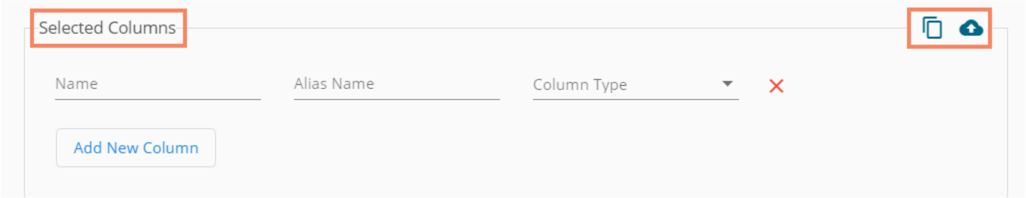


v) Selected Columns: The users can select some specific columns from the table to read data instead of selecting a complete table; this can be achieved via the '**Selected Columns**' section. Select the columns which you want to read and if you want to change the name of the column, then put that name in alias name section otherwise keep alias name same as of column name and then select a Column Type from the drop-down menu.

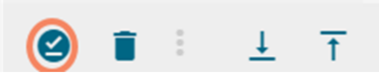
or

Use the '**Download Data**' and '**Upload File**' options to select the desired columns.

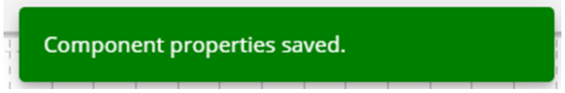
1. Upload File: The user can upload the existing system files (CSV, JSON) using the '**Upload File**' ☁ icon (file size must be less than 2 MB).
2. Download Data (Schema): Users can download the schema structure in JSON format by using the '**Download Data**' ⧉ icon.



vi) Save the configured properties for the RDBMS writer.

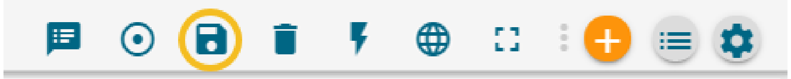


vii) A message appears to notify that the component properties got saved.

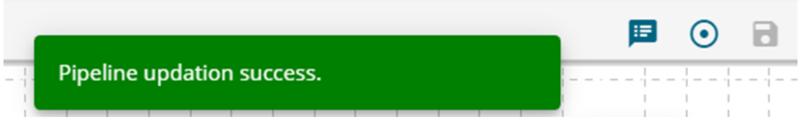


Component properties saved.

viii) Click the ‘**Update Pipeline**’ icon.



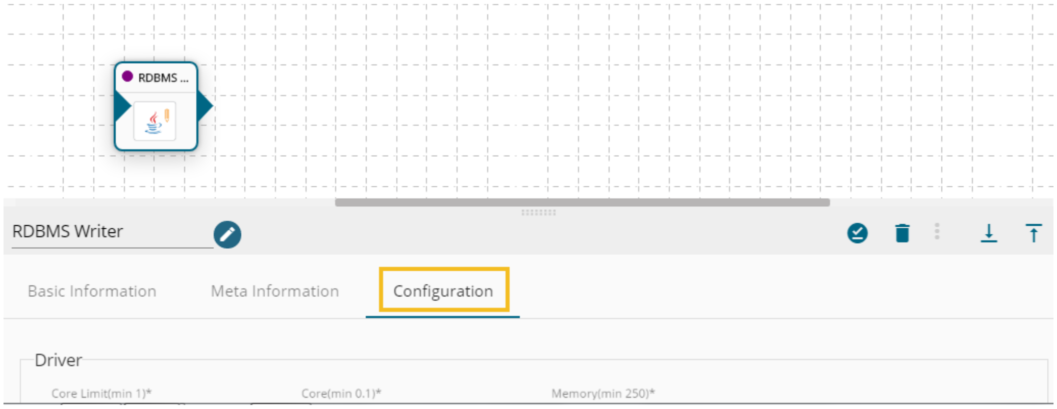
ix) A notification message appears to confirm the pipeline update.



Pipeline updation success.

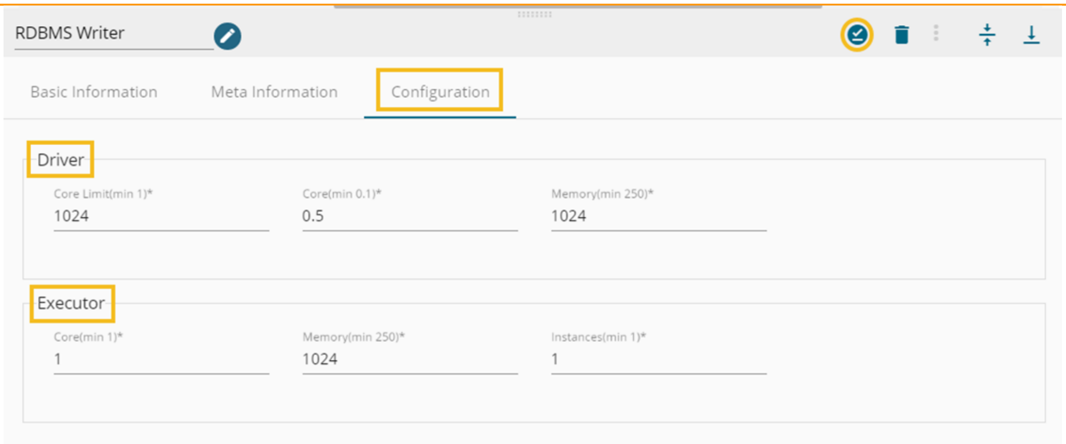
x) Click the dragged RDBMS writer component.

xi) The ‘**Configuration**’ appears for the dragged RDBMS writer.



xii) Modify the Driver and Executor values.

xiii) Click the ‘**Save Component in Storage**’ icon.



Driver

Core Limit(min 1)*: 1024

Core(min 0.1)*: 0.5

Memory(min 250)*: 1024

Executor

Core(min 1)*: 1

Memory(min 250)*: 1024

Instances(min 1)*: 1

xiv) A message appears to notify that the RDBMS component properties are saved.

Component properties saved.

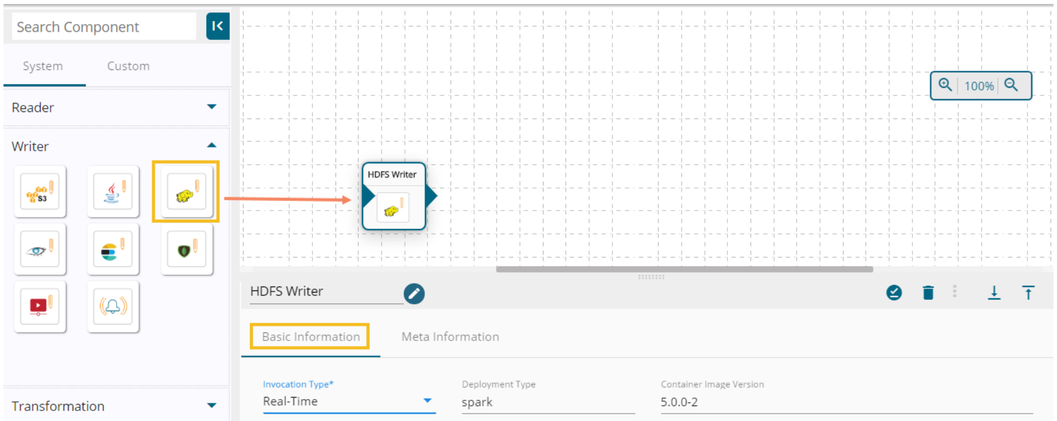
xv) The RDBMS Writer is now ready to be used in the Data Pipeline workflow.

Note: The data input event can take data from Ingestion, Readers, or shared Event.

8.2.3. HDFS Writer

The HDFS writer writes the data to the HDFS location.

i) Drag the HDFS Writer component to the Workflow Editor.
ii) Click the HDFS Writer to open the component properties tabs below.
iii) Basic Information: It is the default tab to open for the HDFS Writer while configuring the component.
   1) Select an Invocation type from the drop-down menu to confirm the running mode of the reader component. Select either 'Real-Time' or 'Batch' from the drop-down menu.
   2) Deployment Type: It displays the deployment type for the component. This field comes pre-selected.
   3) Container Image Version: It displays the image version for the docker container. This field comes pre-selected.



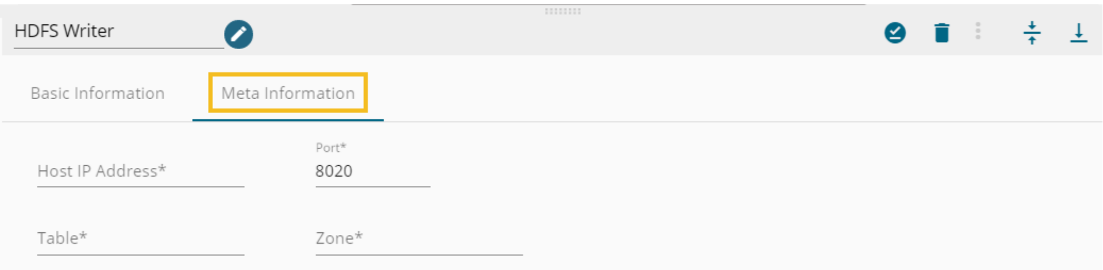
Invocation Type*: Real-Time

Deployment Type: spark

Container Image Version: 5.0.0-2

iv) Open the '**Meta Information**' tab and fill all the connection-specific details for the HDFS Writer.

1. Host IP Address(*): Hadoop IP address of the host
2. Port(*): Port number
3. Table(*): Table or object name which must be read
4. Zone(*): HDFS Zone location


HDFS Writer

Basic Information | Meta Information

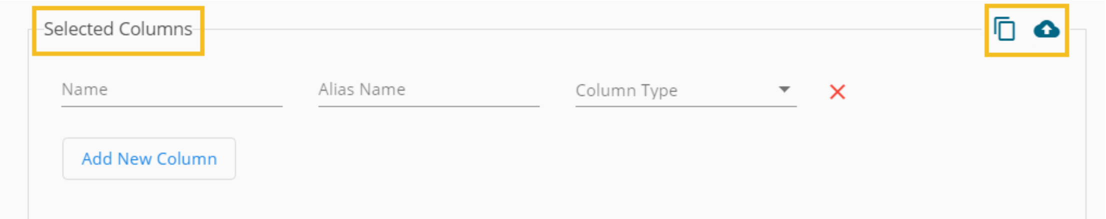
Host IP Address*

Port* 8020

Table*

Zone*

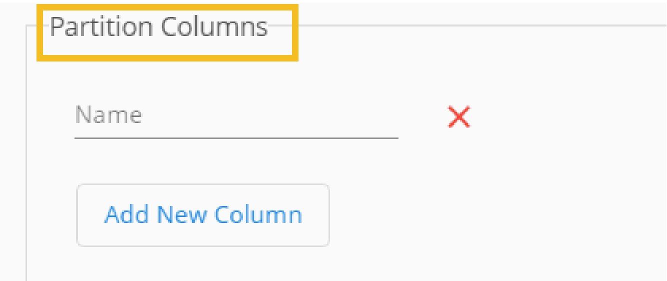
1. Selected Columns: The users can select some specific columns from the table to read data instead of selecting a complete table; this can be achieved via the 'Selected Columns' section. Select the columns which you want to read and if you want to change the name of the column, then put that name in alias name section otherwise keep alias name same as of column name and then select a Column Type from the drop-down menu.
   or
   Use 'Download Data' and 'Upload File' options to select the desired columns.
   1. Upload File: The user can upload the existing system files (CSV, JSON) using the 'Upload File' ☁ icon (file size must be less than 2 MB).
   2. Download Data (Schema): Users can download the schema structure in JSON format by using the 'Download Data' ⧉ icon.


Selected Columns

Name | Alias Name | Column Type

Add New Column

v) Partition Column- Provide a unique Key column name to partition data in Spark.

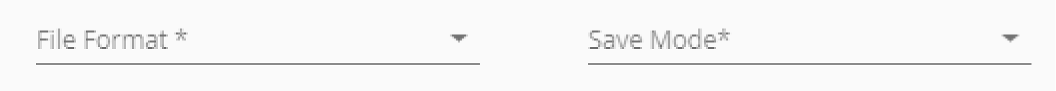

Partition Columns

Name

Add New Column

vi) File Format

The users get the File format and Save Mode fields at the end. The File Format is a mandatory field for the HDFS Writer.



Currently, we support CSV, JSON, PARQUET, AVRO file formats. The configuration fields may vary based on the selection of the file type. The supported Save Mode is Append.

vii) After doing all the configurations click the '**Save Component in Storage**' ✓ icon provided in the Writer configuration panel to save the component.

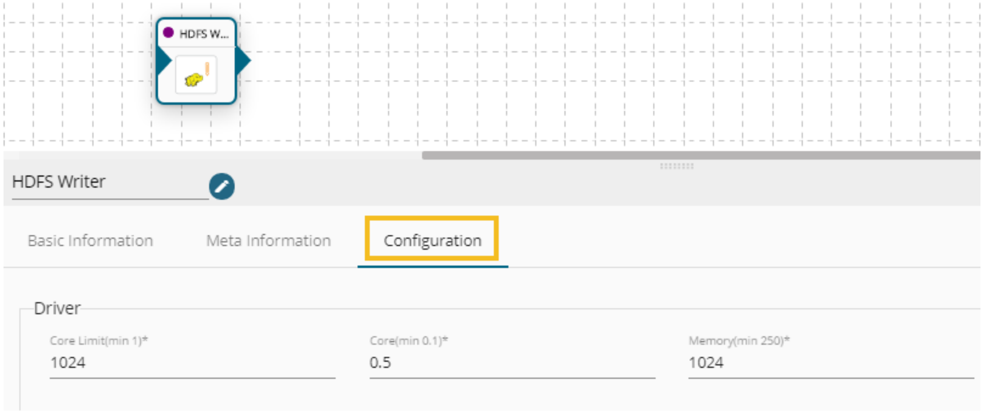
viii) A notification message appears to inform about the component configuration success.

ix) Click the '**Update Pipeline**' 💾 icon.

x) A notification message appears to inform that the pipeline is successfully updated.

xi) Open the HDFS Writer component by a click on it.

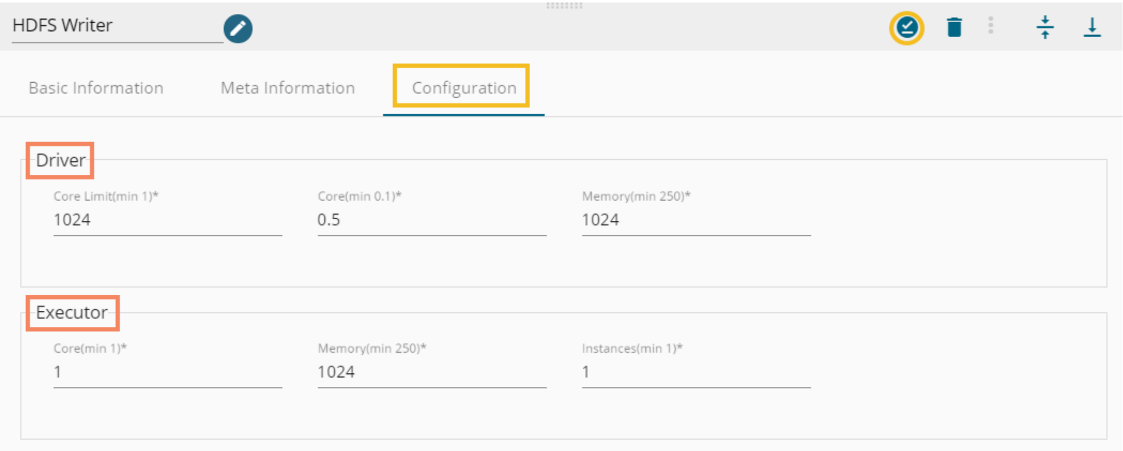
xii) The HDFS Writer gets the '**Configuration**' properties tab.



Core Limit(min 1)*: 1024
Core(min 0.1)*: 0.5
Memory(min 250)*: 1024

xiii) Modify the values for the Limit or Request section from the Configuration tab.

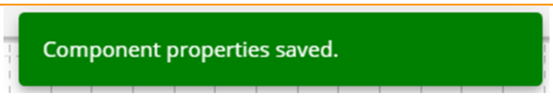
xiv) Click the '**Save Component in Storage**' ✓ icon.



Driver
Core Limit(min 1)*: 1024
Core(min 0.1)*: 0.5
Memory(min 250)*: 1024

Executor
Core(min 1)*: 1
Memory(min 250)*: 1024
Instances(min 1)*: 1

xv) A message appears to notify about the component properties.



Component properties saved.

xvi) The HDFS Writer component configuration gets completed and it is ready to be used in the Pipeline Workflow. It is mandatory to use a writer component with an Event component.

Note: The Input Event can take data from ingestion, Readers, or Shared Event.

8.2.4. Cassandra Writer

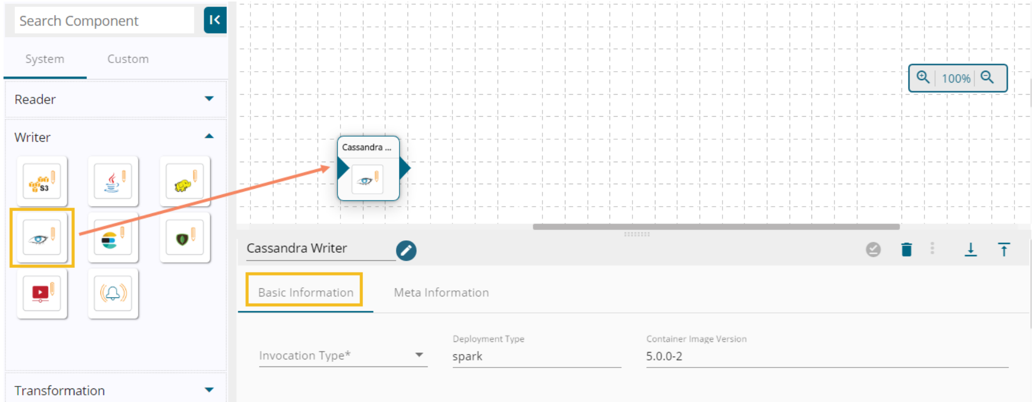
The Cassandra Writer writes data to the Cassandra Database.

i) Drag & drop the Cassandra Writer component to the Workflow Editor.

ii) Click the dragged Cassandra Writer component to open the component properties tabs below.

iii) Basic Information: It is the default tab to open for the Cassandra Writer while configuring the component.

1) Select an Invocation type from the drop-down menu to confirm the running mode of the reader component. Select 'Real-Time' or 'Batch' from the drop-down menu.
2) Deployment Type: It displays the deployment type for the component. This field comes pre-selected.
3) Container Image Version: It displays the image version for the docker container. This field comes pre-selected.



Cassandra Writer

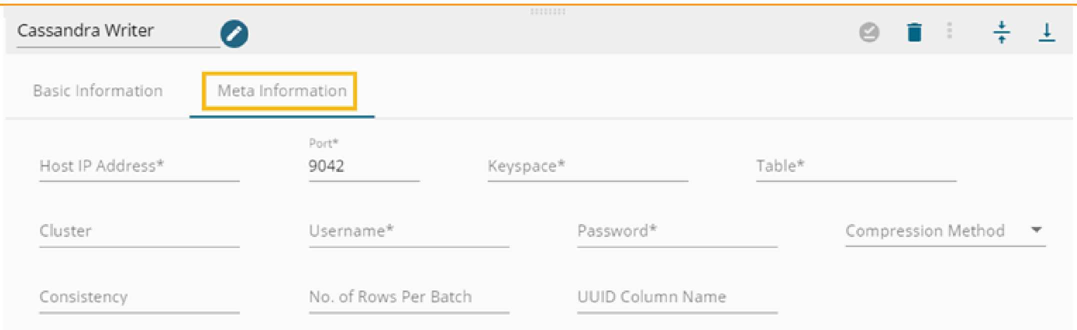
Invocation Type*

Deployment Type: spark

Container Image Version: 5.0.0-2

iv) Open the '**Meta Information**' tab and provide the connection-specific details.

a. Host IP address(*): Provide hostname based to connect the data connector.
b. Port number(*): The server port number gets displayed by default.
c. Keyspace(*): Provide a keyspace.
d. Table(*): Provide a table name where you wish to save the data.
e. Cluster: Provide a number to signify the number of clusters.
f. Username(*): Provide Username of the selected connection.
g. Password(*): Provide a password of the database.
h. Compression Method: Select a method from the drop-down menu (the supported method is SNAPPY)
i. Consistency: Provide a number to define consistency.
j. No. of Rows per Batch: Provide the number of rows to be fetched batch-wise.
k. UUID Column Name- Enter the UUID column name (Optional)


Cassandra Writer

Host IP Address*

Port* 9042

Keyspace*

Table*

Cluster

Username*

Password*

Compression Method

Consistency

No. of Rows Per Batch

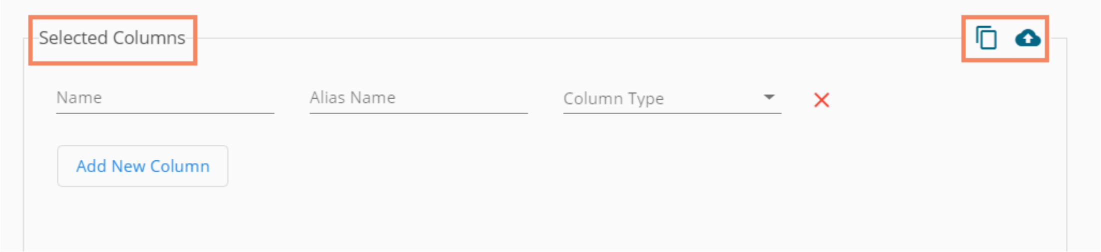
UUID Column Name

v) Selected Columns: The users can select some specific columns from the table to read data instead of selecting a complete table; this can be achieved via the 'Selected Columns' section. Select the columns which you want to read and if you want to change the name of the column, then put that name in alias name section otherwise keep alias name same as of column name and then select a Column Type from the drop-down menu.

or

Use 'Download Data' and 'Upload File' options to select the desired columns.

1. Upload File: The user can upload the existing system files (CSV, JSON) using the 'Upload File' ☁ icon (file size must be less than 2 MB).
2. Download Data (Schema): Users can download the schema structure in JSON format by using the 'Download Data' ⧉ icon.

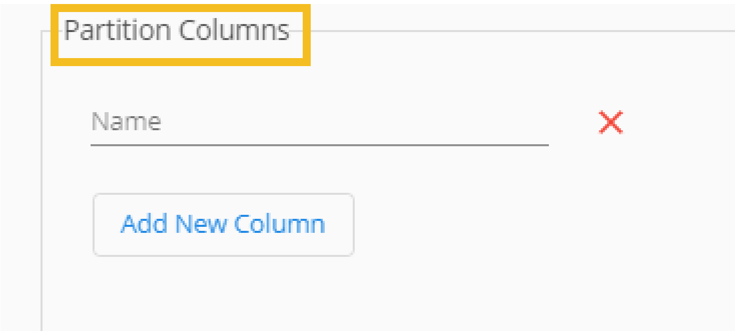

Selected Columns

Name

Alias Name

Column Type

vi) **Partition Columns**: Provide the name of the partition column. Use the 'Add New Column' option to add a new partition column.


Partition Columns

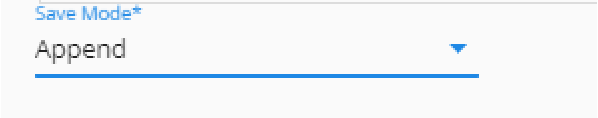
Name

Note:

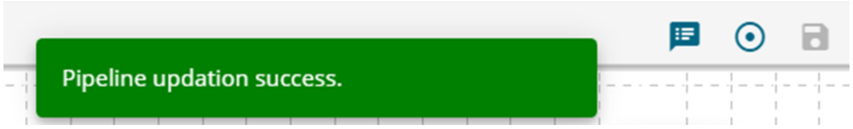
i. Users can add multiple Selected Columns and Partition Columns by clicking the '**Add New Column**' option.

ii. UUID Column Name, Selected Columns, and Partition Columns are optional.

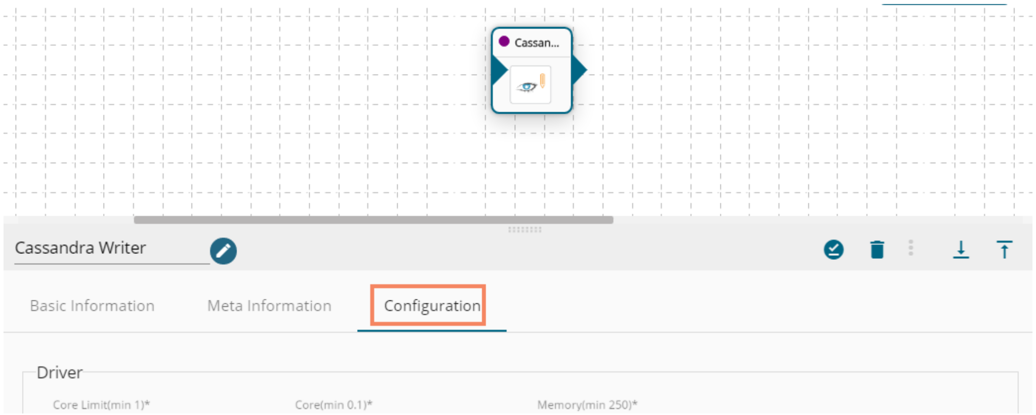
vii) Save Mode- Select the mode to save the data from the drop-down menu (The supported save mode is '**Append**').


Save Mode*
Append

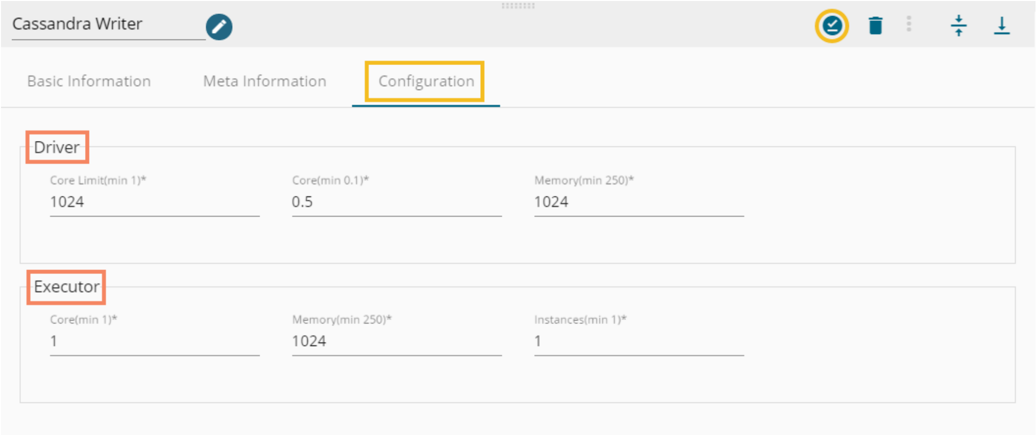
viii) Click the '**Save Component in Storage**' ✓ icon for the Cassandra Writer component.
ix) A message appears to notify the same.
x) Click the '**Update Pipeline**' icon to save the changes.
xi) A message appears to notify that the pipeline has been updated.


Pipeline updation success.

xii) Open the dragged Cassandra writer component again to see the '**Configuration**' properties tab below.



xiii) Modify the values for the Configuration tab and click the '**Save Component in Storage**' ✓ icon.


Cassandra Writer

Basic Information | Meta Information | Configuration

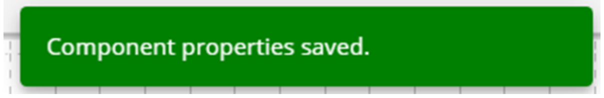
Driver

Core Limit(min 1)*: 1024
Core(min 0.1)*: 0.5
Memory(min 250)*: 1024

Executor

Core(min 1)*: 1
Memory(min 250)*: 1024
Instances(min 1)*: 1

xiv) The message appears to notify that the component properties are saved.

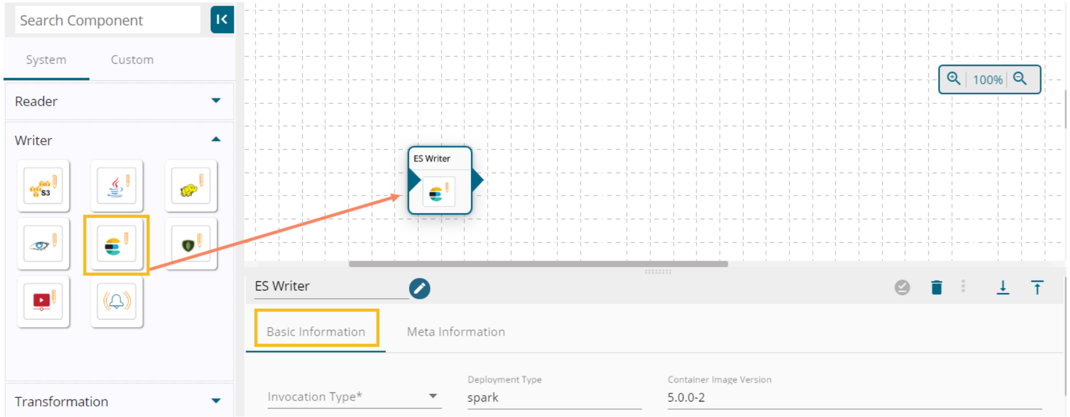

Component properties saved.

xv) The Cassandra Writer component is ready to read the data coming from an input event in the Pipeline Workflow.
Note: The input event can read data from Ingestion, Readers, and Shared Event.

8.2.5. ES Writer

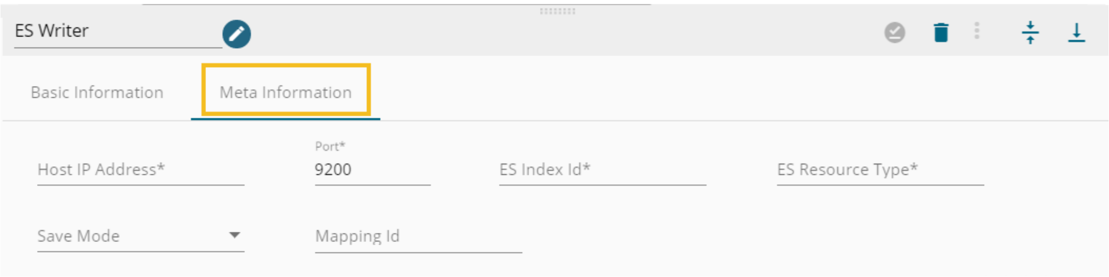
The ES writer writes the data to the Elastic Search engine.

i) Drag & drop the ES Writer component to the Workflow Editor.
ii) Click the dragged ES Writer component to open the component properties tabs below.
iii) Basic Information: It is the default tab to open for the ES Writer while configuring the component.
   1) Select an Invocation type from the drop-down menu to confirm the running mode of the reader component. Select either '**Real-Time**' or '**Batch**' from the drop-down menu.
   2) Deployment Type: It displays the deployment type for the component. This field comes pre-selected.
   3) Container Image Version: It displays the image version for the docker container. This field comes pre-selected.



Invocation Type*
Deployment Type: spark
Container Image Version: 5.0.0-2

iv) Open the '**Meta Information**' tab and fill all the connection-specific details for the ES Writer.
   a. Host IP Address
   b. Port number
   c. ES Index Id
   d. ES Resource Type,
   e. Save mode – it supports only the '**Append**' option
   f. Selected Columns- Data from the selected columns only stores in the ES location. Provide a column name, alias name, and column type. (Optional)
   g. Mapping Id- Provide mapping id (Optional)



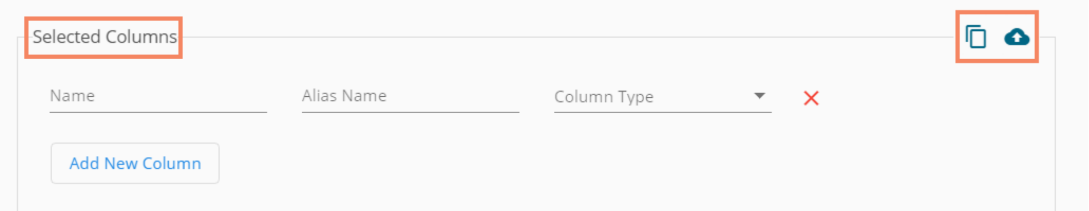
Host IP Address*
Port*: 9200
ES Index Id*
ES Resource Type*
Save Mode
Mapping Id

v) Selected Columns: The users can select some specific columns from the table to read data instead of selecting a complete table; this can be achieved via the ‘**Selected Columns**’ section. Select the columns which you want to read and if you want to change the name of the column, then put that name in alias name section otherwise keep alias name same as of column name and then select a Column Type from the drop-down menu.

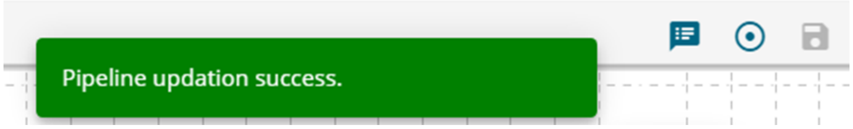
or

Use ‘**Download Data**’ and ‘**Upload File**’ options to select the desired columns.

1. Upload File: The user can upload the existing system files (CSV, JSON) using the ‘Upload File’ ☁ icon (file size must be less than 2 MB).
2. Download Data (Schema): Users can download the schema structure in JSON format by using the ‘Download Data’ ⧉ icon.

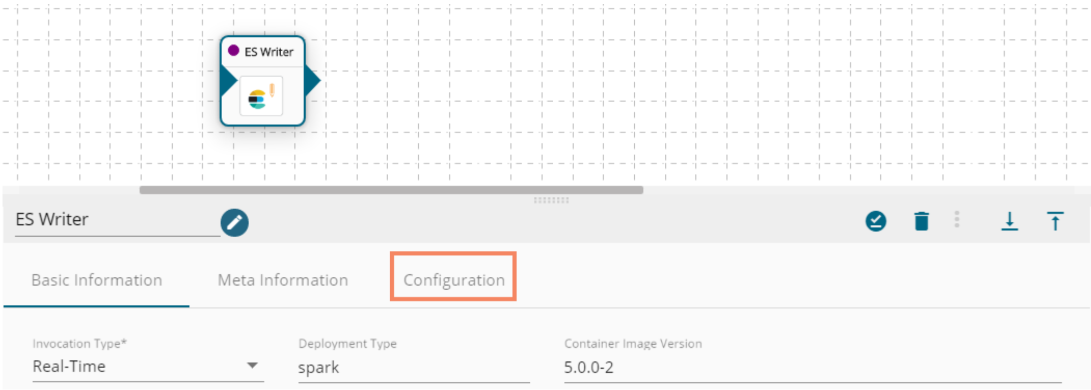


vi) Click the ‘**Save Component in Storage**’ ✓ icon for the ES Writer component.
vii) A message appears to notify the same.
viii) Click the ‘**Update Pipeline**’ icon to save the changes.
ix) A message appears to notify that the pipeline has been updated.



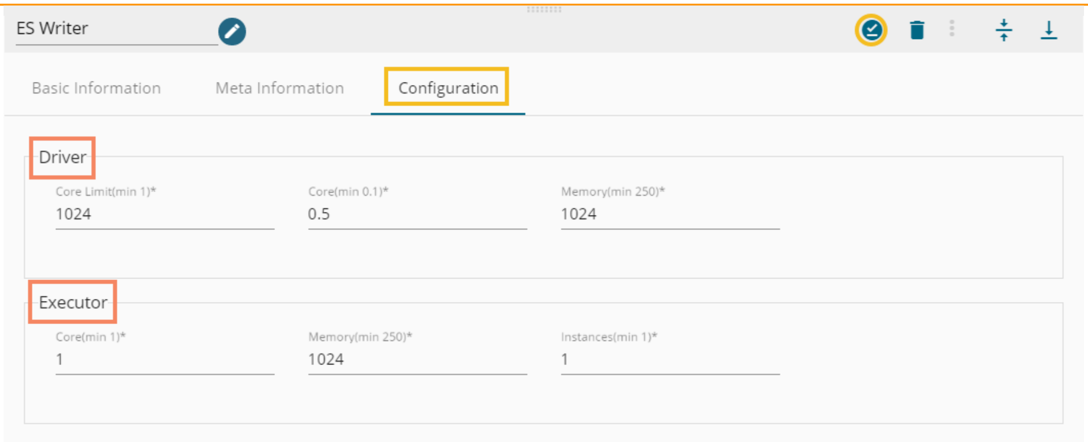
Pipeline updation success.

x) Open the dragged ES writer component again to see the ‘**Configuration**’ properties tab below.



Invocation Type*: Real-Time
Deployment Type: spark
Container Image Version: 5.0.0-2

xi) Modify the Configuration tab and click the ‘**Save Component in Storage**’ ✓ icon.


ES Writer

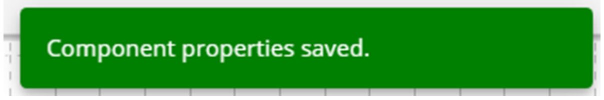
Driver

Core Limit(min 1)*: 1024
Core(min 0.1)*: 0.5
Memory(min 250)*: 1024

Executor

Core(min 1)*: 1
Memory(min 250)*: 1024
Instances(min 1)*: 1

xii) The message appears to notify that the component properties are saved.


Component properties saved.

xiii) The ES Writer component is ready to read the data coming from an input event in the Pipeline Workflow.

Note: The input event can read data from Ingestion, Readers, and Shared Event.

8.2.6. MongoDB Writer

The MongoDB writer writes the data to the database.

i) Drag & drop the MongoDB Writer component to the Workflow Editor.

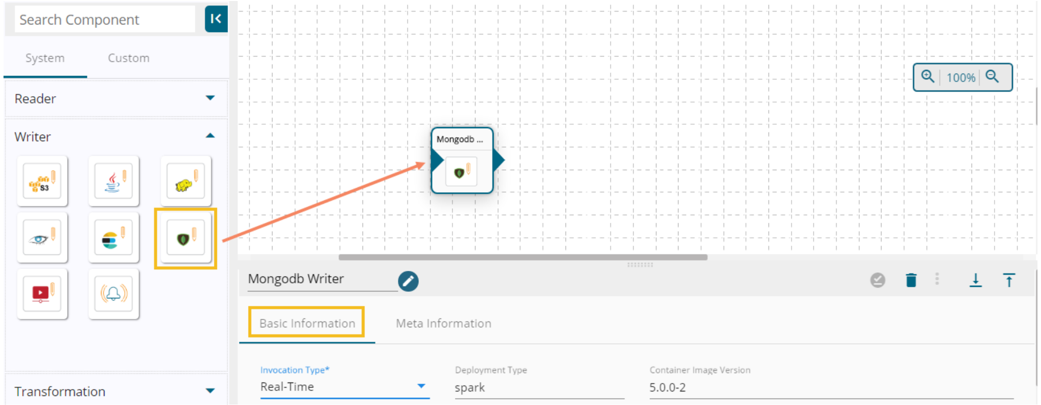
ii) Click the dragged MongoDB Writer component to open the component properties tabs below.

iii) Basic Information: It is the default tab to open for the MongoDB Writer while configuring the component.

1) Select an Invocation type from the drop-down menu to confirm the running mode of the reader component. Select ‘Real-Time’ or ‘Batch’ from the drop-down menu.

2) Deployment Type: It displays the deployment type for the component. This field comes pre-selected.

3) Container Image Version: It displays the image version for the docker container. This field comes pre-selected.



Invocation Type*: Real-Time
Deployment Type: spark
Container Image Version: 5.0.0-2

iv) Open the ‘**Meta Information**’ tab and fill all the connection-specific details for the MongoDB Writer.

a. Host IP Address(*)

b. Port number(*)

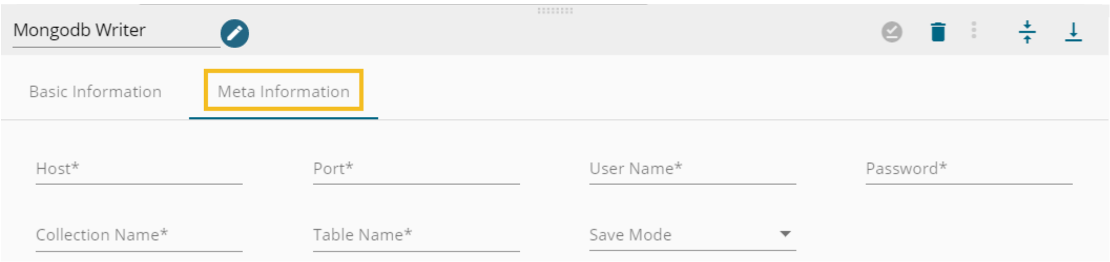
c. Username(*)

d. Password(*)

e. Collection Name(*)

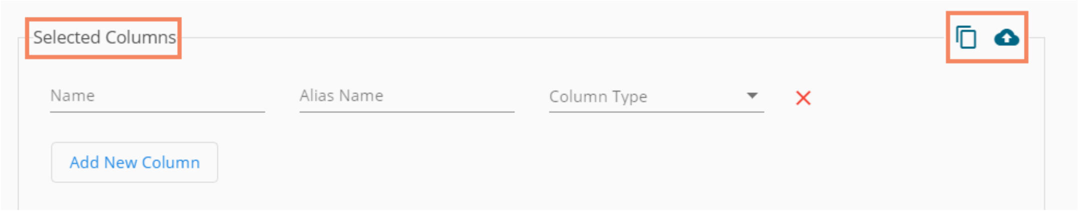
f. Table name(*)

g. Save Mode: select an option from the drop-down menu (the supported option is ‘Append’).



v) Selected Columns: The users can select some specific columns from the table to read data instead of selecting a complete table; this can be achieved via the ‘Selected Columns’ section. Select the columns which you want to read and if you want to change the name of the column, then put that name in alias name section otherwise keep alias name same as of column name and then select a Column Type from the drop-down menu.
or
Use ‘**Download Data**’ and ‘**Upload File**’ options to select the desired columns.

3. Upload File: The user can upload the existing system files (CSV, JSON) using the ‘**Upload File’** icon (file size must be less than 2 MB).
4. Download Data (Schema): Users can download the schema structure in JSON format by using the ‘**Download Data**’ icon.

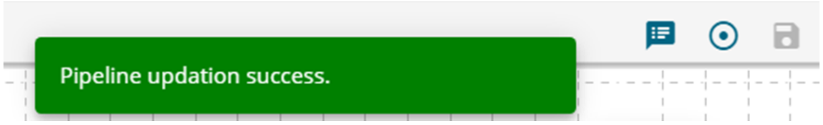


vi) Click the ‘**Save Component in Storage**’ icon for the MongoDB Writer component.

vii) A message appears to notify the same.

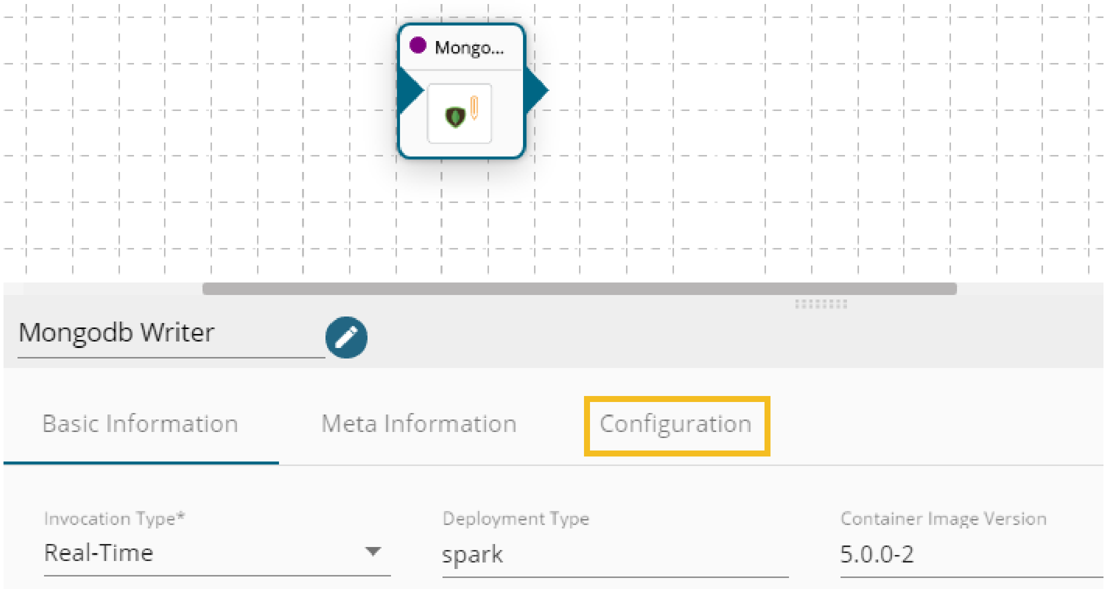
viii) Click the ‘**Update Pipeline**’ icon to save the changes.

ix) A message appears to notify that the pipeline has been updated.



Pipeline updation success.

x) Open the dragged MongoDB writer component again to see the ‘**Configuration**’ properties tab below.

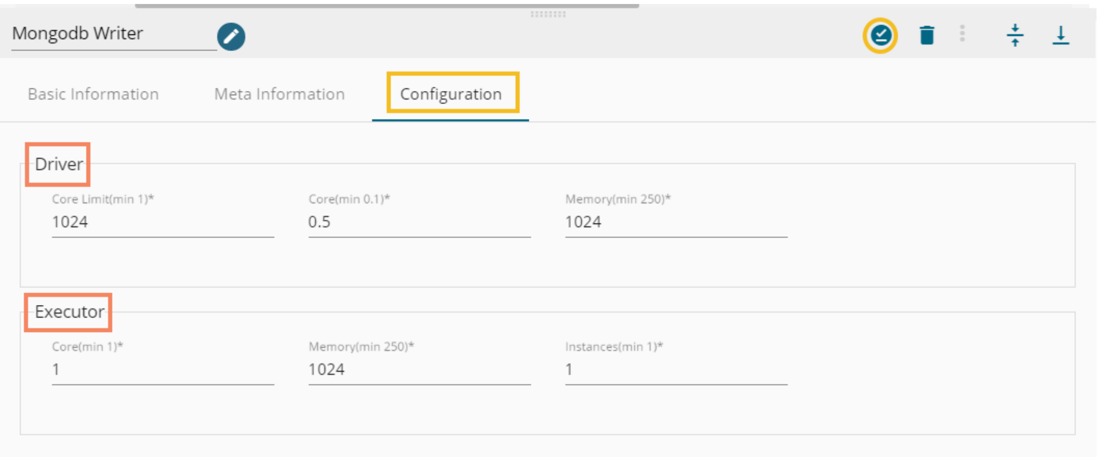


Mongodb Writer

Basic Information | Meta Information | Configuration

Invocation Type*: Real-Time
Deployment Type: spark
Container Image Version: 5.0.0-2

xi) Modify the Configuration tab and click the '**Save Component in Storage**' ✓ icon.

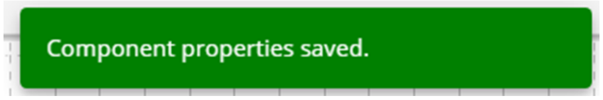

Mongodb Writer

Basic Information | Meta Information | Configuration

Driver
Core Limit(min 1)*: 1024
Core(min 0.1)*: 0.5
Memory(min 250)*: 1024

Executor
Core(min 1)*: 1
Memory(min 250)*: 1024
Instances(min 1)*: 1

xii) The message appears to notify that the component properties are saved.


Component properties saved.

xiii) The MongoDB Writer component is ready to read the data coming from an input event in the Pipeline Workflow.

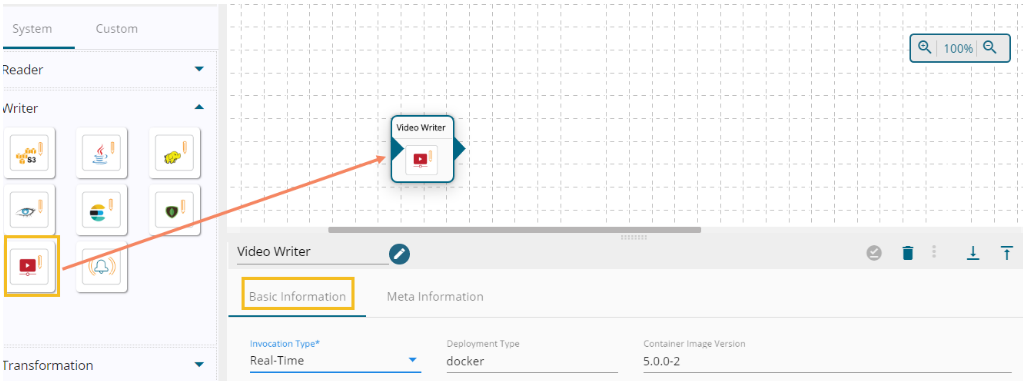
Note: The input event can read data from Ingestion, Readers, and Shared Event.

8.2.7. Video Writer

The Video writer writes the input video data.

i) Drag & drop the Video Writer component to the Workflow Editor.
ii) Click the dragged Video Writer component to open the component properties tabs below.
iii) Basic Information: It is the default tab to open for the Video Writer while configuring the component.

1) Select an Invocation type from the drop-down menu to confirm the running mode of the reader component. Select 'Real-Time' or 'Batch' from the drop-down menu.
2) Deployment Type: It displays the deployment type for the component. This field comes pre-selected.
3) Container Image Version: It displays the image version for the docker container. This field comes pre-selected.



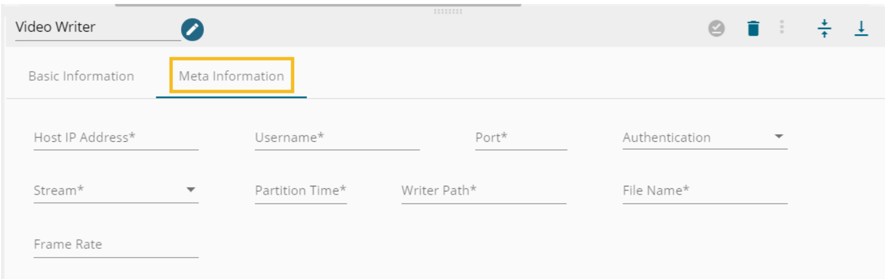
Video Writer

Basic Information | Meta Information

Invocation Type*: Real-Time
Deployment Type: docker
Container Image Version: 5.0.0-2

iv) Open the '**Meta Information**' tab and fill all the connection-specific details for the Video Writer.

a. Host IP Address(*)
b. Username(*)
c. Port number(*)
d. Authentication
e. Stream(*)
f. Partition Time(*)
g. Writer Path(*)
h. File Name(*)
i. Frame Rate



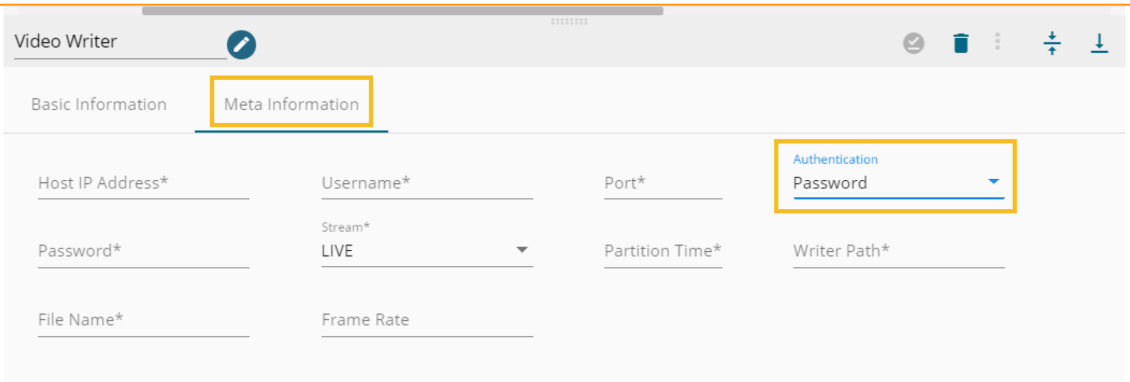
Video Writer

Basic Information | Meta Information

Host IP Address*
Username*
Port*
Authentication
Stream*
Partition Time*
Writer Path*
File Name*
Frame Rate

The fields for the Meta Information tab changes based on the selection of the Authentication option.

When the authentication option is Password



Host IP Address*

Username*

Port*

Authentication: Password

Password*

Stream*: LIVE

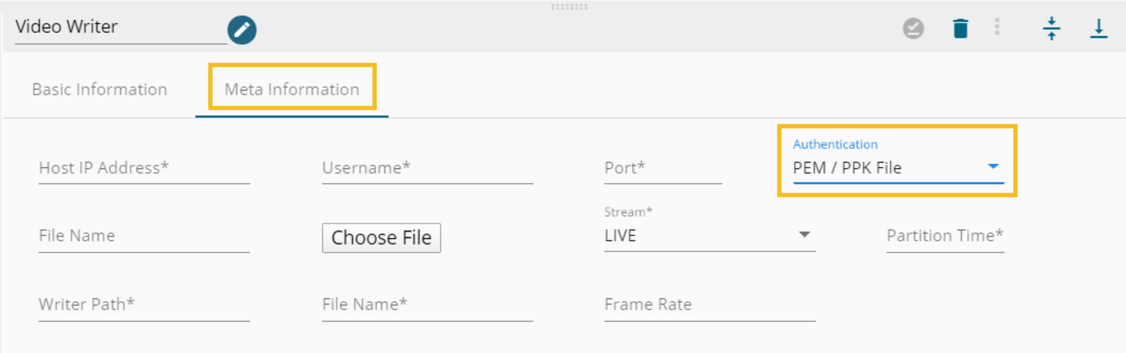
Partition Time*

Writer Path*

File Name*

Frame Rate

When the authentication option is PEM/PPK file



Host IP Address*

Username*

Port*

Authentication: PEM / PPK File

File Name

Stream*: LIVE

Partition Time*

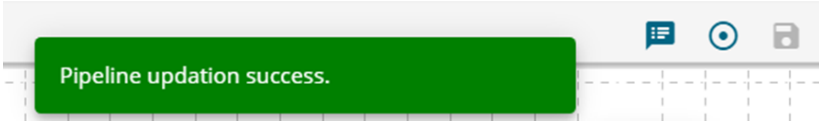
Writer Path*

File Name*

Frame Rate

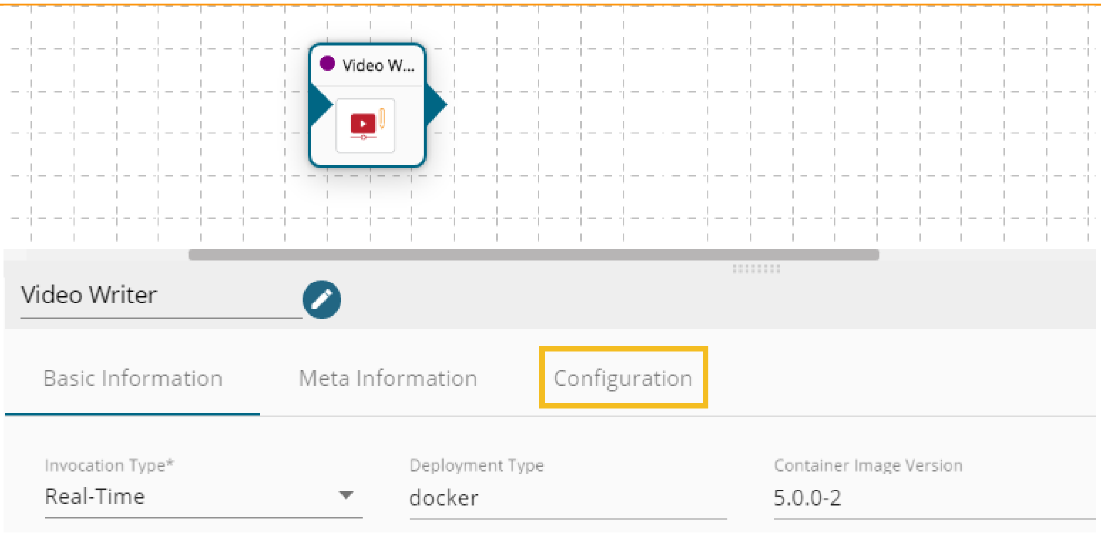
While choosing the PEM/PPK File authentication option, the user needs to select a file using the Choose File option.

v) Click the '**Save Component in Storage**' icon for the Video Writer component.
vi) A message appears to notify the same.
vii) Click the '**Update Pipeline**' icon to save the changes.
viii) A message appears to notify that the pipeline has been updated.



Pipeline updation success.

ix) Open the dragged Video writer component again to see the 'Configuration' properties tab below.



Video Writer

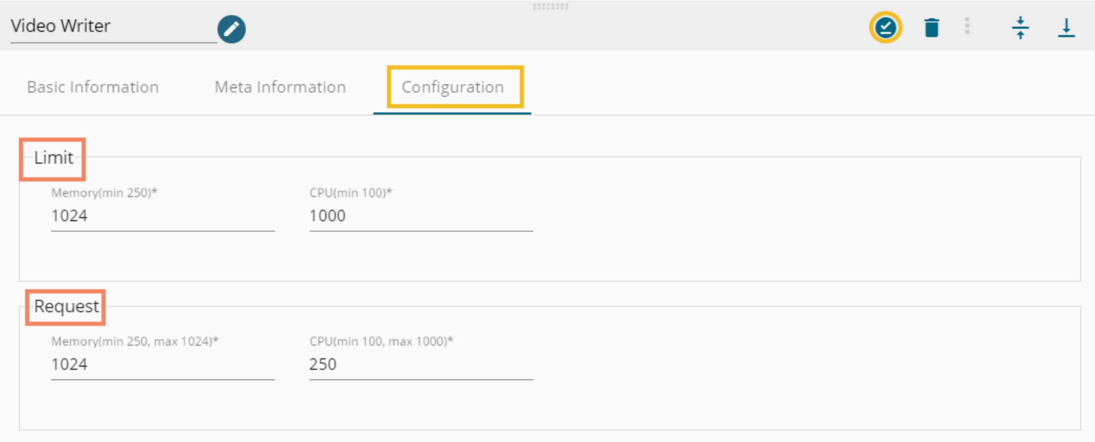
Basic Information | Meta Information | Configuration

Invocation Type*: Real-Time

Deployment Type: docker

Container Image Version: 5.0.0-2

x) Modify the Configuration tab and click the '**Save Component in Storage**' ✓ icon.



Video Writer

Basic Information | Meta Information | Configuration

Limit

Memory(min 250)*: 1024

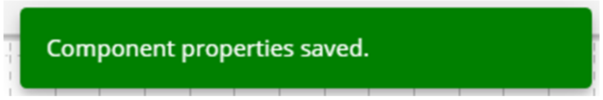
CPU(min 100)*: 1000

Request

Memory(min 250, max 1024)*: 1024

CPU(min 100, max 1000)*: 250

xi) The message appears to notify that the component properties are saved.



Component properties saved.

xii) The Video Writer component is ready to read the data coming from an input event in the Pipeline Workflow.

Note: The input event can read data from Ingestion, Readers, and Shared Event.

8.2.8. Notification Publisher

Notification Publisher Component as the name suggests is meant to publish the messages coming to Kafka as a notification. It is very helpful in acting scenarios where an immediate reaction is required in some conditions.

Example: A pipeline is monitoring an IoT sensor that sends the temperature of a machine. If the temperature reaches a certain threshold; the process engineer is to be notified immediately to maintain the same to stabilize the machine. Here, the Notification Publisher component is useful.

i) Drag and drop the Notification Publisher component to the Workflow Editor.

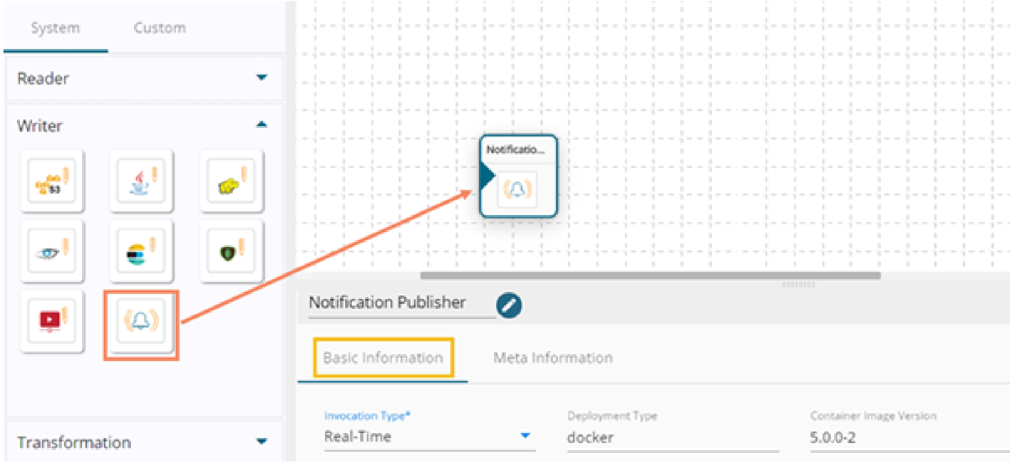
ii) Click the 'Notification Publisher' component to open the component properties below.

iii) Basic Information: It is the default tab to open for the Notification Publisher while configuring the component.

1) Select an Invocation type from the drop-down menu to confirm the running mode of the reader component. Select 'Real-Time' or 'Batch' from the drop-down menu.

2) Deployment Type: It displays the deployment type for the component. This field comes pre-selected.

3) Container Image Version: It displays the image version for the docker container. This field comes pre-selected.



Invocation Type* Real-Time

Deployment Type docker

Container Image Version 5.0.0-2

iv) Open the '**Meta Information**' tab and fill all the connection-specific details for the Notification Writer.

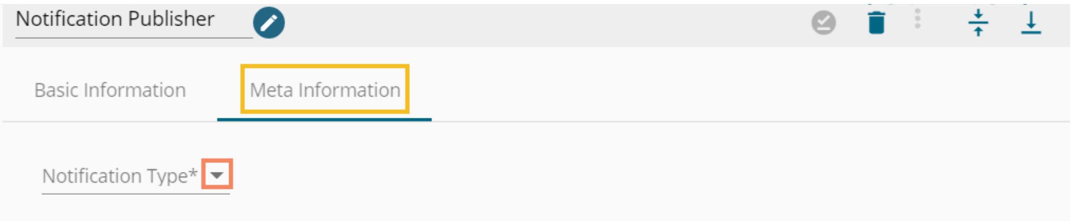
a. Type: Select a notification type using the drop-down menu. The provided options are:

i. SNS

ii. Google Pub/Sub

b. Based on the selected Notification Type the remaining configuration fields get displayed.

c. The user needs to complete the configuration process for the selected notification type and click the 'Save' ✓icon to complete the configuration process for the Notification Publisher.



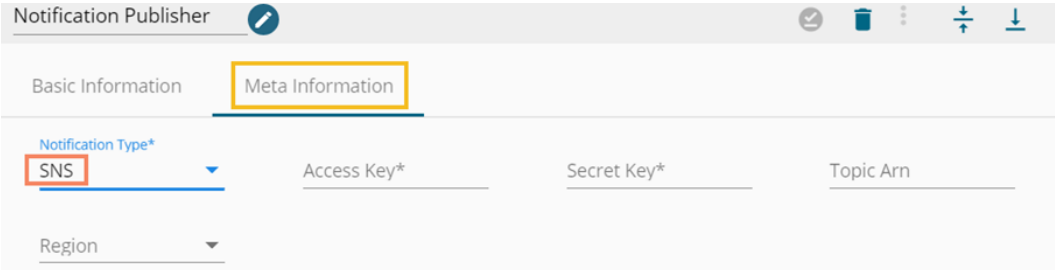
8.2.8.1. SNS as Notification Type

i) Navigate to the Meta Information tab provided for the Notification Publisher.

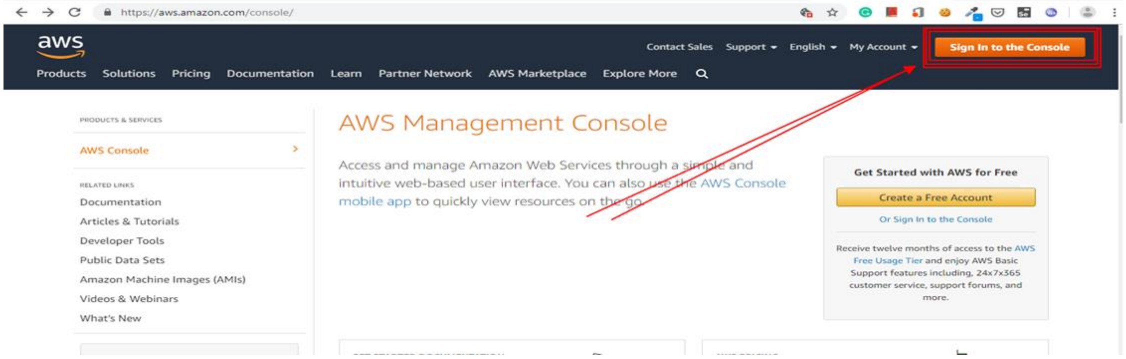
ii) Select the 'SNS' as the Notification Type option using the drop-down menu.

iii) The following configuration fields appear:

a. Access Key: Access keys consist of two parts: an access key ID (for example, AKIAIOSFODNN7EXAMPLE) can be retrieved from AWS key management.
b. Secret Key: A secret access key (for example, wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY) can also be retrieved from the AWS key management.
c. Topic Arn: The field gets filled after the topic creation. Navigate to Topics and then click the desired topic.
d. Region: Select an option from the drop-down menu where the topic has been created.



iv) The user needs to follow some more steps to create and subscribe to a topic. The below given description explains those steps:
a. Navigate to the URL: https://aws.amazon.com/console/
b. Click the '**Sign In to the Console**' option.

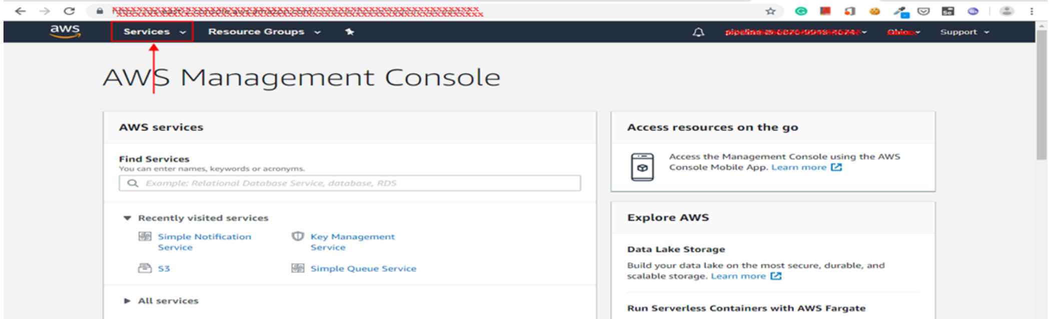


AWS Management Console

Access and manage Amazon Web Services through a simple and intuitive web-based user interface. You can also use the AWS Console mobile app to quickly view resources on the go.

Receive twelve months of access to the AWS Free Usage Tier and enjoy AWS Basic Support features including, 24x7x365 customer service, support forums, and more.

c. Sign in with your credentials.
d. The AWS Management Console opens.
e. Click the '**Services**' option from the header drop-down.
f. Search SNS service using the '**Find Services**' space.


[redacted]

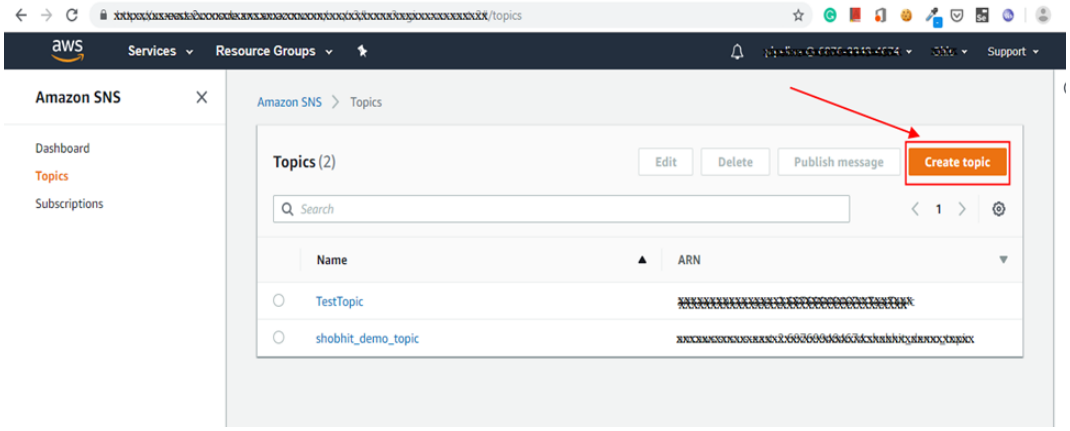
You can enter names, keywords or acronyms.

Example: Relational Database Service, database, RDS

Access the Management Console using the AWS Console Mobile App. Learn more

Build your data lake on the most secure, durable, and scalable storage. Learn more

g. The Amazon SNS page opens.
h. Click the '**Create topic**' option.


[redacted]/topics

[redacted]

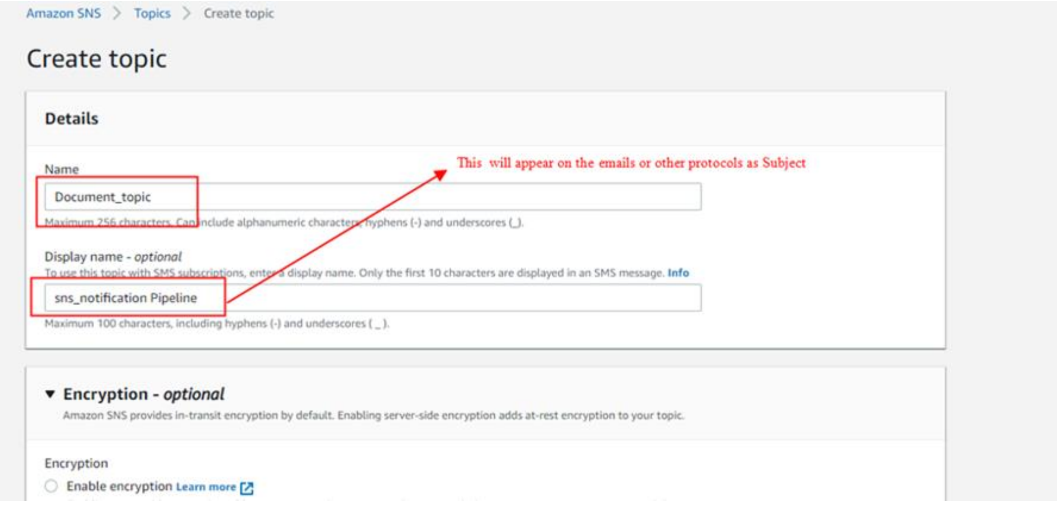
[redacted]

Topics (2)

	Name	ARN
○	TestTopic	[redacted]
○	shobhit_demo_topic	[redacted]

i. The Create topic page opens.

j. Provide essential information for the topic.


Amazon SNS > Topics > Create topic
This will appear on the emails or other protocols as Subject

Create topic

Details

Name

Document_topic

Maximum 256 characters. Can include alphanumeric characters, hyphens (-) and underscores (_).

Display name - *optional*

To use this topic with SMS subscriptions, enter a display name. Only the first 10 characters are displayed in an SMS message. Info

sns_notification Pipeline

Maximum 100 characters, including hyphens (-) and underscores (_).

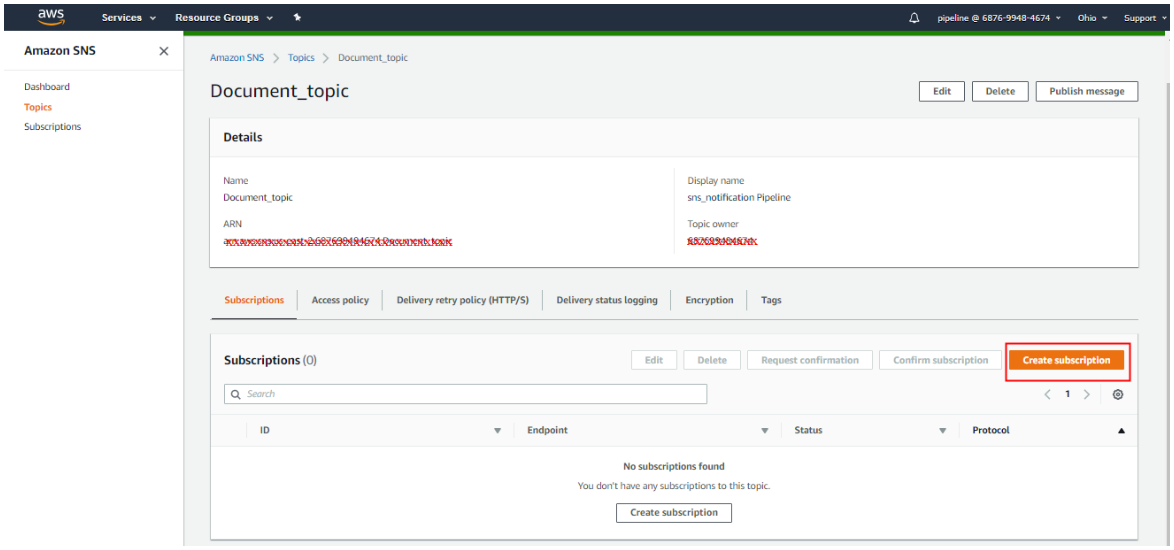
Encryption - *optional*

Amazon SNS provides in-transit encryption by default. Enabling server-side encryption adds at-rest encryption to your topic.

Encryption

○ Enable encryption Learn more

k. Click the '**Create Subscription**' option.



Document_topic

Details

Name
Document_topic

Display name
sns_notification Pipeline

ARN
[redacted]

Topic owner
[redacted]

Subscriptions (0)

ID	Endpoint	Status	Protocol

No subscriptions found

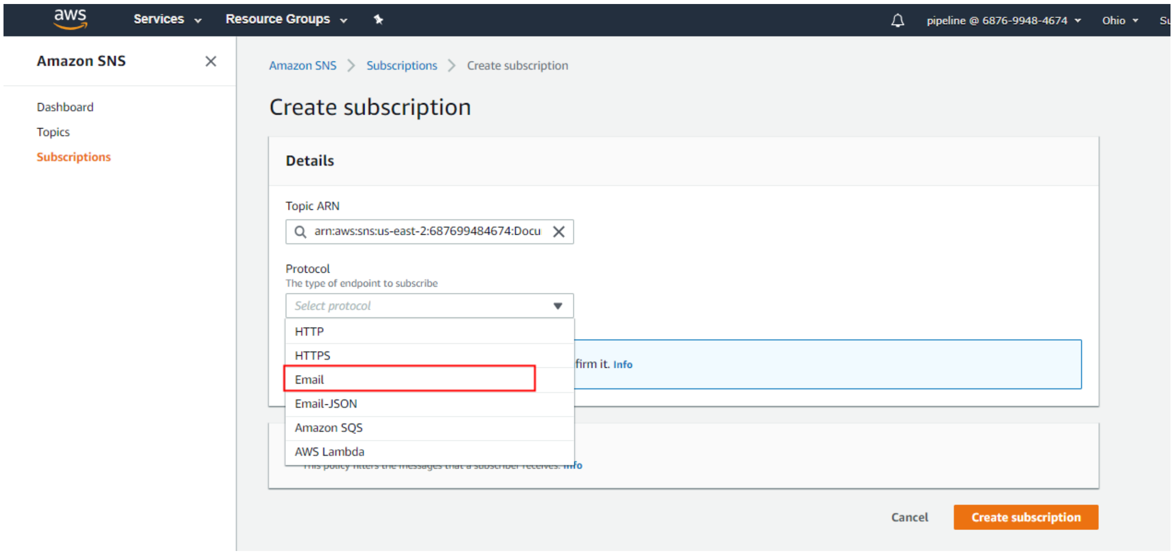
You don't have any subscriptions to this topic.

Create subscription

l. Search or provide the topic ARN.

m. Select a protocol from the available list.

n. Click the ‘**Create Subscription**’ option.



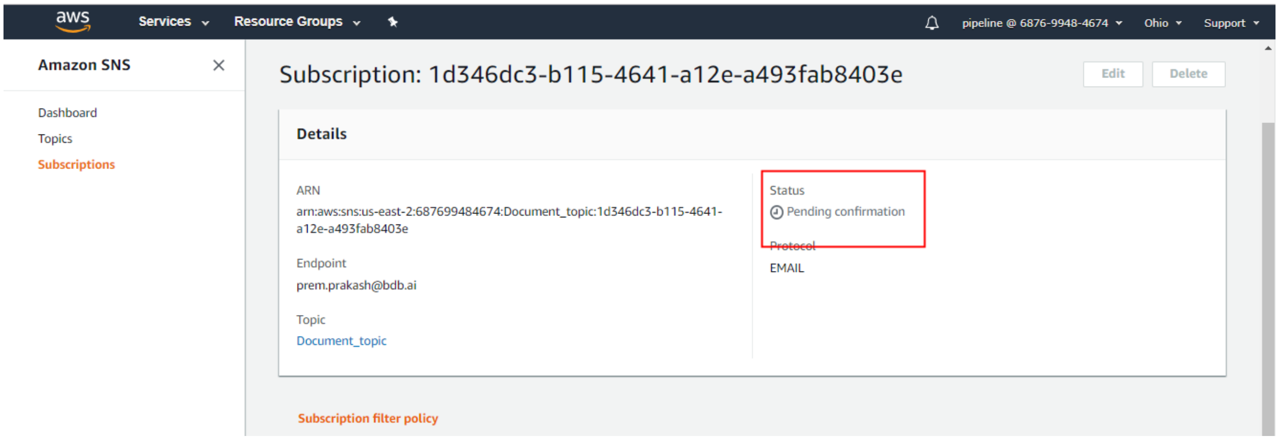
Create subscription

Details

Topic ARN: arn:aws:sns:us-east-2:687699484674:Docu

Protocol
The type of endpoint to subscribe
Select protocol

NOTE: The user may get ‘**Pending Confirmation**’ notification of subscriptions as shown in the below image. In such cases navigate to the email id and confirm the subscription request.



Subscription: 1d346dc3-b115-4641-a12e-a493fab8403e

Details

ARN: arn:aws:sns:us-east-2:687699484674:Document_topic:1d346dc3-b115-4641-a12e-a493fab8403e

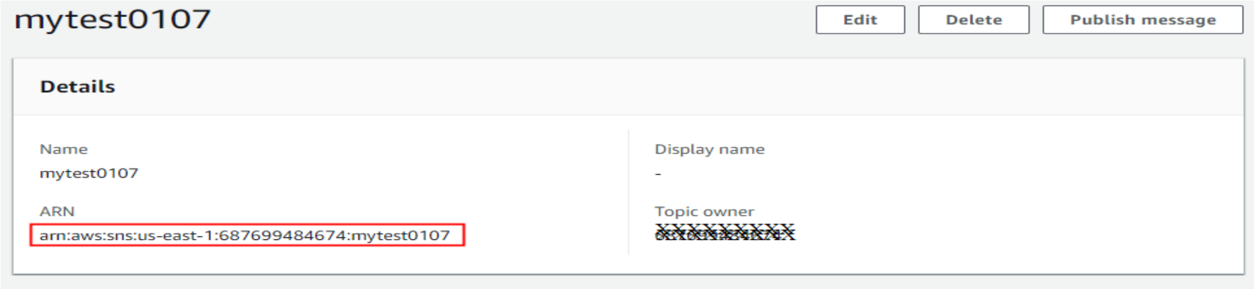
Endpoint: prem.prakash@bdb.ai

Topic: Document_topic

Status: Pending confirmation

Protocol: EMAIL

o. Copy the ARN of the Topic.



mytest0107

Details

Name: mytest0107

ARN: arn:aws:sns:us-east-1:687699484674:mytest0107

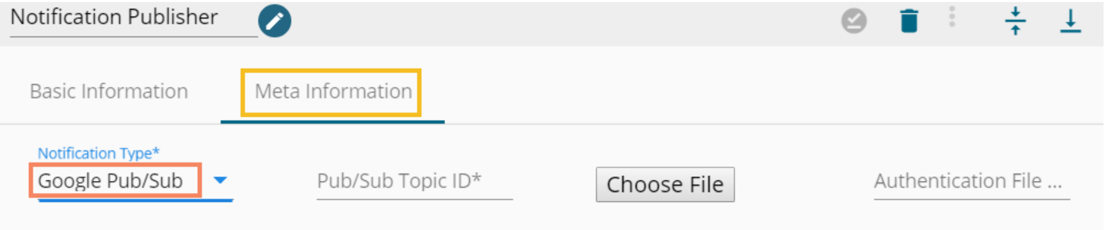
Display name: -

Topic owner: [redacted]

8.2.8.2. Google Pub/Sub as the selected Notification Type

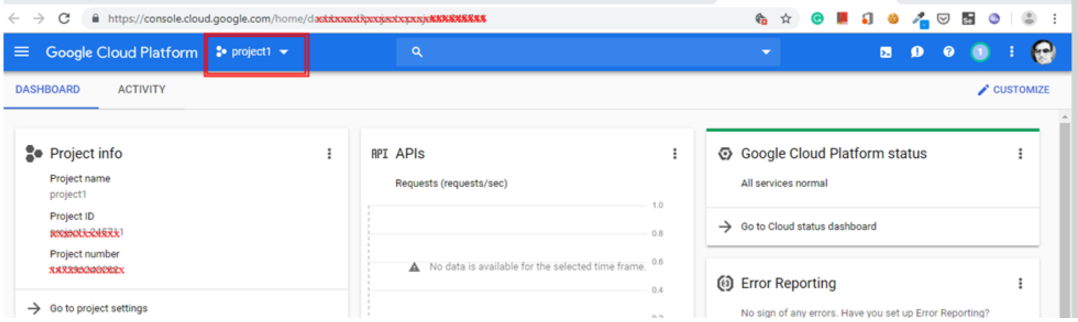
i) Navigate to the Meta Information tab provided for the Notification Publisher.

ii) Select the ‘**Google Pub/Sub**’ as the Notification Type option using the drop-down menu.



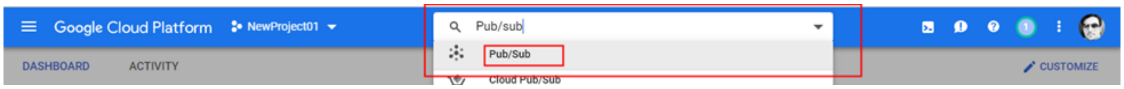
Notification Type*: Google Pub/Sub

iii) The user needs to follow some more steps to get the Pub/Subtopic ID
iv) Navigate to the https://console.cloud.google.com/
v) Log in with your credentials.
vi) The Landing page opens.
vii) Click on the marked area.
viii) Create a Project or use the desired one.

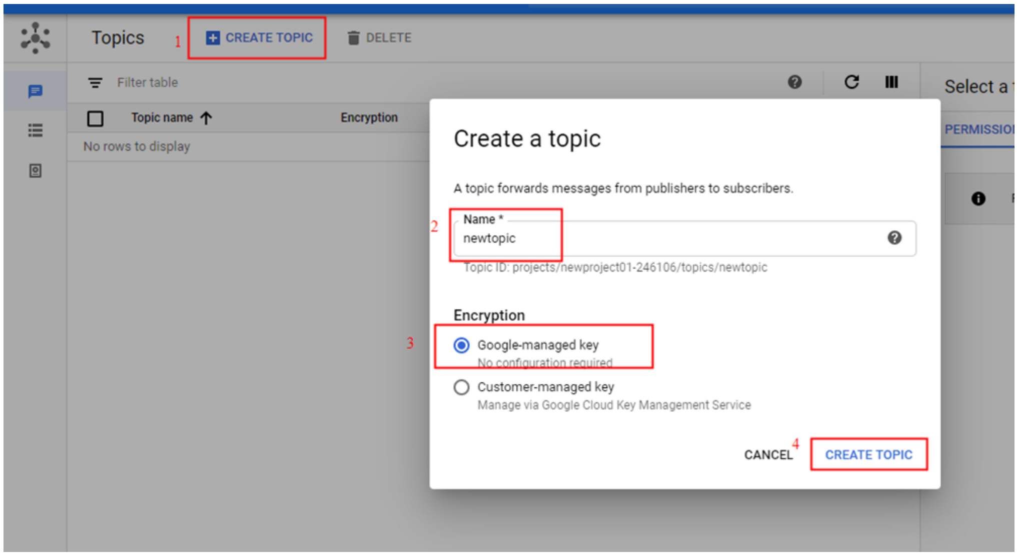

https://console.cloud.google.com/home/d[redacted]
Google Cloud Platform
project1
DASHBOARD
ACTIVITY
CUSTOMIZE
Project info
APIs
Requests (requests/sec)
1.0
0.8
0.6
0.4
No data is available for the selected time frame.
Google Cloud Platform status
All services normal
Go to Cloud status dashboard
Error Reporting
No sign of any errors. Have you set up Error Reporting?
Go to project settings

Project name: project1
Project ID: [redacted]
Project number: [redacted]

ix) Search for the Pub/Sub option in the Search bar.



x) Create a Topic using the following steps:
   a. Click the ‘**CREATE TOPIC**’ option.
   b. Provide a name for the new topic
   c. Select the ‘**Google-managed key**’ as the Encryption option.
   d. Click the ‘**CREATE TOPIC**’ option.



Create a topic

A topic forwards messages from publishers to subscribers.

Name *: newtopic

Topic ID: projects/newproject01-246106/topics/newtopic

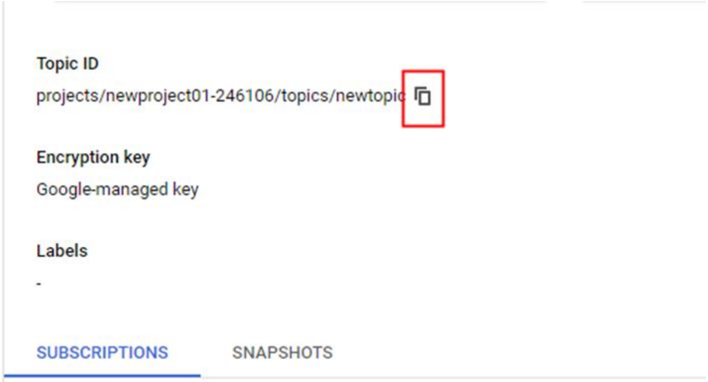
Encryption

● Google-managed key
No configuration required

○ Customer-managed key
Manage via Google Cloud Key Management Service

xi) The user gets either of the following screens:

a. While using an existing Topic ID



Topic ID

projects/newproject01-246106/topics/newtopic

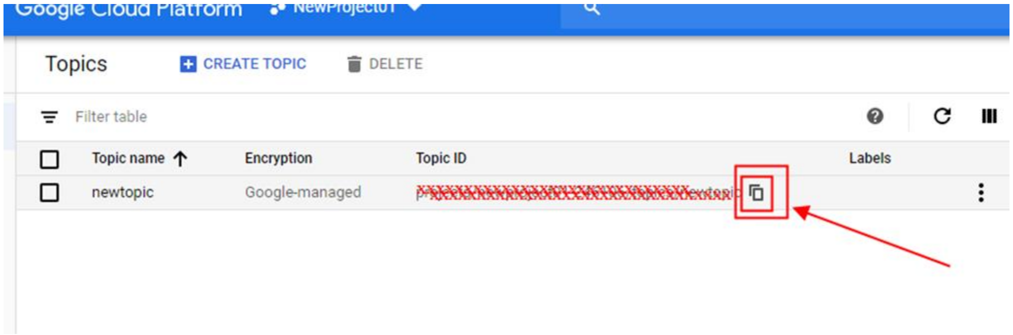
Encryption key

Google-managed key

Labels

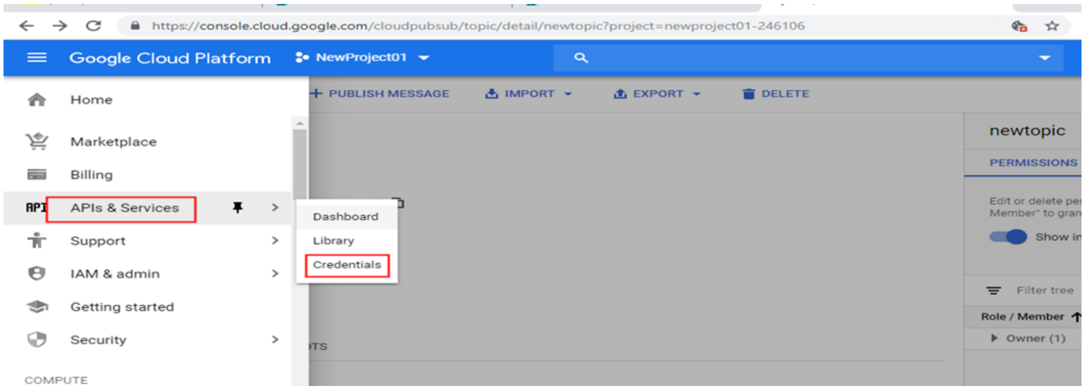
-

Alternatively, while using the newly created Topic ID



Topic name	Encryption	Topic ID	Labels
newtopic	Google-managed	[redacted]	

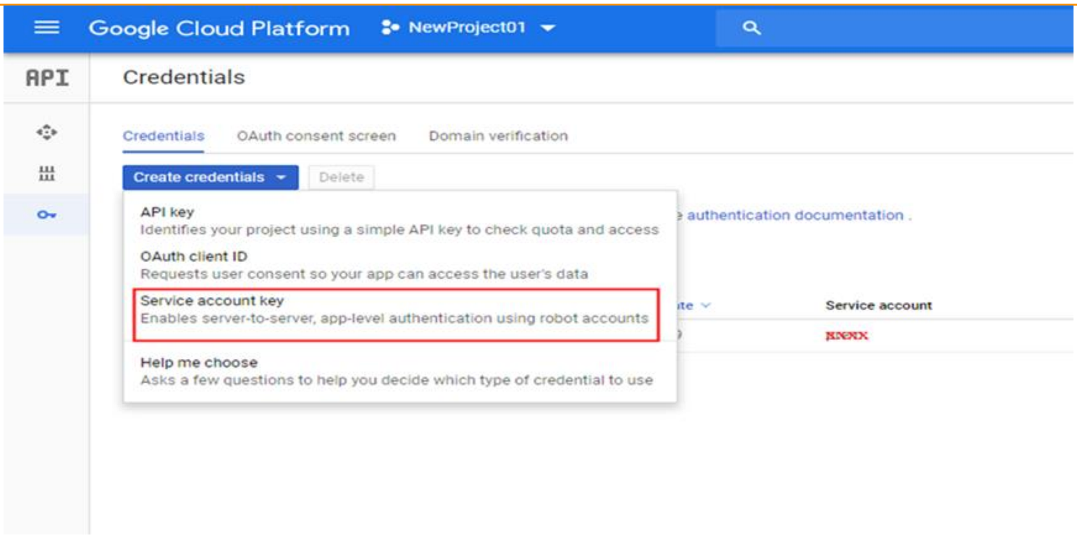
b. Create Auth-JSON for the current Project.



https://console.cloud.google.com/cloudpubsub/topic/detail/newtopic?project=newproject01-246106

c. Create a Service account key by following the given steps:

i. Click the '**Service account key**' option from the Credentials tab.



API key
Identifies your project using a simple API key to check quota and access

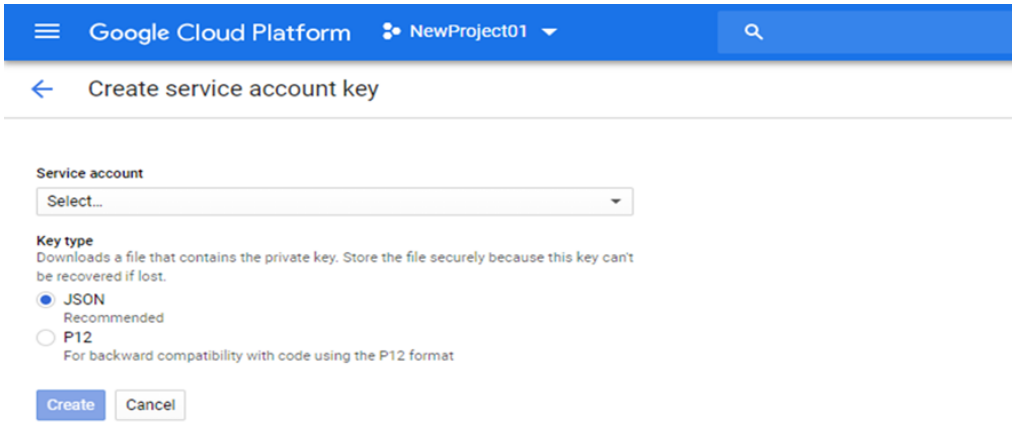
OAuth client ID
Requests user consent so your app can access the user's data

Service account key
Enables server-to-server, app-level authentication using robot accounts

Help me choose
Asks a few questions to help you decide which type of credential to use

ii. Select a Service account using the drop-down menu.
iii. Download the JSON file using the radio button.
iv. Click the '**Create**' option.

Use the copied Topic ID or the service account as metadata input.



Service account
Select...

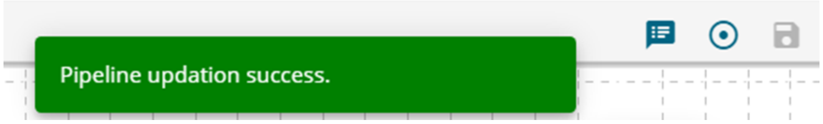
Key type
Downloads a file that contains the private key. Store the file securely because this key can't be recovered if lost.

● JSON
Recommended

○ P12
For backward compatibility with code using the P12 format

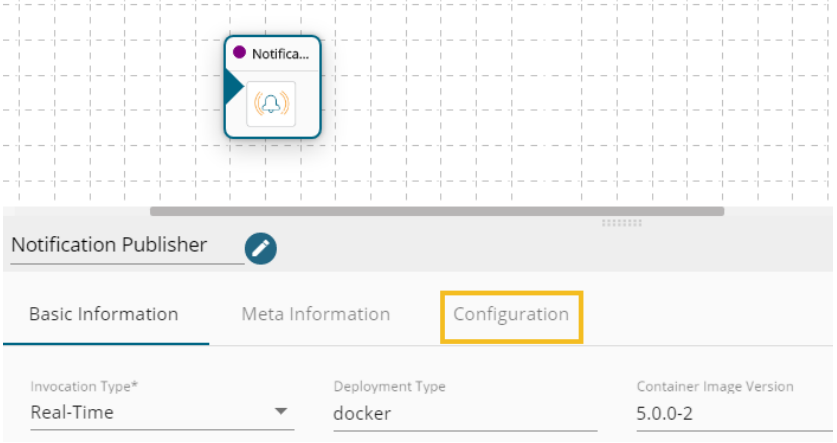
.

xii) Click the '**Save Component in Storage**' icon to save the component Properties for the Notification Publisher.
xiii) A message appears to notify the same.
xiv) Click the '**Update Pipeline**' icon to save the changes.
xv) A message appears to notify that the pipeline has been updated.



Pipeline updation success.

xvi) Open the dragged Notification Publisher component again to see the 'Configuration' properties tab below.



Notification Publisher

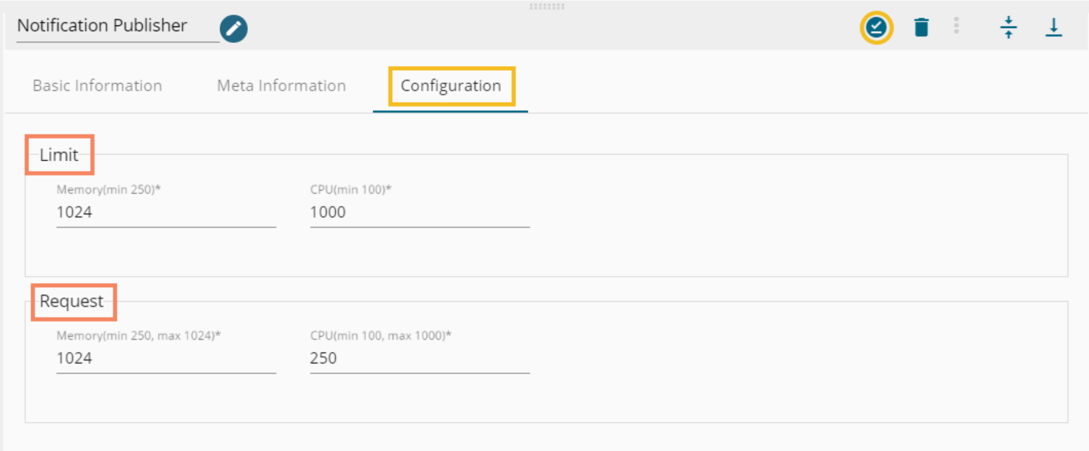
Basic Information | Meta Information | Configuration

Invocation Type*: Real-Time

Deployment Type: docker

Container Image Version: 5.0.0-2

xvii) Modify the Configuration tab and click the '**Save Component in Storage**' ✓ icon.



Notification Publisher

Basic Information | Meta Information | Configuration

Limit

Memory(min 250)*: 1024

CPU(min 100)*: 1000

Request

Memory(min 250, max 1024)*: 1024

CPU(min 100, max 1000)*: 250

xviii) The message appears to notify that the component properties are saved.

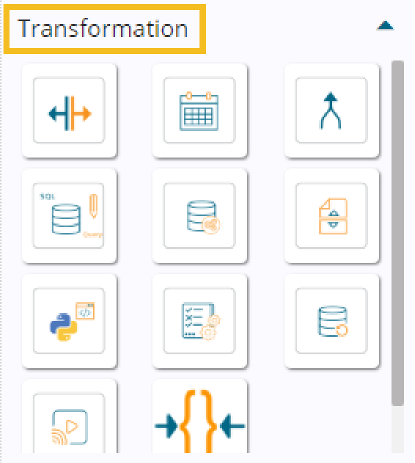
Component properties saved.

xix) The Notification Publisher component is ready to read the data coming from an input event in the Pipeline Workflow.

Note: The input event can read data from Ingestion, Readers, and Shared Event.

8.3. Transformation

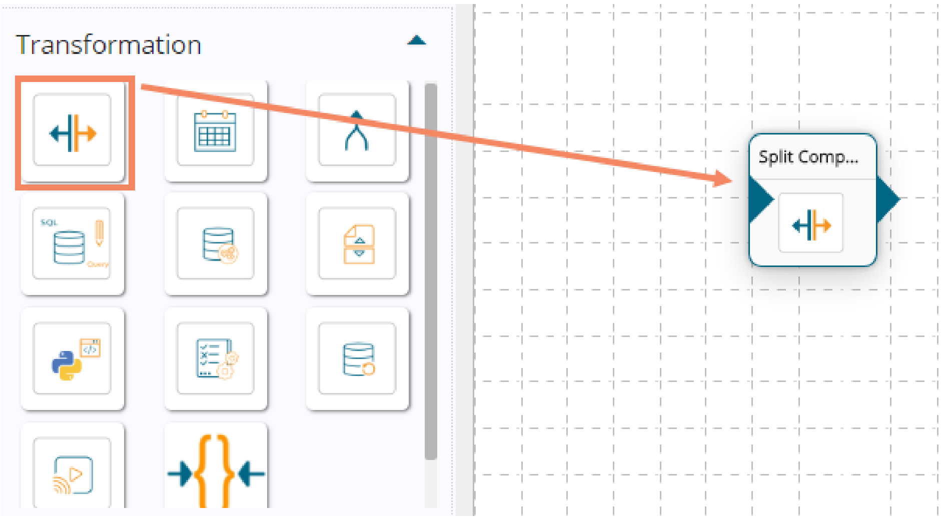
Transformation components allow users to transform the data. BDB Data Pipeline provides the following Transformation components to transform a variety of data.



8.3.1. Split Component

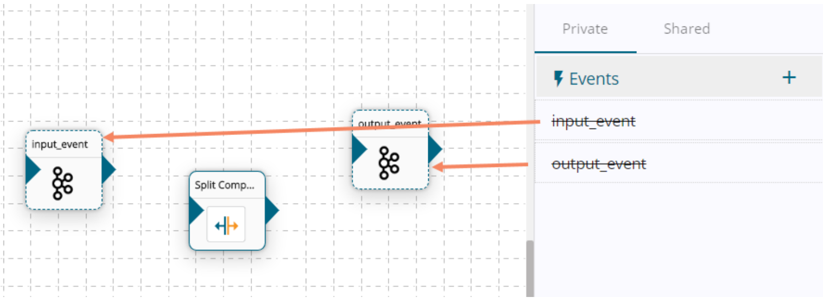
The Split component helps users to split the selected column(s) from the input data set based on the given regular expression.

i) Drag & drop the Split Component to the Workflow Editor.

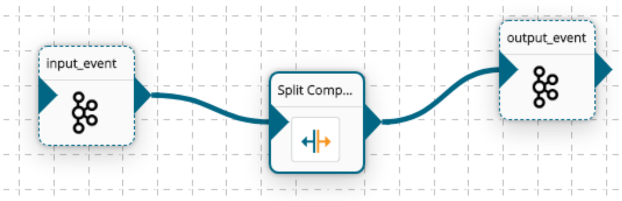


ii) The transformation component requires an input event (to get the data) and sends the data to an output event.

iii) Create two Events and drag them to the Workspace.



iv) Connect the input event and the output event (The data in the input event can come from any Ingestion, Readers or shared events).



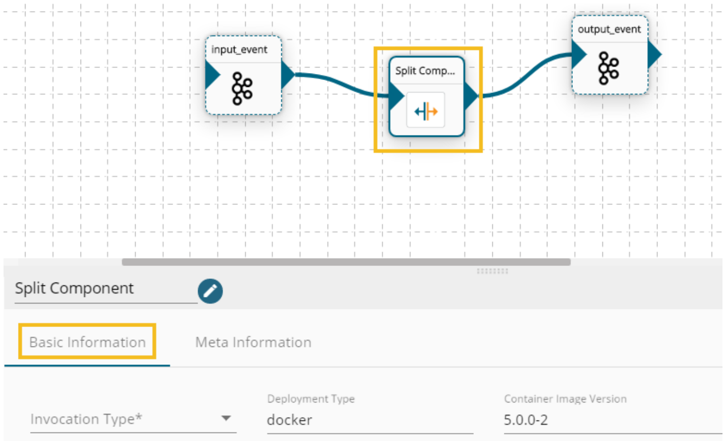
v) Click the dragged Split component to get the component properties tabs below.

vi) **Basic Information**: It is the default tab to open for the Split Component while configuring the component.

a. Select an Invocation type from the drop-down menu to confirm the running mode of the reader component. Select ‘**Real-Time**’ or ‘**Batch**’ from the drop-down menu.

b. Deployment Type: It displays the deployment type for the component. This field comes pre-selected.

c. Container Image Version: It displays the image version for the docker container. This field comes pre-selected.



Invocation Type*

Deployment Type: docker

Container Image Version: 5.0.0-2

vii) Open the ‘**Meta Information**’ **tab** and fill all the connection-specific details for the Notification Writer.

a. Fill the Regular Expression by which the data gets split (E.g., Comma (,) is used as the Regular Expression in the below given image).

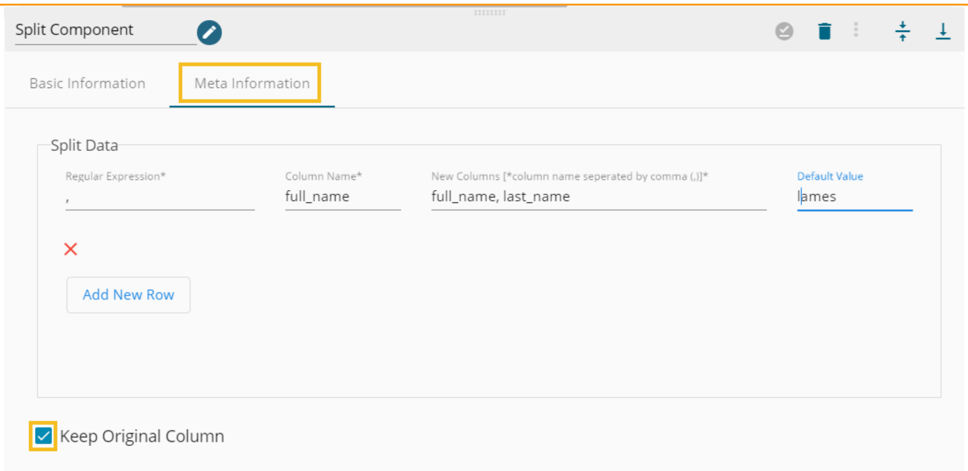
b. Column Name- Provide the input column name.

c. New Columns- Provide Output columns name separated by a comma. These output columns get created after transformation.

d. Provide Default Value in the given field.

e. To perform multiple Split transformation actions, click the ‘**Add New Row**’ option and follow the above steps.

f. Enable the ‘**Keep Original Column**’ option to keep the original column.



Regular Expression*: ,

Column Name*: full_name

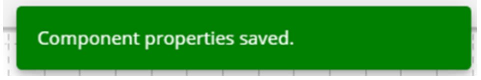
New Columns [*column name seperated by comma (,)]*: full_name, last_name

Default Value: James

☑ Keep Original Column

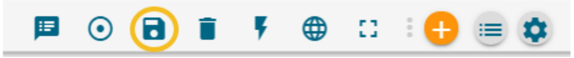
viii) Click the '**Save Component in Storage**' icon to save the configured properties for the dragged Aggregation component.

ix) A message appears to notify the successful update of the component.



Component properties saved.

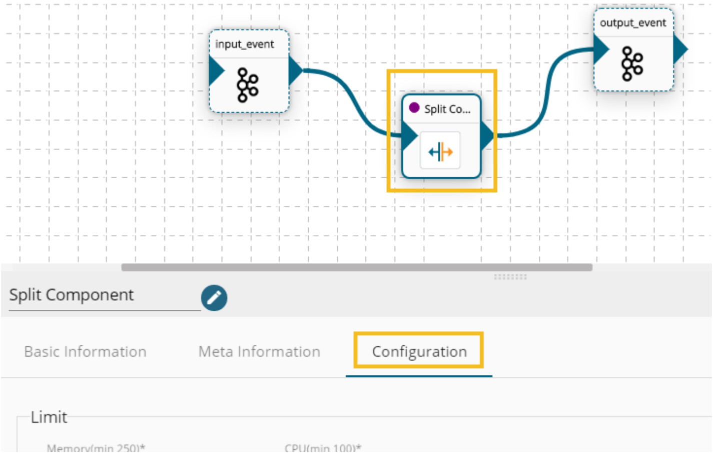
x) Click the '**Update Pipeline**' icon.



xi) A message appears to confirm the action.

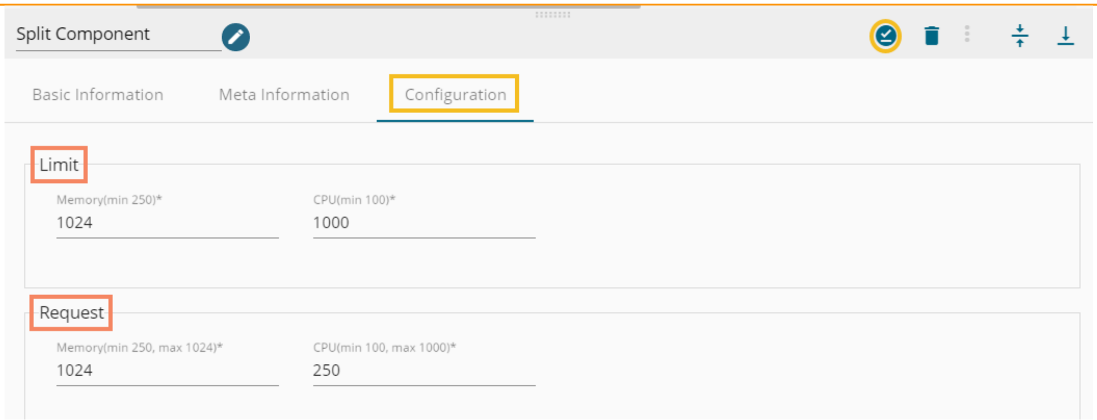
Pipeline updation success.

xii) Click the Dragged Split component again to see the Configuration tab below.



xiii) Modify the values given in the Configuration tab.

xiv) Click the '**Save Component in Storage**' icon.


Split Component

Basic Information

Meta Information

Configuration

Limit

Memory(min 250)*
1024

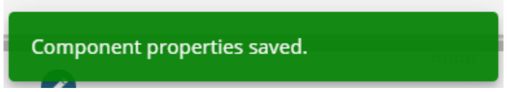
CPU(min 100)*
1000

Request

Memory(min 250, max 1024)*
1024

CPU(min 100, max 1000)*
250

xv) A message appears to notify the successful update of the component properties.

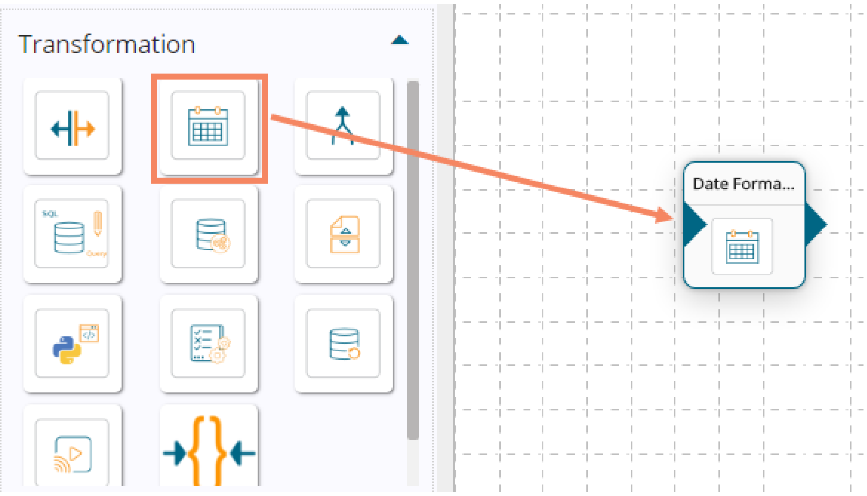

Component properties saved.

xvi) The Split component is ready to read the data coming from the input event, transform the data and gives the output data with the newly created column(s).

8.3.2. Date Formatter Component

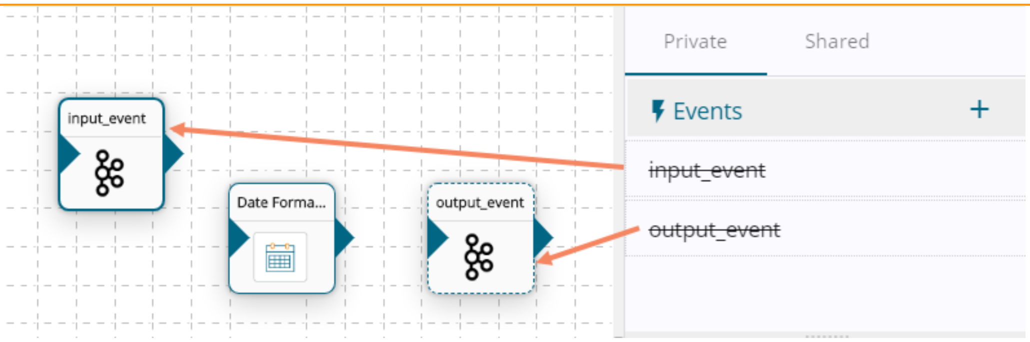
Users can alter the Date format by using this transform component.

i) Drag & drop the Replace Text component to the Workflow Editor.

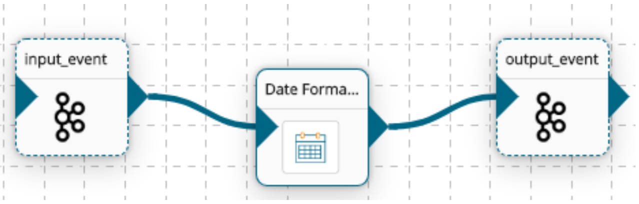


ii) The transformation component requires an input event (to get the data) and sends the data to an output event.

iii) Create two Events and drag them to the Workspace.



iv) Connect the input event and the output event (The data in the input event can come from any Ingestion, Reader or shared events).

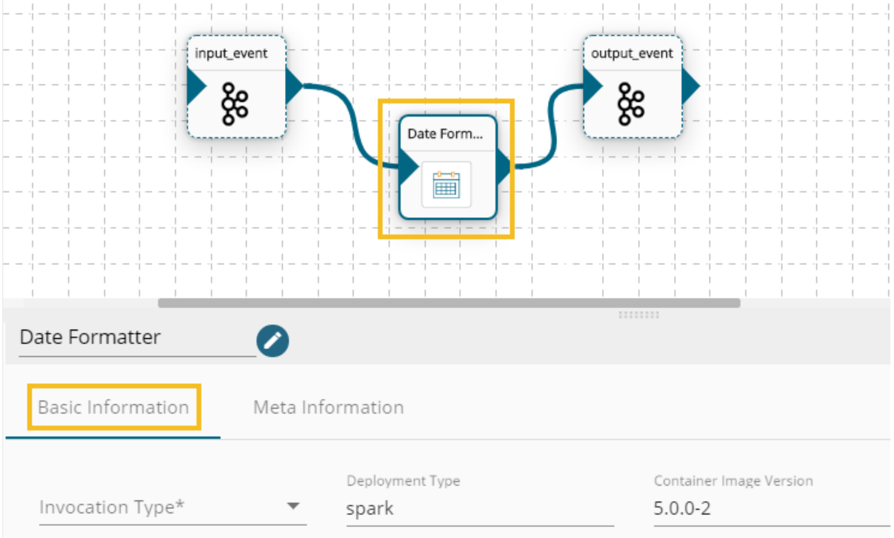


v) **Basic Information**: It is the default tab to open for the Date Formatter while configuring the component.

a. Select an Invocation type from the drop-down menu to confirm the running mode of the reader component. Select '**Real-Time**' or '**Batch**' from the drop-down menu.

b. Deployment Type: It displays the deployment type for the component. This field comes pre-selected.

c. Container Image Version: It displays the image version for the docker container. This field comes pre-selected.



Invocation Type*

Deployment Type: spark

Container Image Version: 5.0.0-2

vi) Open the **Meta Information tab** and provide the connection-specific details.

a. Fill in the Column Name.

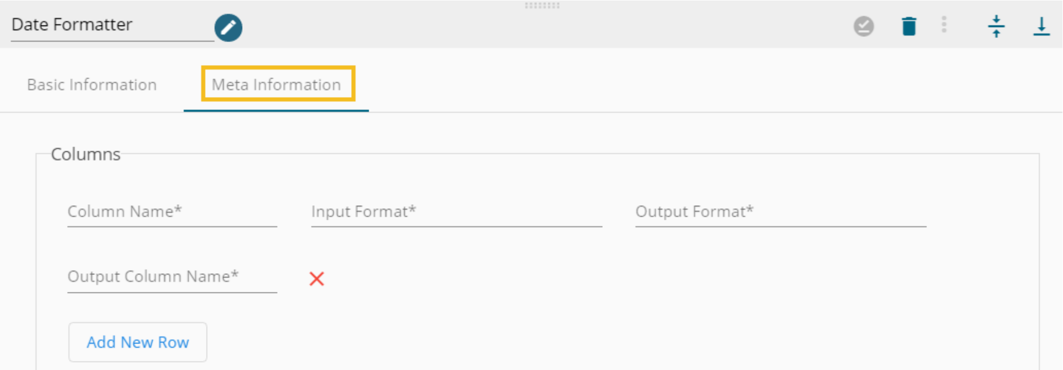
b. Select the input format.

c. Select the output format.

d. Provide a name for the Output Column.

(This component transforms date type for selected input column to provided output data type.)

e. To perform multiple Date Formatter transformations, click the ‘Add New Row’ option and follow the above steps.

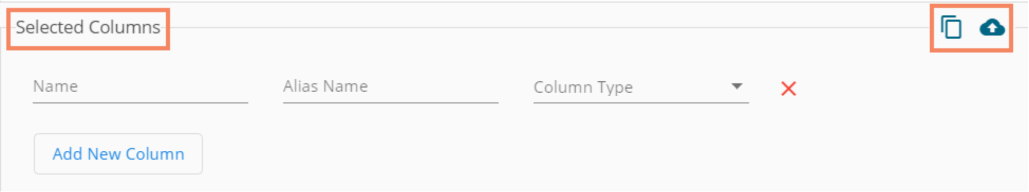


vii) **Selected Columns:** The users can select some specific columns from the table to read data instead of selecting a complete table; this can be achieved via the ‘Selected Columns’ section. Select the columns which you want to read and if you want to change the name of the column, then put that name in alias name section otherwise keep alias name same as of column name and then select a Column Type from the drop-down menu.

or

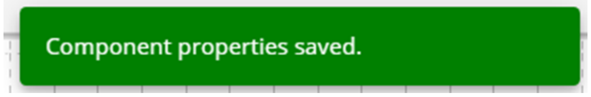
Use ‘Download Data’ and ‘Upload File’ options to select the desired columns.

1. Upload File: The user can upload the existing system files (CSV, JSON) using the ‘Upload File’ icon (file size must be less than 2 MB).
2. Download Data (Schema): Users can download the schema structure in JSON format by using the ‘Download Data’ icon.



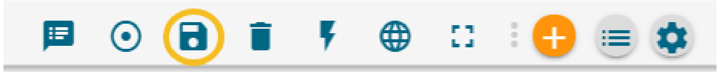
viii) Click the ‘**Save Component in Storage**’ icon to save the configured properties of the Date Formatter component.

ix) A message appears to notify the successful update of the component.

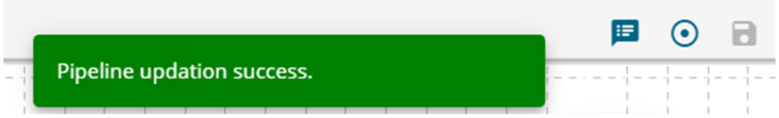


Component properties saved.

x) Click the ‘**Update Pipeline**’ icon.

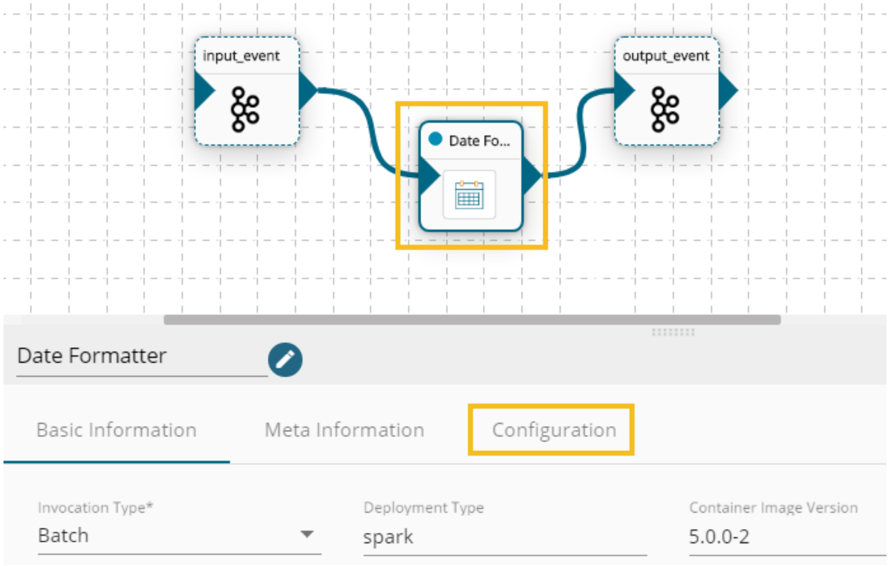


xi) A message appears to confirm the action.



Pipeline updation success.

xii) Click the Dragged Replace Text component again to see the **Configuration tab** below.



Date Formatter

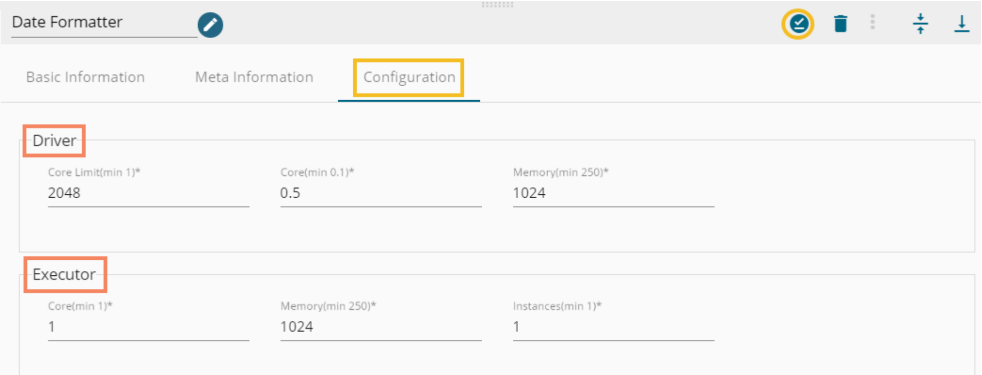
Invocation Type* Batch

Deployment Type spark

Container Image Version 5.0.0-2

xiii) Modify the values given in the Configuration tab.

xiv) Click the '**Save Component in Storage**' icon.



Date Formatter

Driver

Core Limit(min 1)* 2048

Core(min 0.1)* 0.5

Memory(min 250)* 1024

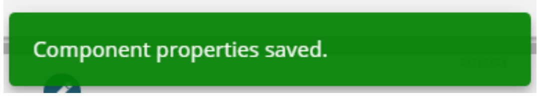
Executor

Core(min 1)* 1

Memory(min 250)* 1024

Instances(min 1)* 1

xv) A message appears to notify the successful update of the component properties.



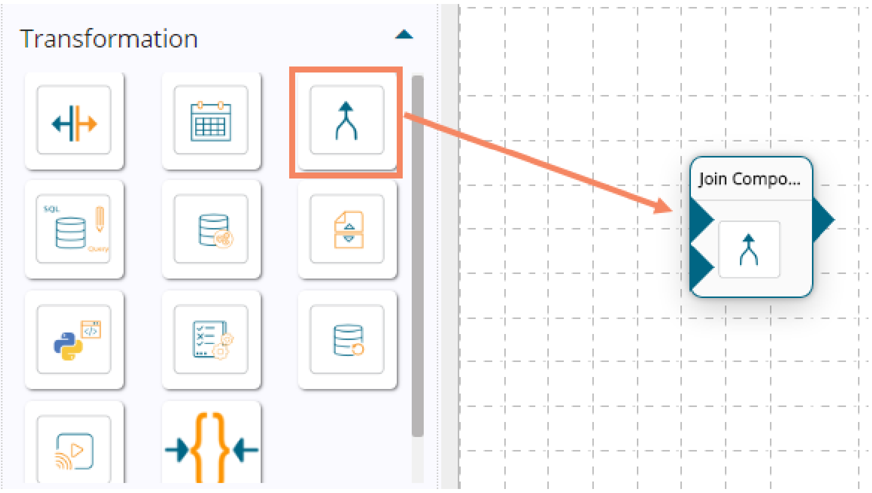
Component properties saved.

xvi) The Date Formatter component is ready to read the data coming from the input event, transform the data and send the newly formatted Dates column(s) to the output event.

8.3.3. Join Component

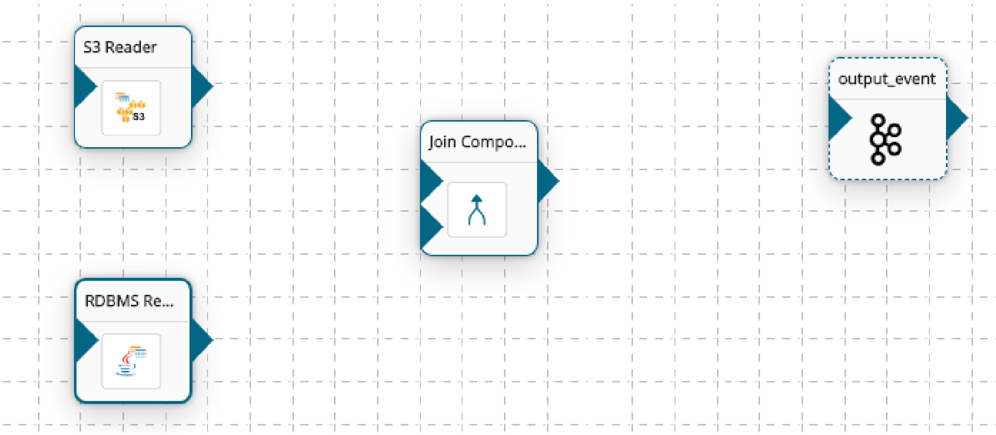
The Join transform allows users to join two or more input data sets as per the user-defined join conditions.

i) Drag and Drop the Join component to the Workflow Editor.

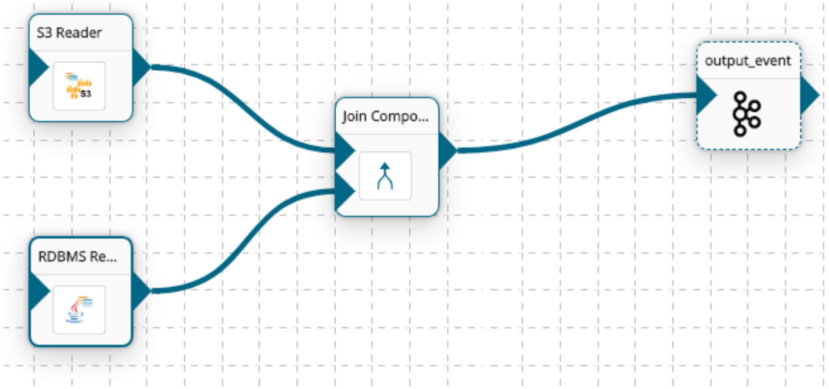


ii) This transformation component requires two input data sources and sends the data to an output event.

iii) Create two input data sources and one output Event and drag them to the workspace.

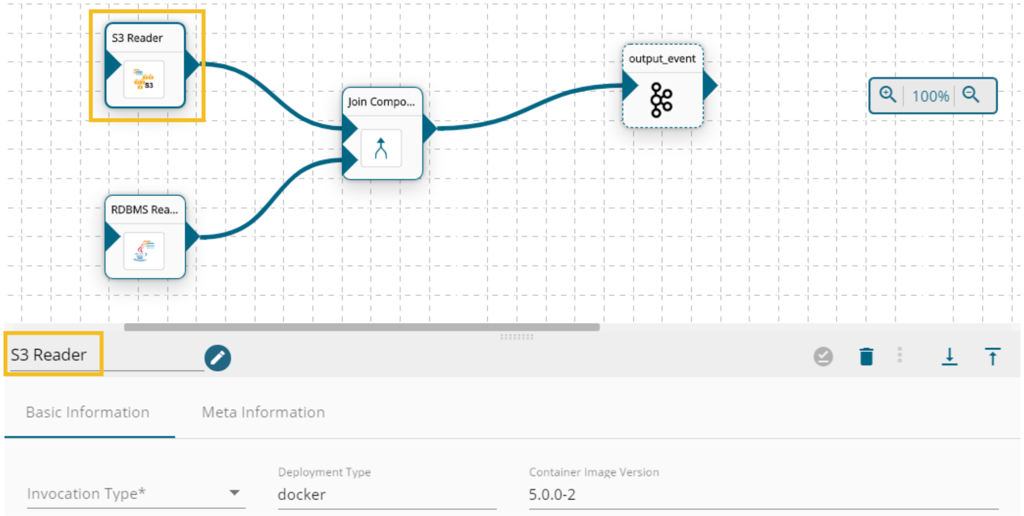


iv) Connect the input data sources and the output event. The data in input data sources can select only from Readers.



v) Configure the Input Data sources.

Note: Refer to the Reader section of the document for more information.



S3 Reader

Basic Information | Meta Information

Invocation Type*

Deployment Type: docker

Container Image Version: 5.0.0-2

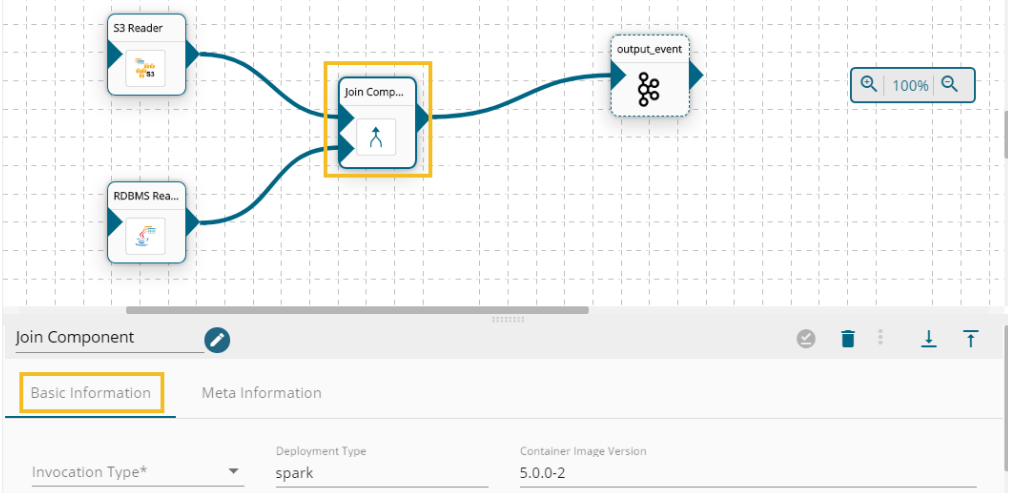
vi) Click the Join component to get the component properties field for the Join Component.

vii) The **Basic Information tab** opens by default.

a. Select an Invocation type from the drop-down menu to confirm the running mode of the reader component. Select ‘**Real-Time**’ or ‘**Batch**’ from the drop-down menu.

b. Deployment Type: It displays the deployment type for the component. This field comes pre-selected.

c. Container Image Version: It displays the image version for the docker container. This field comes pre-selected.



Join Component

Basic Information | Meta Information

Invocation Type*

Deployment Type: spark

Container Image Version: 5.0.0-2

viii) Open the ‘**Meta Information**’ **tab** and provide the connection-specific details.

a. Select the first data source.

b. Select a Join Type from the drop-down menu.

c. Select the Second Data Source.

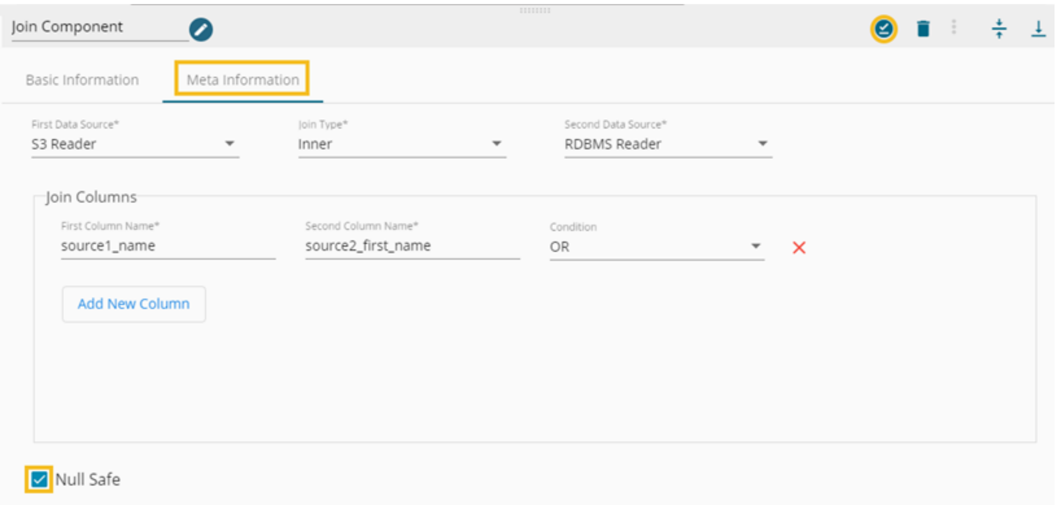
d. Configure the Join Columns- Enter the First Column Name, Second Column Name, and select a Condition from the drop-down menu.

Note: The Condition field can be ignored for the single join.

e. To use multiple joins, click on the ‘**Add New Column**’ option.

f. Use a checkmark in the box beside the ‘**Null Safe**’ option to include the ‘null’ values for join columns.

ix) Click the ‘**Save Component in Storage**’ icon to save the Join component properties.



Join Component

First Data Source*: S3 Reader

Join Type*: Inner

Second Data Source*: RDBMS Reader

Join Columns

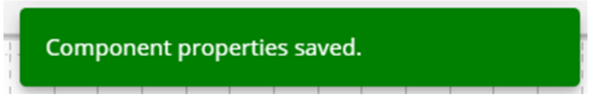
First Column Name*: source1_name

Second Column Name*: source2_first_name

Condition: OR

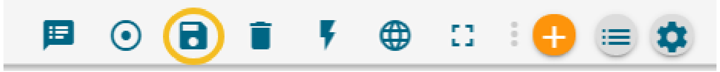
☑ Null Safe

x) A message appears to notify the successful update of the component.

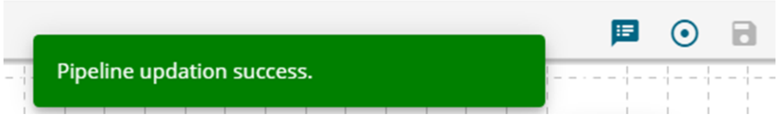


Component properties saved.

xi) Click the ‘**Update Pipeline**’ icon.

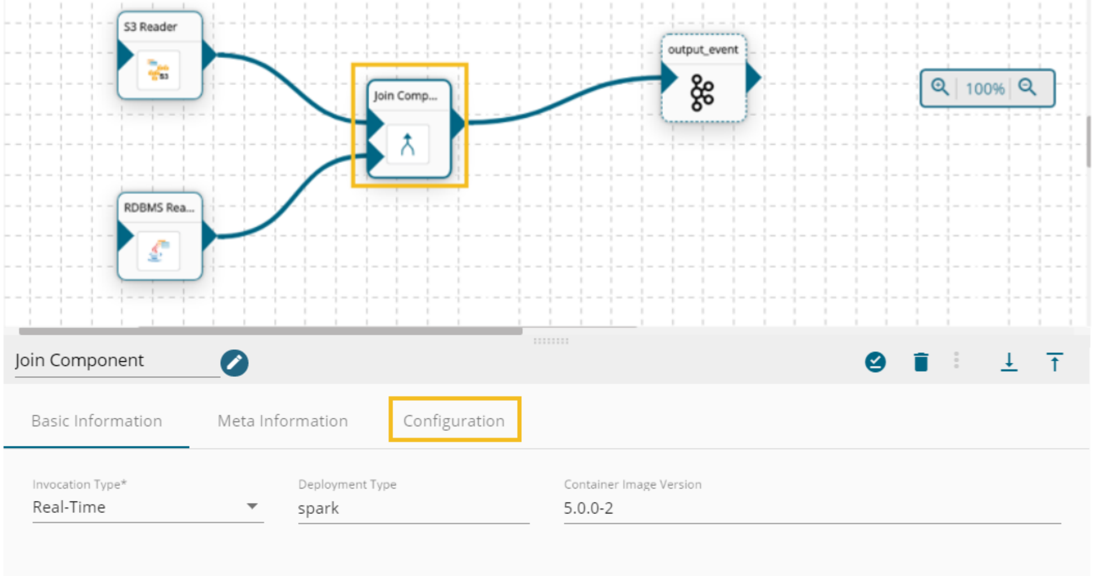


xii) A message appears to confirm the action.



Pipeline updation success.

xiii) Click the Join component again to see the **Configuration tab** below.



Join Component

Basic Information | Meta Information | Configuration

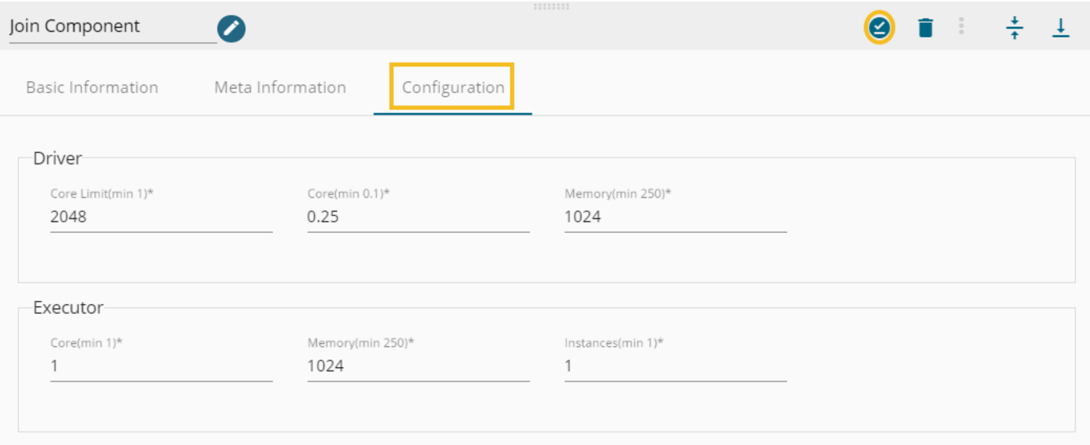
Invocation Type* Real-Time

Deployment Type spark

Container Image Version 5.0.0-2

xiv) Modify the values given in the Configuration tab.

xv) Click the '**Save Component in Storage'** icon.



Join Component

Basic Information | Meta Information | Configuration

Driver

Core Limit(min 1)* 2048

Core(min 0.1)* 0.25

Memory(min 250)* 1024

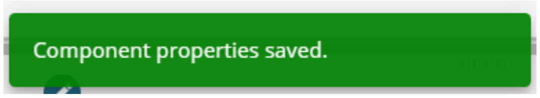
Executor

Core(min 1)* 1

Memory(min 250)* 1024

Instances(min 1)* 1

xvi) A message appears to notify the successful update of the component properties.



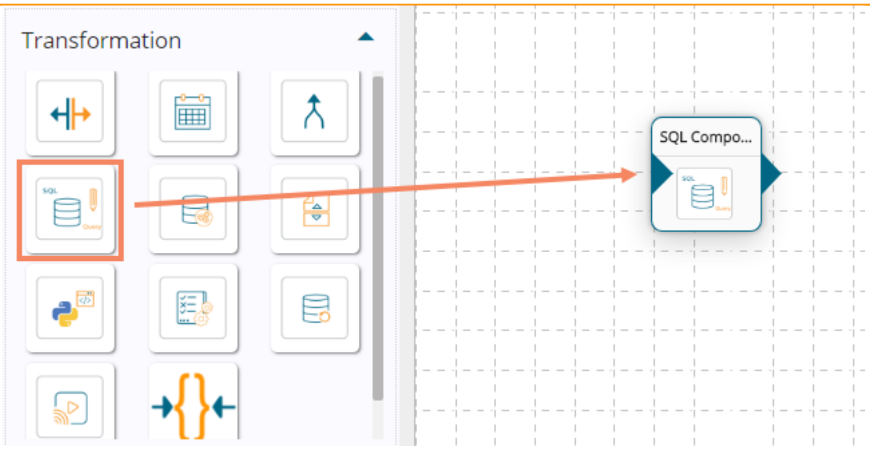
Component properties saved.

xvii) The Join component is ready to read the data coming from the provided inputs, transform the data and send the joined data to the output event.

8.3.4. SQL Component

This component helps users to get data as per the entered query.

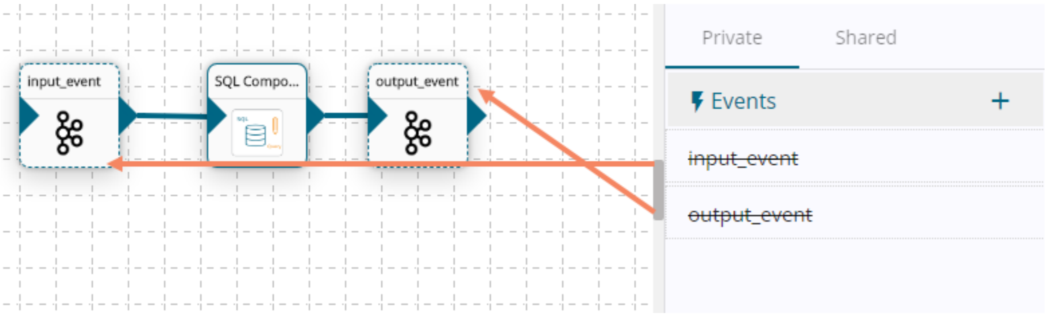
i) Drag and Drop the SQL component to the Workflow Editor.



ii) The transformation component requires an input event (to get the data) and sends the data to an output event.

iii) Create two Events and drag them to the Workspace.

iv) Connect the input event and the output event (The data in the input event can come from any Ingestion, Reader or shared events).



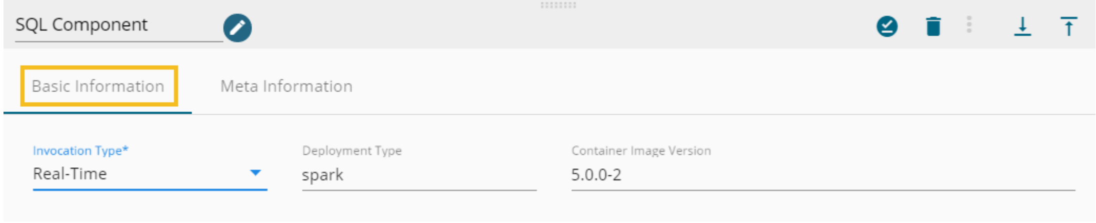
v) Click the SQL component to get the component properties tabs.

vi) The **Basic Information** tab opens by default.

a. Select an Invocation type from the drop-down menu to confirm the running mode of the reader component. Select '**Real-Time**' or '**Batch**' from the drop-down menu.

b. Deployment Type: It displays the deployment type for the component. This field comes pre-selected.

c. Container Image Version: It displays the image version for the docker container. This field comes pre-selected.


SQL Component

Basic Information | Meta Information

Invocation Type*: Real-Time

Deployment Type: spark

Container Image Version: 5.0.0-2

vii) Open the '**Meta Information' tab** and provide the connection-specific details**.**

a. Enter a valid data query to fetch data.

b. Provide the Table Name.

c. Selected Columns- This option helps for schema conversion. Provide a Name, Alias name and Column type for the column from which you wish to select the data.

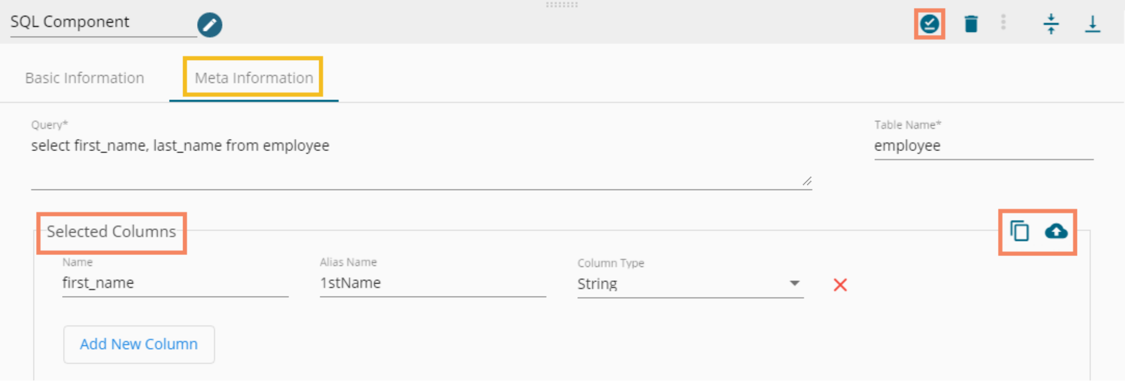
viii) **Selected Columns**: The users can select some specific columns from the table to read data instead of selecting a complete table; this can be achieved via the 'Selected Columns' section. Select the columns which you want to read and if you want to change the name of the column, then put that name in alias name section otherwise keep alias name same as of column name and then select a Column Type from the drop-down menu.

or

Use 'Download Data' and 'Upload File' options to select the desired columns.

1. Upload File: The user can upload the existing system files (CSV, JSON) using the 'Upload File' ☁ icon (file size must be less than 2 MB).
2. Download Data (Schema): Users can download the schema structure in JSON format by using the 'Download Data' ⧉ icon.

ix) Click the '**Save Component in Storage**' ✓ icon to save the component properties.



SQL Component

Basic Information | Meta Information

Query*
select first_name, last_name from employee

Table Name*
employee

Selected Columns

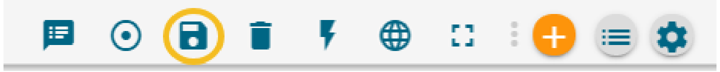
Name	Alias Name	Column Type
first_name	1stName	String

Add New Column

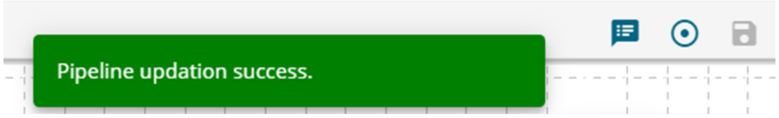
x) A message appears to notify the successful update of the component.

Component properties saved.

xi) Click the '**Update Pipeline**' icon.

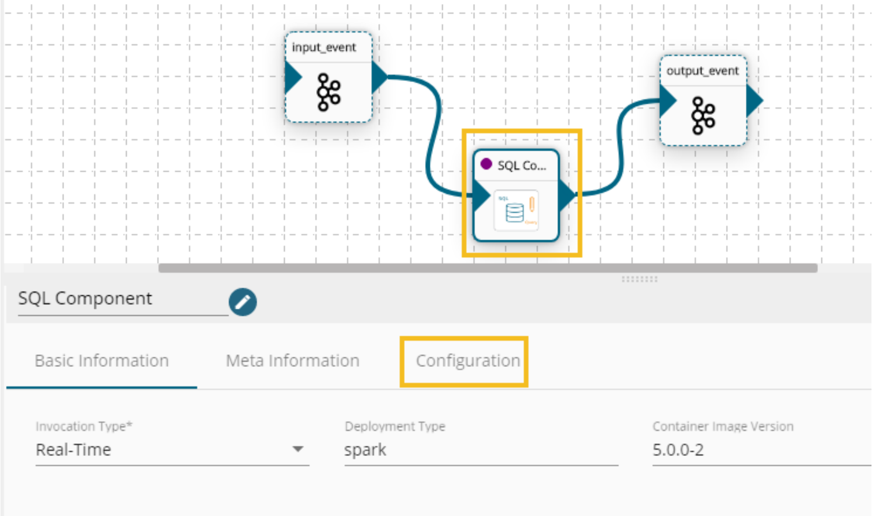


xii) A message appears to confirm the action.



Pipeline updation success.

xiii) Click the Join component again to see the **Configuration tab** below.



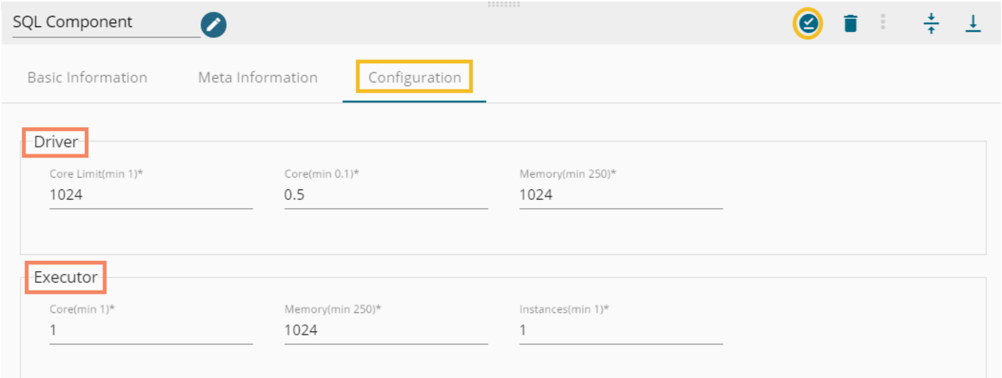
SQL Component

Basic Information | Meta Information | Configuration

Invocation Type*: Real-Time
Deployment Type: spark
Container Image Version: 5.0.0-2

xiv) Modify the values given in the Configuration tab.

xv) Click the ‘**Save Component in Storage**’ icon.



SQL Component

Basic Information | Meta Information | Configuration

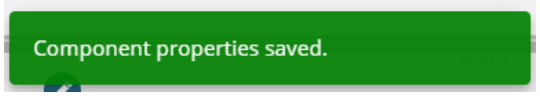
Driver

Core Limit(min 1)*: 1024
Core(min 0.1)*: 0.5
Memory(min 250)*: 1024

Executor

Core(min 1)*: 1
Memory(min 250)*: 1024
Instances(min 1)*: 1

xvi) A message appears to notify the successful update of the component properties.



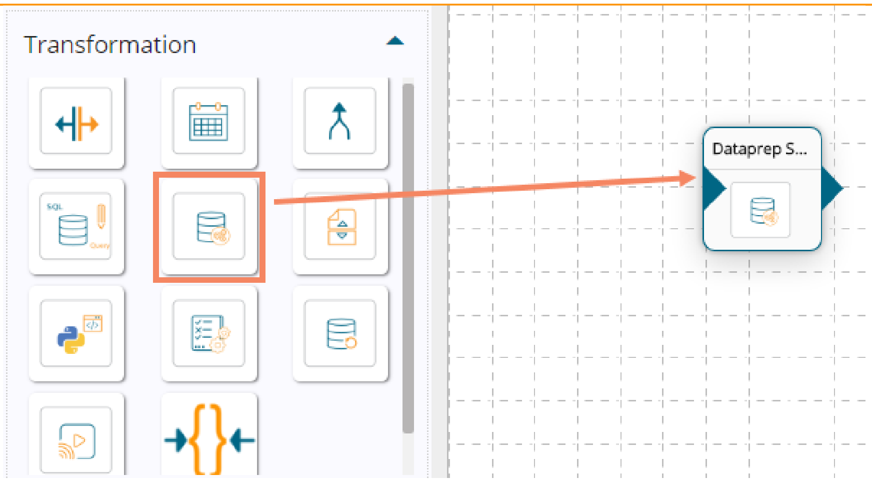
Component properties saved.

xvii) The SQL component gets configured to read the data coming from the input event, transform the data and return it to the output events.

8.3.5. Dataprep Script Runner Component

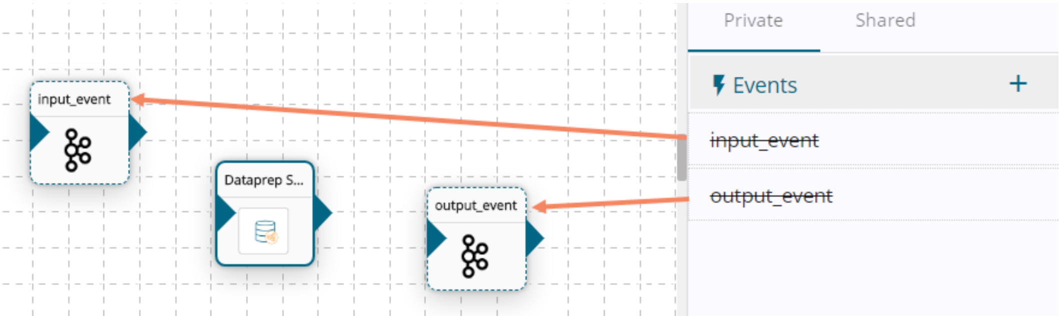
This transform component enables the users to use the exported Dataprep scripts in the Data Pipeline.

i) Drag and Drop Dataprep Script Runner component to the Workflow Editor.

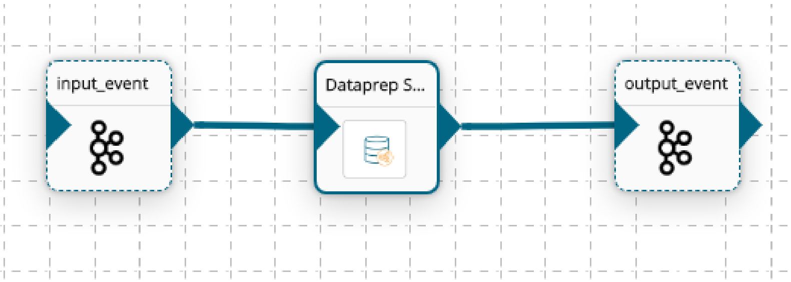


ii) The Transformation component requires an input event (to get the data) and sends the processed data to an output event.

iii) Create two Events and drag them to the Workspace.



iv) Connect the input event and the output event. (The data in input event can come from any Ingestion, Reader or shared events)



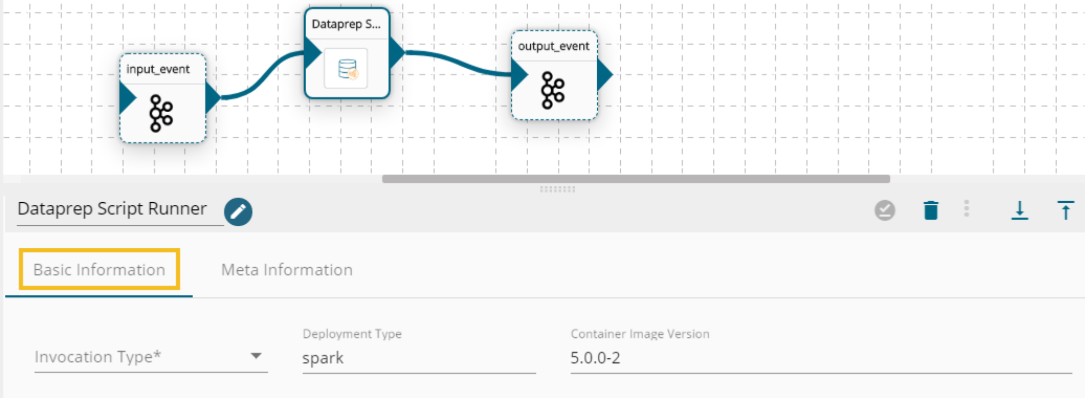
v) Click the Dataprep Script Runner component to get the component properties tabs.

vi) **Basic Information**: It is the default tab to open for the Dataprep Script Runner while configuring the component.

a. Select an Invocation type from the drop-down menu to confirm the running mode of the reader component. Select '**Real-Time**' or '**Batch**' from the drop-down menu.

b. Deployment Type: It displays the deployment type for the component. This field comes pre-selected.

c. Container Image Version: It displays the image version for the docker container. This field comes pre-selected.



Dataprep Script Runner

Basic Information | Meta Information

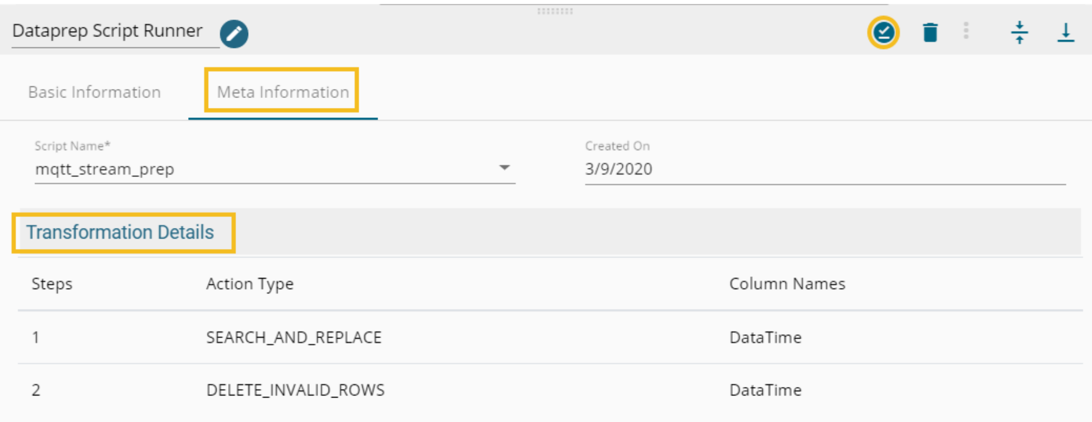
Invocation Type*

Deployment Type: spark

Container Image Version: 5.0.0-2

vii) Select the '**Meta Information' tab** and fill all the required inputs fields.
   a. Script Name: Select a script that is exported from Dataprep Application.
   b. Created On: By selecting a script the date of creation reflects by default.
   c. Transformation Details: The transformation details display below.

viii) Click the '**Save Component in Storage**' icon to save the component properties.


Dataprep Script Runner

Basic Information | Meta Information

Script Name*: mqtt_stream_prep

Created On: 3/9/2020

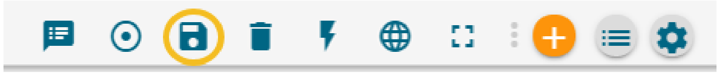
Transformation Details

Steps	Action Type	Column Names
1	SEARCH_AND_REPLACE	DataTime
2	DELETE_INVALID_ROWS	DataTime

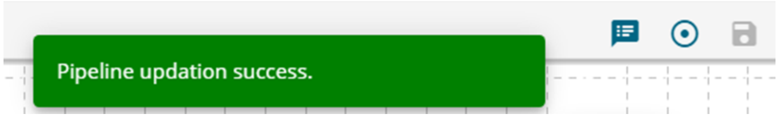
ix) A message appears to notify the successful update of the component.

Component properties saved.

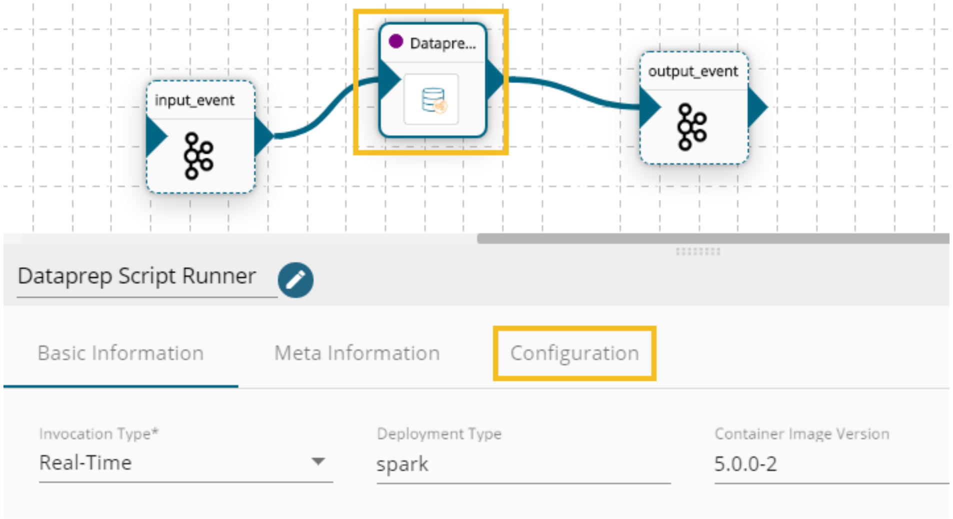
x) Click the '**Update Pipeline**' icon.



xi) A message appears to confirm the action.


Pipeline updation success.

xii) Click the Dataprep Script Runner component again to see the **Configuration tab** below.



Dataprep Script Runner

Basic Information | Meta Information | Configuration

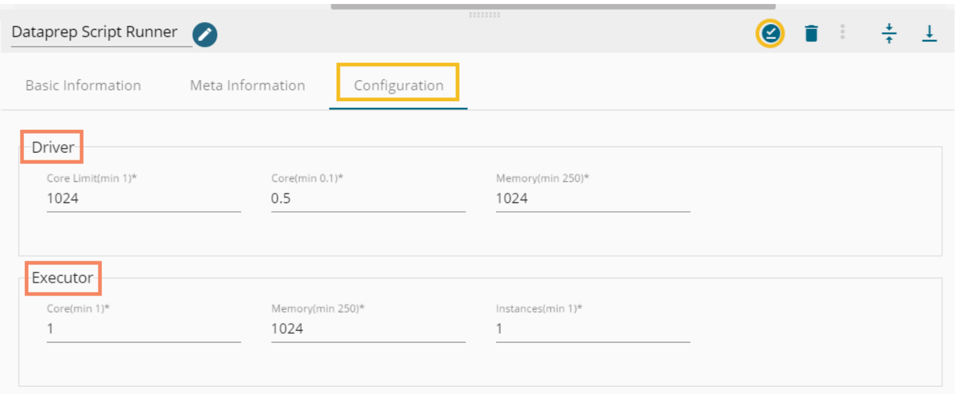
Invocation Type* Real-Time

Deployment Type spark

Container Image Version 5.0.0-2

xiii) Modify the values given in the Configuration tab.

xiv) Click the ‘**Save Component in Storage**’ icon.



Dataprep Script Runner

Basic Information | Meta Information | Configuration

Driver

Core Limit(min 1)* 1024

Core(min 0.1)* 0.5

Memory(min 250)* 1024

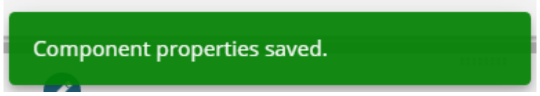
Executor

Core(min 1)* 1

Memory(min 250)* 1024

Instances(min 1)* 1

xv) A message appears to notify the successful update of the component properties.



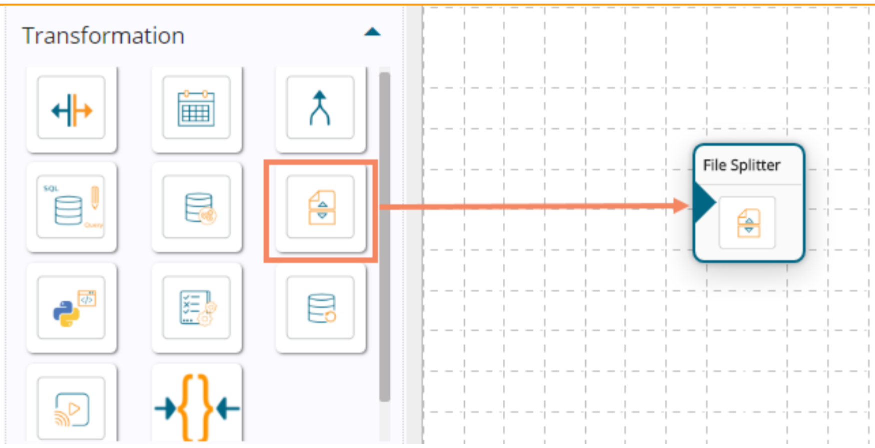
Component properties saved.

xvi) The Dataprep Script Runner component gets configured to read the data coming from the input event, transform the data and return it to the configured output nodes.

8.3.6. File Splitter

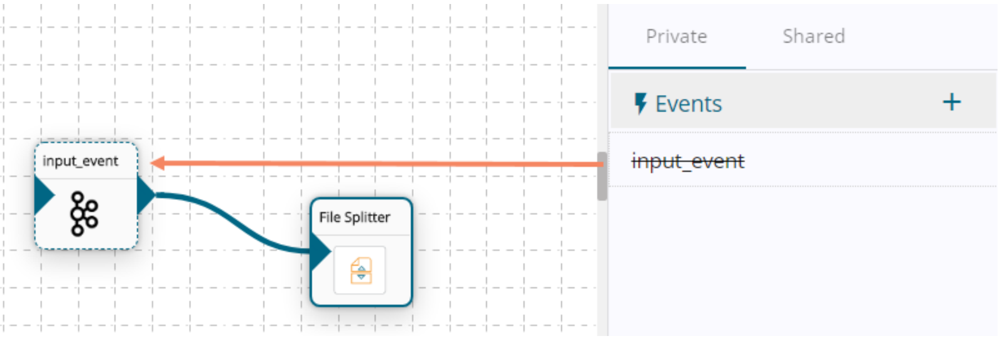
This transform component enables the users to split file.

i) Drag and Drop the File Splitter component to the Workflow Editor.



ii) The Transformation component requires an input event (to get the data). The dragged File Splitter will not have the output connection. The user needs to configure it from the Meta Information tab.

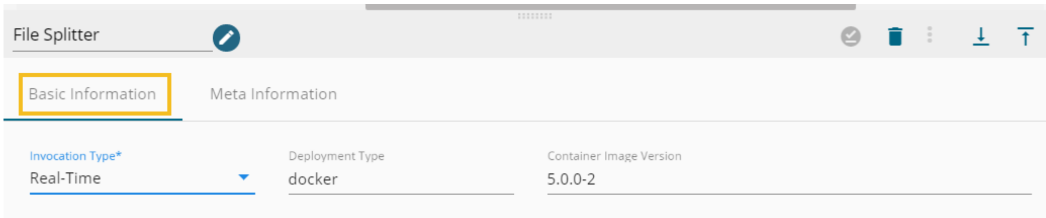
iii) Create one input event, drag it to the Workflow editor and connect it with the File Splitter component.



iv) Click the File Splitter component to get the component properties tabs.

v) **Basic Information**: It is the default tab to open for the File Splitter component while configuring the component.

a. Select an Invocation type from the drop-down menu to confirm the running mode of the reader component. The supported invocation type for this component is Real-time.
b. Deployment Type: It displays the deployment type for the component. This field comes pre-selected.
c. Container Image Version: It displays the image version for the docker container. This field comes pre-selected.



File Splitter

Basic Information | Meta Information

Invocation Type*: Real-Time

Deployment Type: docker

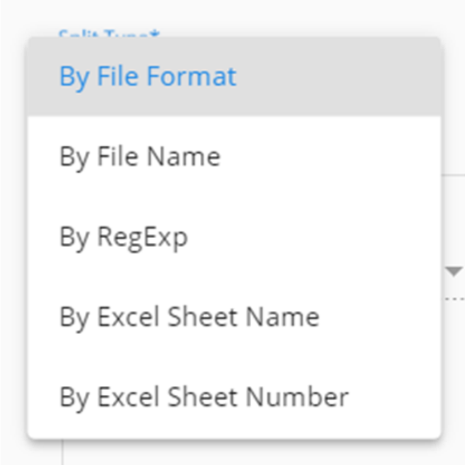
Container Image Version: 5.0.0-2

vi) Select the '**Meta Information' tab** and fill all the required properties fields.

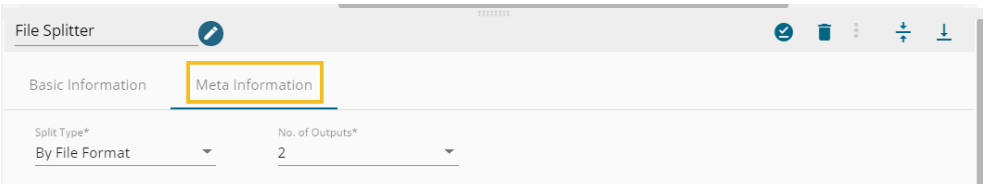
a. Split Type: Select a split type from the drop-down.

The supported Split Types are given below:

1. By File Format
2. By File Name
3. By RegExp
4. By Excel Sheet Name
5. By Excel Sheet Number



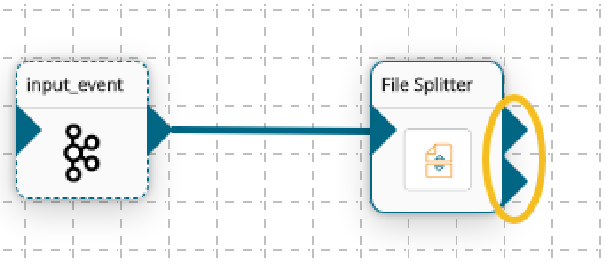
b. No. of Outputs: Select a number from the drop-down menu to create the Output nodes.



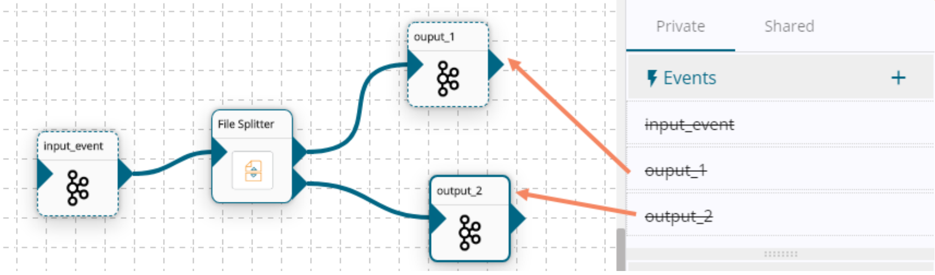
Split Type*: By File Format

No. of Outputs*: 2

vii) By selecting the '**No. of Outputs**' for the File Splitter component the dragged File Splitter component gets those many output nodes.



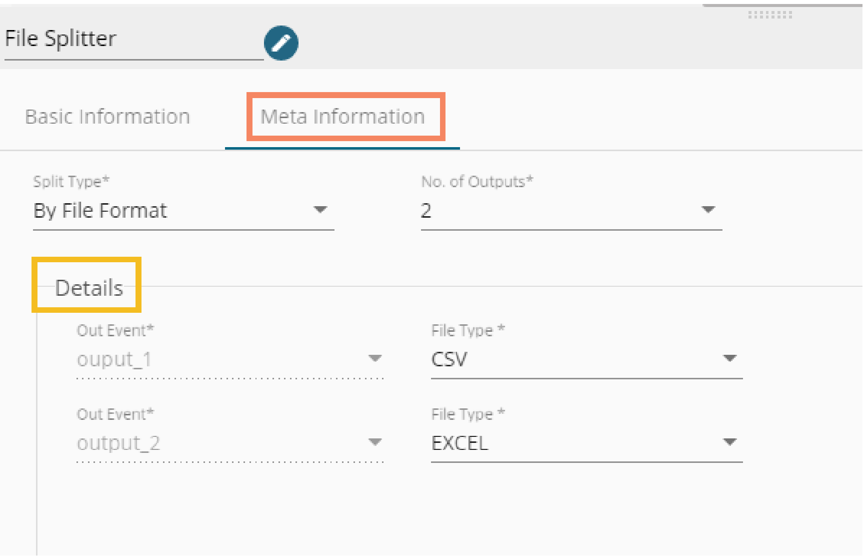
viii) Create and connect the output events to the output nodes of the File Splitter component (E.g., In the present case, 2 output events are created and connected to the File Splitter component).



ix) Open the '**Meta Information**' **tab** and provide the connection-specific details.

x) The Details field displays both the connected output events.

xi) Select File Type for the given output event as displayed in the following image (the supported file types are PDF, CSV, Excel, Others).



File Splitter

Basic Information | Meta Information

Split Type*: By File Format

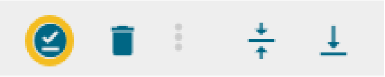
No. of Outputs*: 2

Details

Out Event*: ouput_1 — File Type *: CSV

Out Event*: output_2 — File Type *: EXCEL

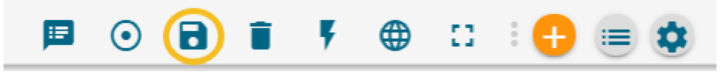
xii) Click the '**Save Component in Storage**' icon to save the component properties.



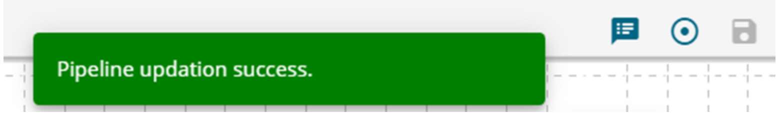
xiii) A message appears to notify the successful update of the component.

Component properties saved.

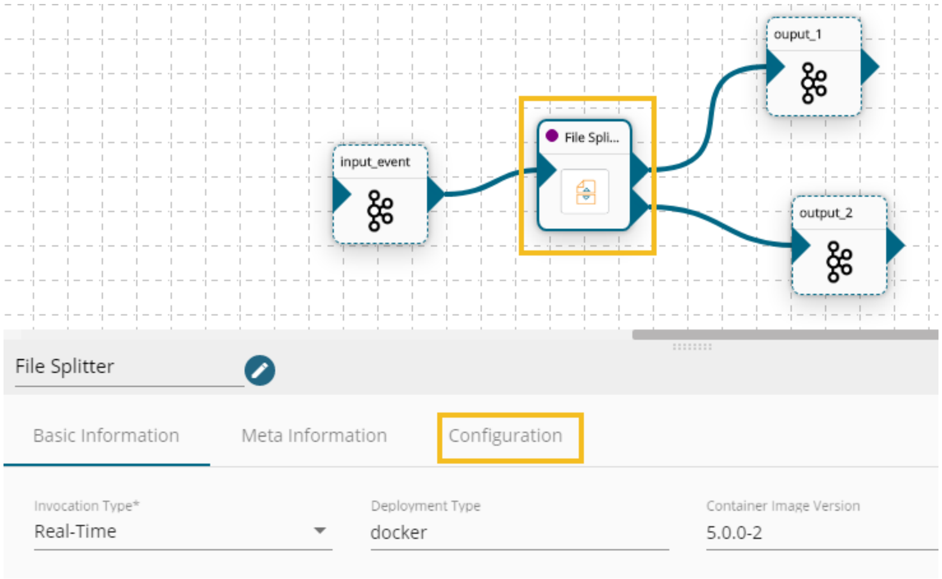
xiv) Click the '**Update Pipeline**' icon.



xv) A message appears to confirm the action.


Pipeline updation success.

xvi) Click the File Splitter component again to see the **Configuration tab** below.



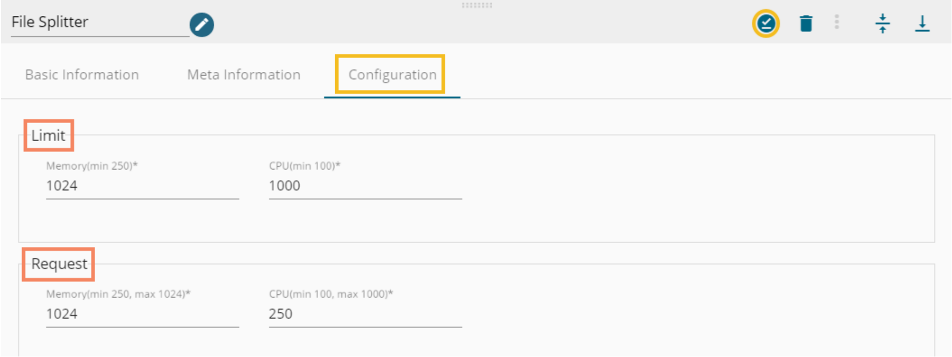
Invocation Type*: Real-Time

Deployment Type: docker

Container Image Version: 5.0.0-2

xvii) Modify the values given in the Configuration tab.

xviii) Click the ‘**Save Component in Storage**’ icon.



Limit

Memory(min 250)*: 1024

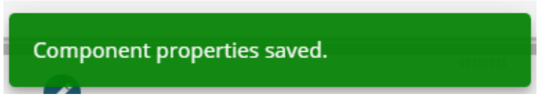
CPU(min 100)*: 1000

Request

Memory(min 250, max 1024)*: 1024

CPU(min 100, max 1000)*: 250

xix) A message appears to notify the successful update of the component properties.

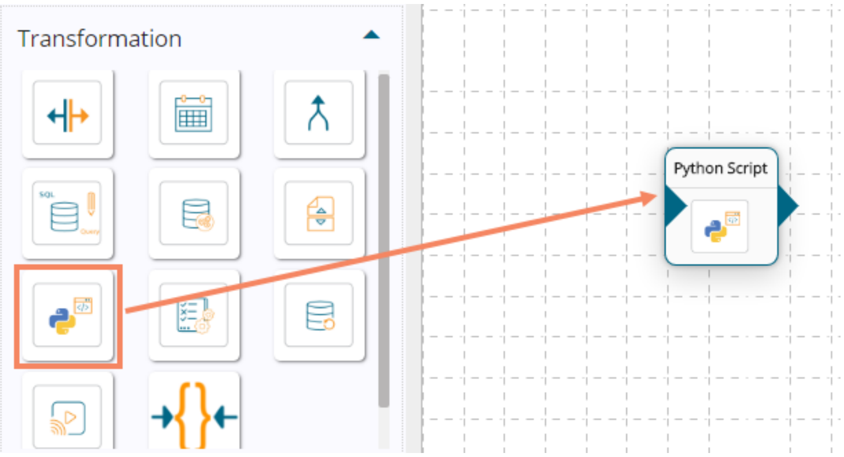

Component properties saved.

xx) The File Splitter component gets configured to read the data coming from the input event, transform the data and return it to the configured output nodes.

8.3.7. Python Script (Custom Python Script)

Python Script component works as a normal python compile.

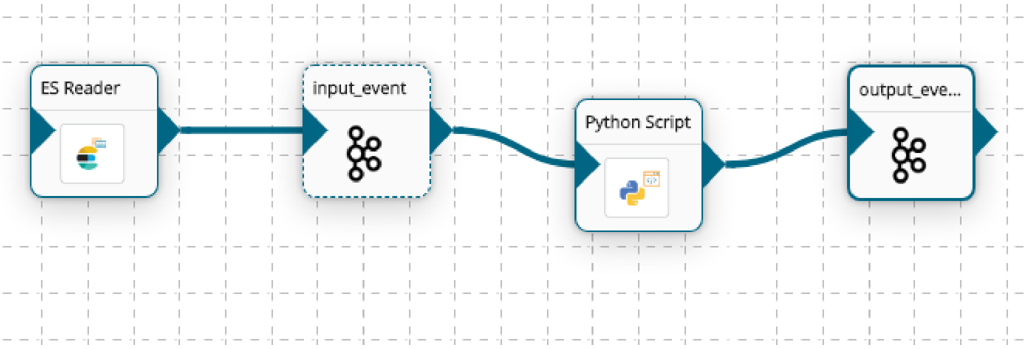
i) Drag and drop the Python Script to the Workflow Editor.



ii) The Python script will read data from an input event and will pass the processed data into an output event, so create and connect two events to the Python Script component.

iii) One reader is needed to pass the data to the input event. (in this case, the ES Reader component is used).

iv) Connect the ES Reader, the Kafka Events, and Python Script components as displayed below:



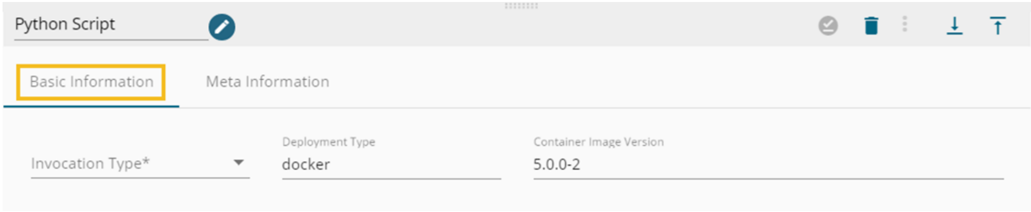
v) Click the dragged Python Script component to get the component properties tabs:

vi) **Basic Information**: It is the default tab to open for the Python Script component while configuring the component.

a. Select an Invocation type from the drop-down menu to confirm the running mode of the reader component. The supported invocation type for this component is Real-time.

b. Deployment Type: It displays the deployment type for the component. This field comes pre-selected.

c. Container Image Version: It displays the image version for the docker container. This field comes pre-selected.



Python Script

Basic Information | Meta Information

Invocation Type*

Deployment Type: docker

Container Image Version: 5.0.0-2

vii) Open the '**Meta Information' tab** to open the fields and configure them.

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.

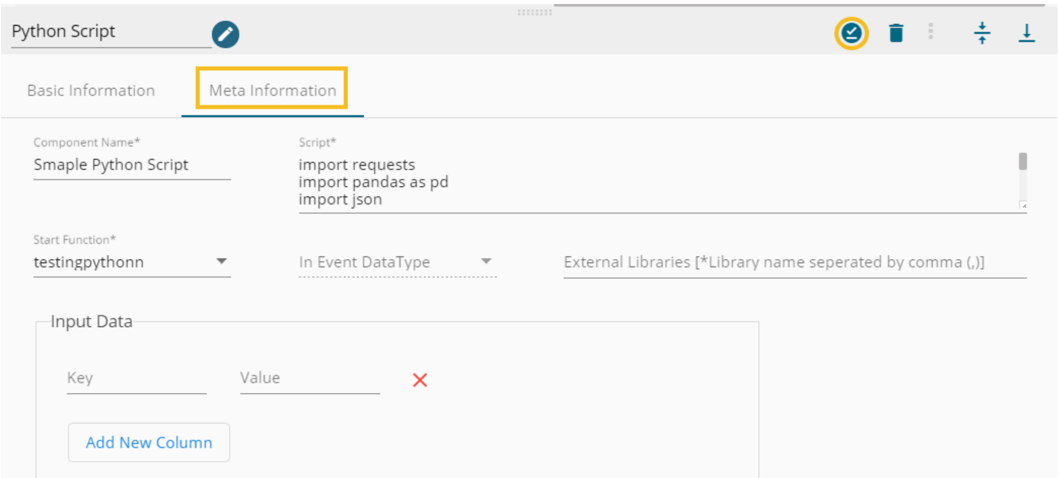
c. Start Function Name: It displays all the function names used in python script in a drop-down menu. Select one function name with which you want to start.

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 names as key and provide the required value.

viii) Click the '**Save Component in Storage'** ✓ icon.



Python Script

Basic Information | Meta Information

Component Name*: Smaple Python Script

Script*:

```
import requests
import pandas as pd
import json
```

Start Function*: testingpythonn

In Event DataType

External Libraries [*Library name seperated by comma (,)]

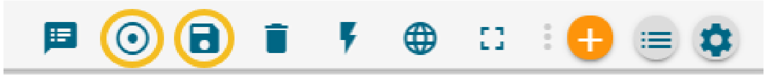
Input Data

Key | Value ✕

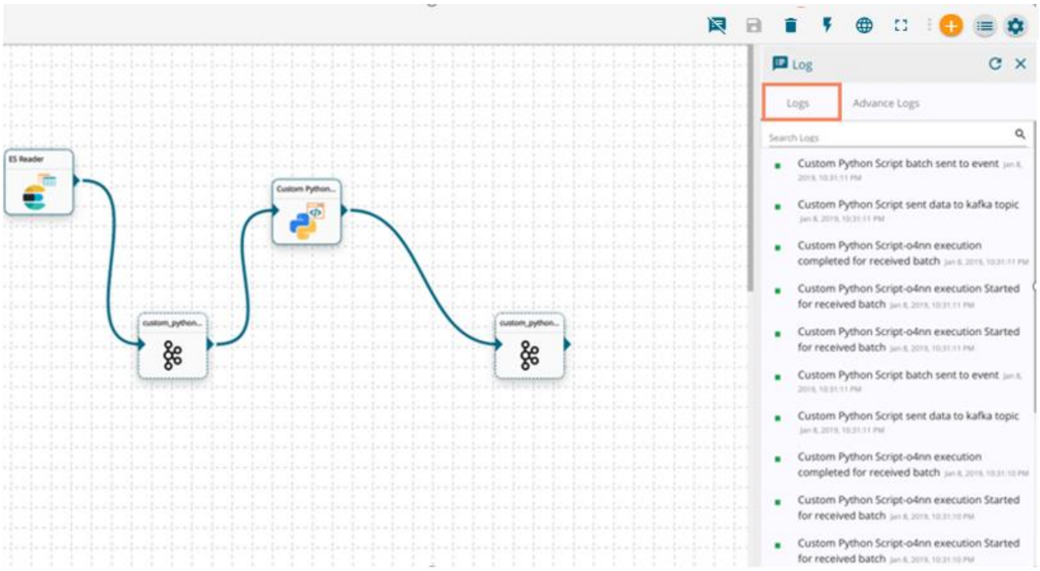
Add New Column

ix) Click the '**Update Pipeline**' icon to save the Pipeline workflow. (After getting the success message)

x) Activate the Pipeline workflow.



xi) Open the Log section to see the logs.



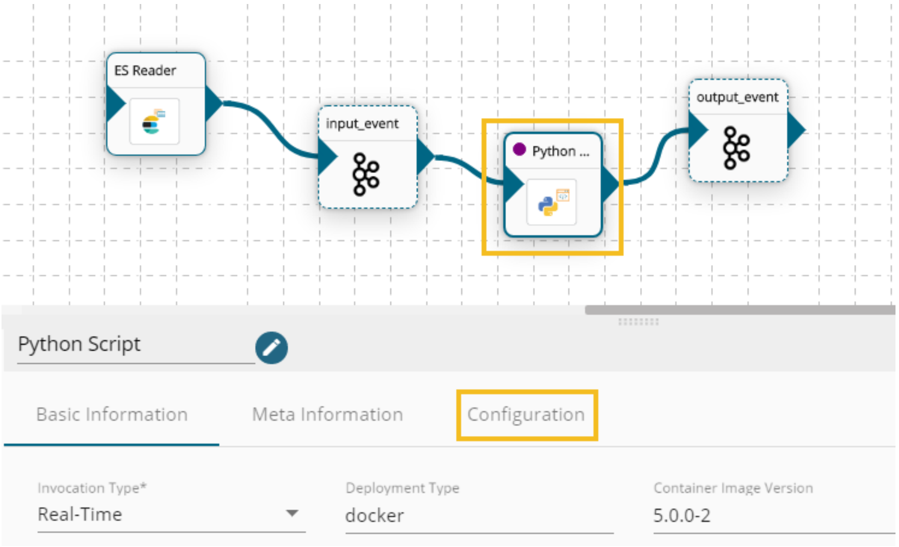
Log

Logs | Advance Logs

Search Logs

```
Custom Python Script batch sent to event Jan 8, 2019, 10:31:11 PM
Custom Python Script sent data to kafka topic Jan 8, 2019, 10:31:11 PM
Custom Python Script-o4nn execution completed for received batch Jan 8, 2019, 10:31:11 PM
Custom Python Script-o4nn execution Started for received batch Jan 8, 2019, 10:31:11 PM
Custom Python Script-o4nn execution Started for received batch Jan 8, 2019, 10:31:11 PM
Custom Python Script batch sent to event Jan 8, 2019, 10:31:11 PM
Custom Python Script sent data to kafka topic Jan 8, 2019, 10:31:11 PM
Custom Python Script-o4nn execution completed for received batch Jan 8, 2019, 10:31:10 PM
Custom Python Script-o4nn execution Started for received batch Jan 8, 2019, 10:31:10 PM
Custom Python Script-o4nn execution Started for received batch Jan 8, 2019, 10:31:10 PM
```

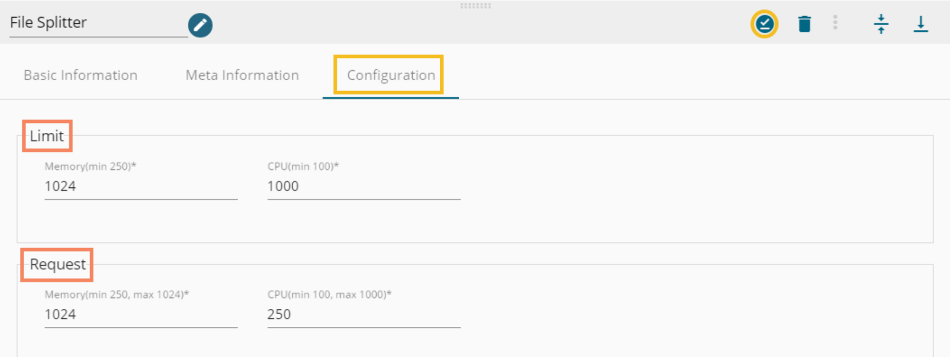
xii) Click the Python Script component again to see the **Configuration tab** below.



Invocation Type*: Real-Time
Deployment Type: docker
Container Image Version: 5.0.0-2

xiii) Modify the values given in the Configuration tab.

xiv) Click the '**Save Component in Storage**' icon.



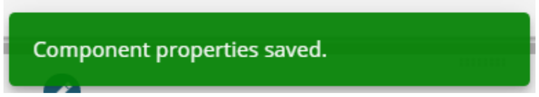
Limit

Memory(min 250)*: 1024
CPU(min 100)*: 1000

Request

Memory(min 250, max 1024)*: 1024
CPU(min 100, max 1000)*: 250

xv) A message appears to notify the successful update of the component properties.



Component properties saved.

xvi) The Python Script component is ready to read the data coming from the input event, it transforms the data and returns output data.

Note:

a. 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 a 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 the input data filed. The 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 a reader, transformation, and writer.

b. 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

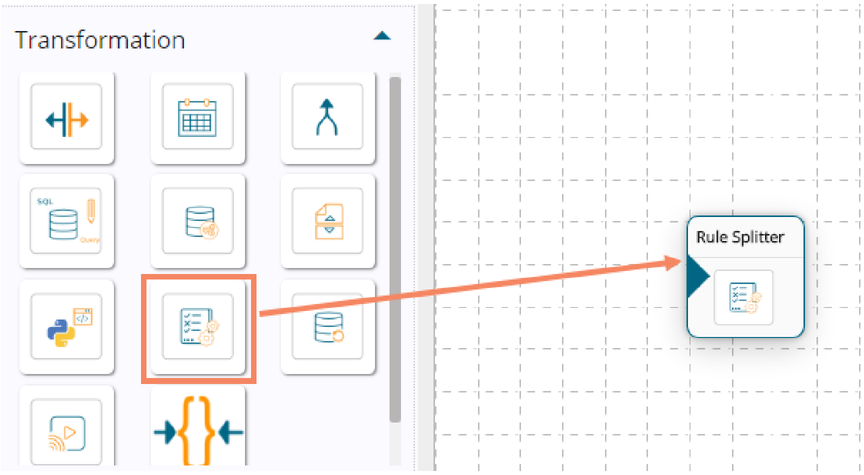
3. Custom Argument with Data frame: If there is custom argument with data frame means Input Event then first argument is always data frame variable. For Example,

i. def getcsvdata(df, range):
ii. cond1 = df['Unit Price'] > range
iii. filter_df = df[cond1]
iv. return filter_df

8.3.8. Rule Splitter

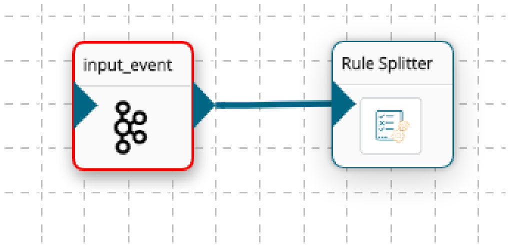
This component in BDB Pipeline helps the users to apply a filter on the data as per the condition and move to separate events.

i) Drag and drop the Rule Splitter component to the Workflow Editor.



ii) The transformation component requires an input event (to get the data) and sends the data to an output event.

iii) Create an input event and connect it to the Rule Splitter component.



iv) Click the Rule Splitter component to get the tabs containing the component properties.

v) The **Basic Information tab** opens by default.

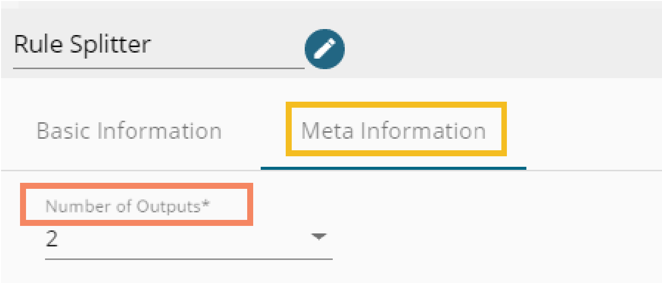
a. Select an Invocation type from the drop-down menu to confirm the running mode of the reader component. Select '**Real-Time**' or '**Batch**' from the drop-down menu.

b. Deployment Type: It displays the deployment type for the component. This field comes pre-selected.

c. Container Image Version: It displays the image version for the docker container. This field comes pre-selected.

vi) Open the **Meta Information tab** and configure the required information:

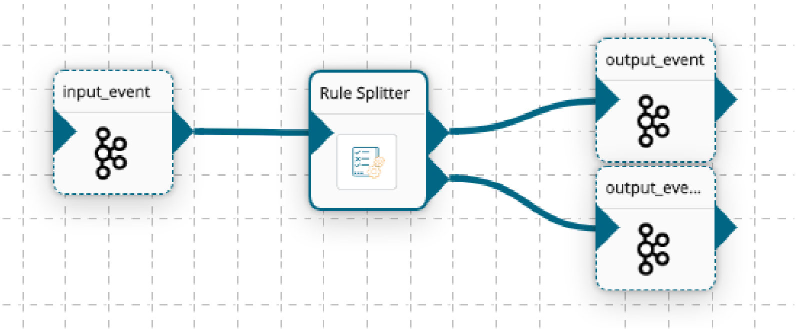
vii) Select the Number of Outputs from the drop-down menu.



Number of Outputs*
2

viii) Based on the No. of Outputs the Rule Splitter component gets the output nodes.

ix) Connect the output events to the Rule Splitter and create a workflow as given below:



x) Open the ‘**Meta Information’ tab** and provide the connection-specific details.

xi) Configure the ‘**Event Relation**’ fields.

i. **Column Name**- Provide a name for the column

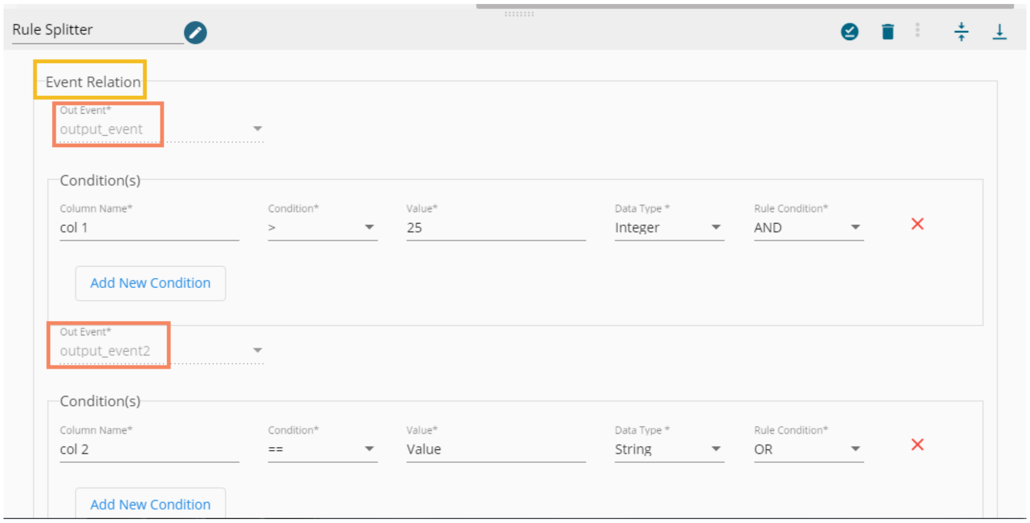
ii. **Condition**- Select a condition from the provided options

iii. **Value**- Enter a defining value for the column

iv. **Data Type** (String/Integer/Double/Date)

v. **Rule Condition** (AND/OR)

xii) Click the ‘**Save Component in Storage**’ ✓ icon after providing the required information.



Rule Splitter

Event Relation

Out Event*
output_event

Condition(s)

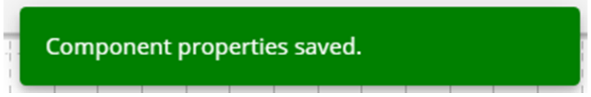
Column Name*	Condition*	Value*	Data Type *	Rule Condition*
col 1	>	25	Integer	AND

Out Event*
output_event2

Condition(s)

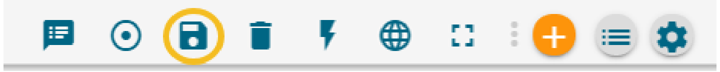
Column Name*	Condition*	Value*	Data Type *	Rule Condition*
col 2	==	Value	String	OR

xiii) A message appears to notify the successful update of the component.

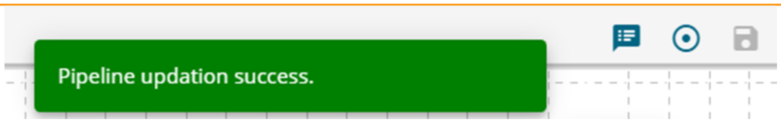


Component properties saved.

xiv) Click the ‘**Update Pipeline**’ icon.

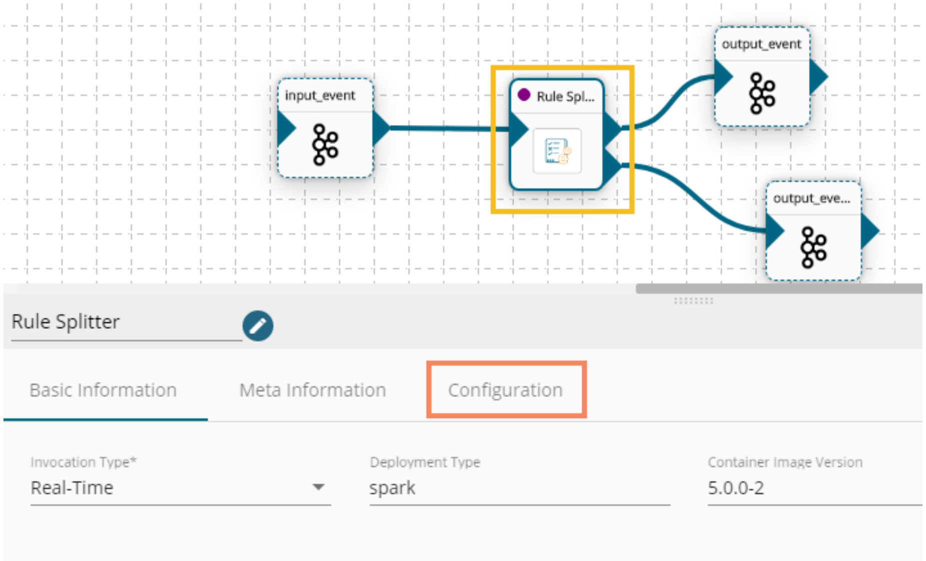


xv) A message appears to confirm the action.



Pipeline updation success.

xvi) Click the Rule Splitter component again to see the **Configuration tab** below.



Rule Splitter

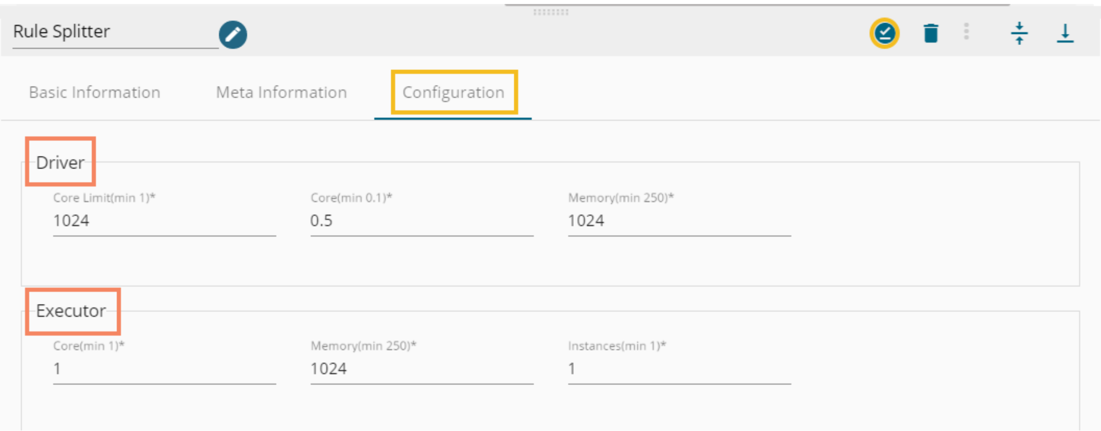
Invocation Type*: Real-Time

Deployment Type: spark

Container Image Version: 5.0.0-2

xvii) Modify the values given in the Configuration tab.

xviii) Click the '**Save Component in Storage**' icon.



Rule Splitter

Driver

Core Limit(min 1)*: 1024

Core(min 0.1)*: 0.5

Memory(min 250)*: 1024

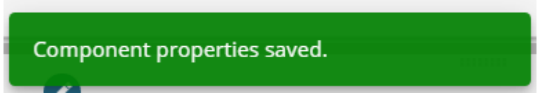
Executor

Core(min 1)*: 1

Memory(min 250)*: 1024

Instances(min 1)*: 1

xix) A message appears to notify the successful update of the component properties.



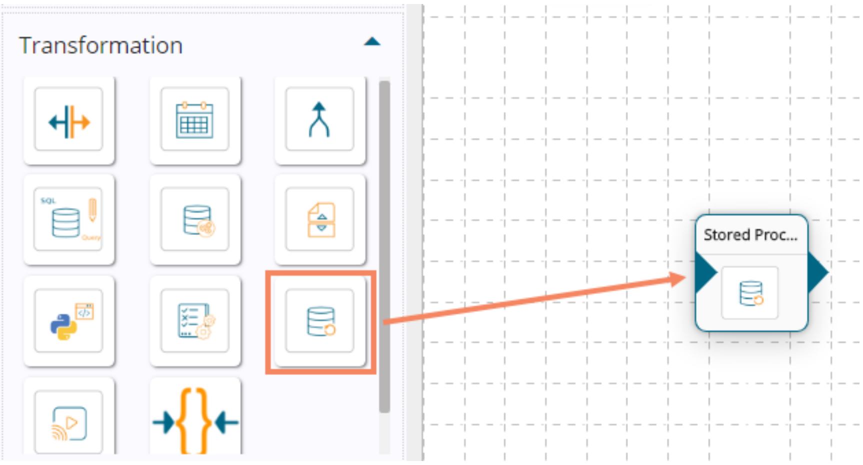
Component properties saved.

xx) The Rule Splitter component is ready to read the data coming from the input event, transform the data and return output data (based on the configured output nodes).

8.3.9. Stored Procedure Runner

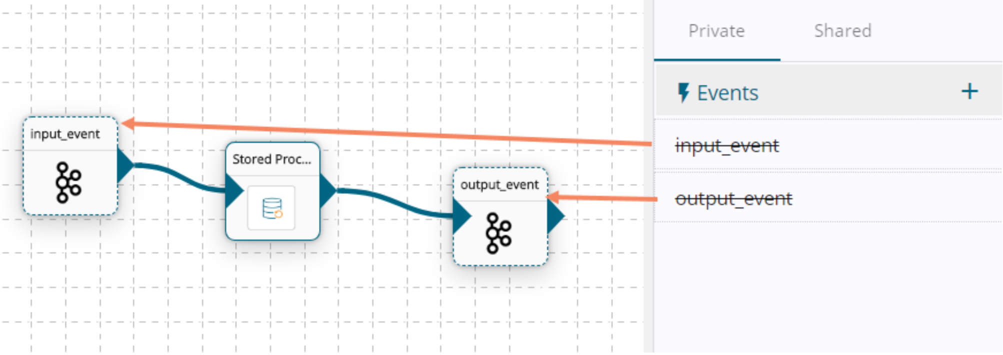
Stored Procedure Runner Component can be used to run a stored procedure using a Pipeline. It is a **Batch** component. As soon as the data comes from the previous components the Stored Procedure Runner gets triggered and performs its task only once for that batch. This Transformation component cannot be used with Stream data, because the velocity of data in case of streaming is very high.

i) Drag and drop the Stored Procedure Runner component to the Workflow Editor.



ii) The transformation component requires an input event (to get the data) and sends the data to an output event.

iii) Create one input event and an output event connect them to the Stored Producer Runner component.



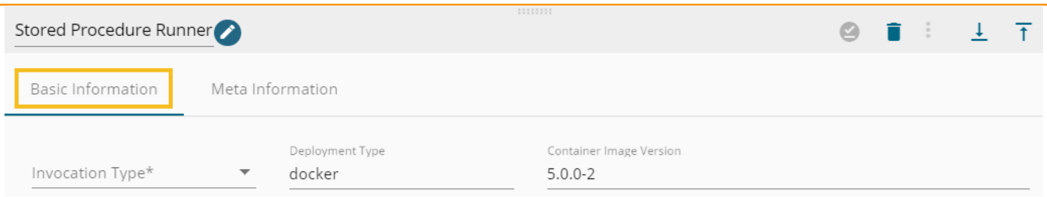
iv) Click the Stored Procedure Runner component to get the tabs containing the component properties.

v) The **Basic Information tab** opens by default.

   a. Select an Invocation type from the drop-down menu to confirm the running mode of the reader component (Only the '**Batch**' invocation type is supported).

   b. Deployment Type: It displays the deployment type for the component. This field comes pre-selected.

   c. Container Image Version: It displays the image version for the docker container. This field comes pre-selected.



Stored Procedure Runner

Basic Information | Meta Information

Invocation Type*

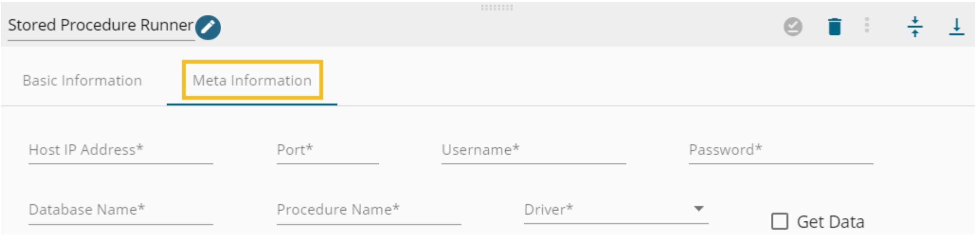
Deployment Type: docker

Container Image Version: 5.0.0-2

vi) Open the '**Meta Information' tab** and provide the connection-specific details.

vii) Configure the mandatory fields by providing the required information:

a. Host (JDBC Host)
b. Port
c. Username
d. Password
e. Database Name
f. Procedure Name (Name of the Stored Procedure)
g. Driver: Select a driver using the drop-down menu
h. Get Data: Enable this option using a checkmark to get returned data from Stored Procedure



Stored Procedure Runner

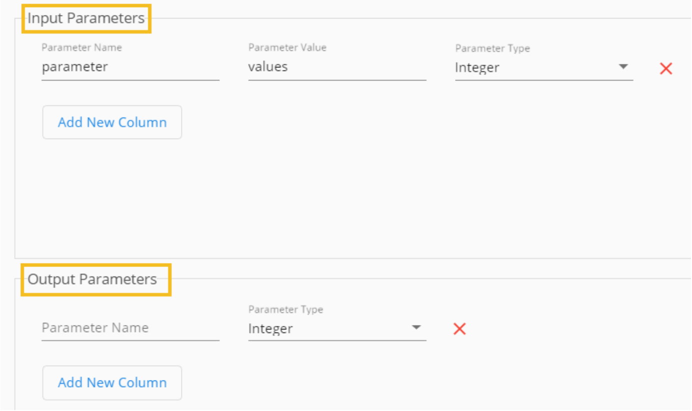
Basic Information | Meta Information

Host IP Address* | Port* | Username* | Password*

Database Name* | Procedure Name* | Driver* | ☐ Get Data

viii) Configure the Stored Procedure specific Input and Output parameters:

a. Input Parameters
   i. Parameter Name
   ii. Parameter Value
   iii. Parameter Type
b. Output Parameters
   i. Parameter Name
   ii. Parameter Type



Input Parameters

Parameter Name	Parameter Value	Parameter Type
parameter	values	Integer

Add New Column

Output Parameters

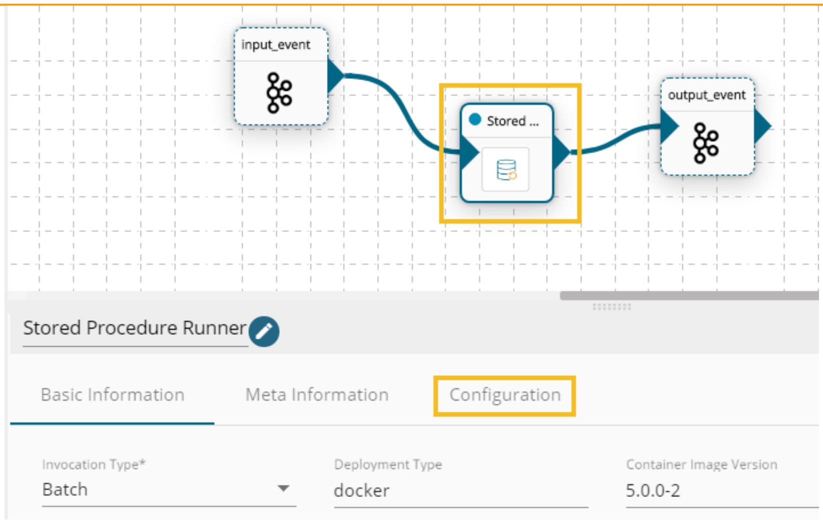
Parameter Name	Parameter Type
	Integer

Add New Column

ix) Click the '**Save Component in Storage**' icon.

x) Click the '**Update Pipeline**' icon after getting the notification that the component properties are saved.

xi) Click the Stored Procedure Runner component again to see the **Configuration tab** below.



Stored Procedure Runner

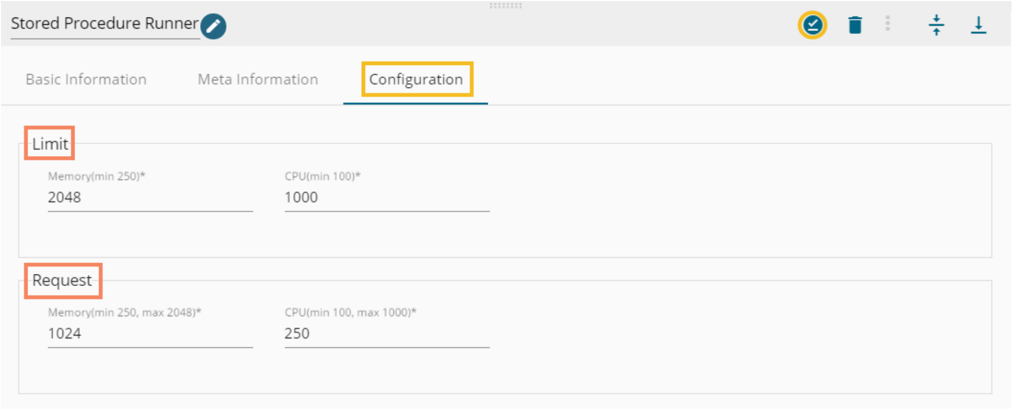
Invocation Type*: Batch

Deployment Type: docker

Container Image Version: 5.0.0-2

xii) Modify the values given in the Configuration tab.

xiii) Click the '**Save Component in Storage**' ✓ icon.


Stored Procedure Runner

Limit

Memory(min 250)*: 2048

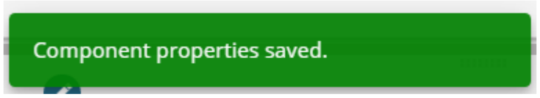
CPU(min 100)*: 1000

Request

Memory(min 250, max 2048)*: 1024

CPU(min 100, max 1000)*: 250

xiv) A message appears to notify the successful update of the component properties.


Component properties saved.

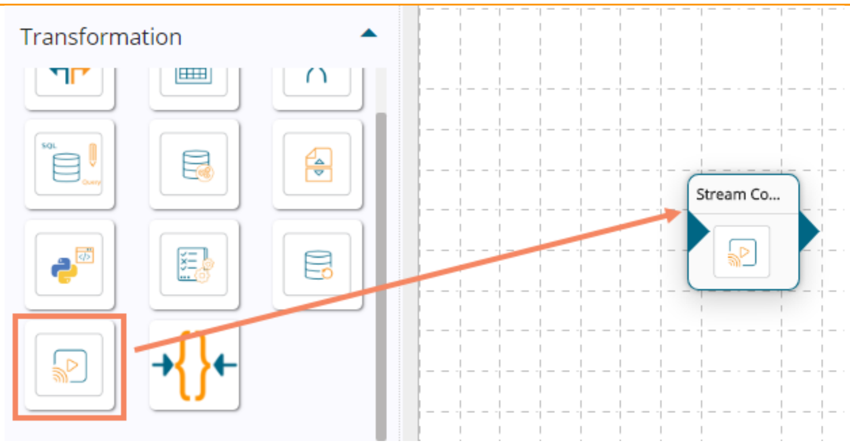
xv) The Stored Procedure Runner component is ready to read the data coming from the input event, transform the data and return output data to the output events.

Note: Data from an Input Event cannot be taken as Input Parameter for the Stored Procedure Runner component.

8.3.10. Stream Component

The steps to configure the Stream component is described in this section.

i) Drag and drop the Stream Component to the Pipeline Workflow Editor.



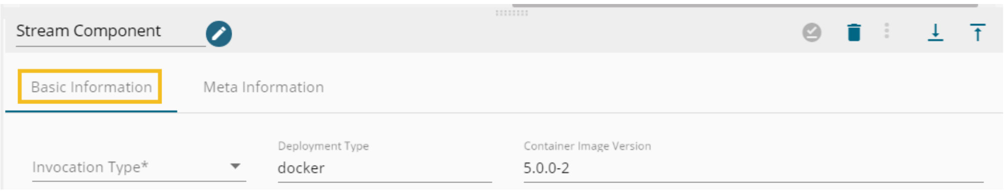
ii) Click the dragged Stream component to get the Configuration fields:

iii) The **Basic Information tab** opens by default.

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

b. Deployment Type: It comes preselected based on the component.

c. Container Image Version: It comes preselected based on the component.



Invocation Type*

Deployment Type: docker

Container Image Version: 5.0.0-2

iv) Open the **Meta Information tab** and provide the connection-specific details.

a. Input Fields: User needs to provide all/selected columns that IoT device is producing along with their Data Type.

1. Field Name: Provide a name for the field.
2. Filed Type: Select a field type from the drop-down menu (The supported field types are String, Long, Integer, Float, Double, Date).

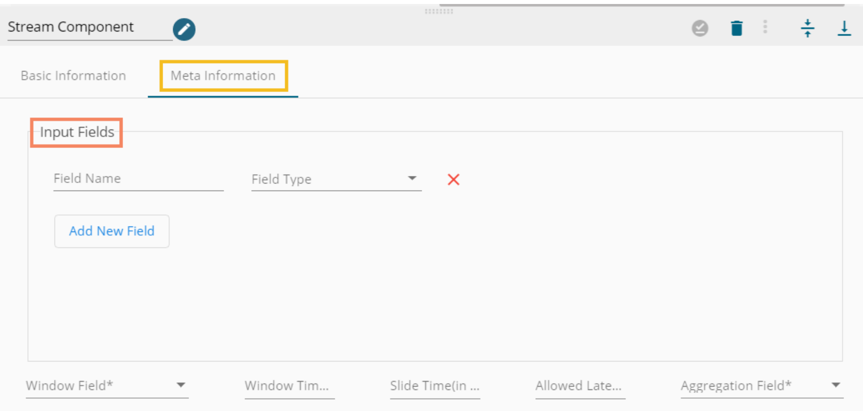
b. Window Field: The user needs to provide the field using which we want to do bucketing of data. This field must be a timestamp value.

c. Window Time: Time duration of Window (Bucket).

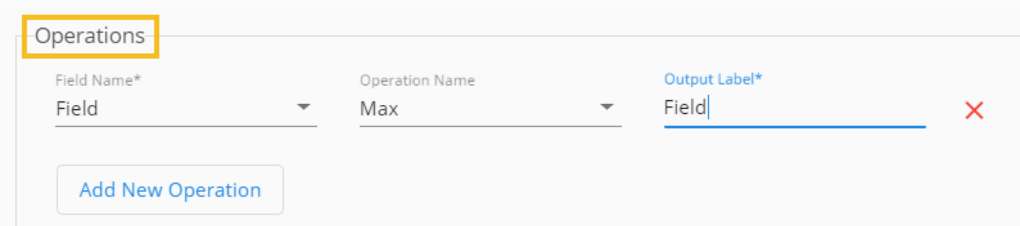
d. Allowed Lateness: Allowed time up to which the component accepts late data.

e. Aggregation Field: User needs to provide field using which component performs aggregation operations on operation fields.

f. Operations: The user needs to select the field name, operation name, and provide the output field label. Multiple operations are also supported.



g. Operations: Configure the following details.
   1. Field Name: The configured input fields appear in the drop-down. Select a field from the drop-down menu.
   2. Operation Name: Select an option to select the Operation name.
   3. Output Label: Provide an output label.



Operations

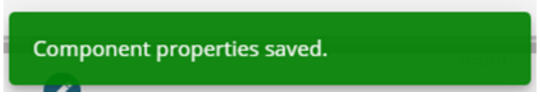
Field Name*: Field

Operation Name: Max

Output Label*: Field

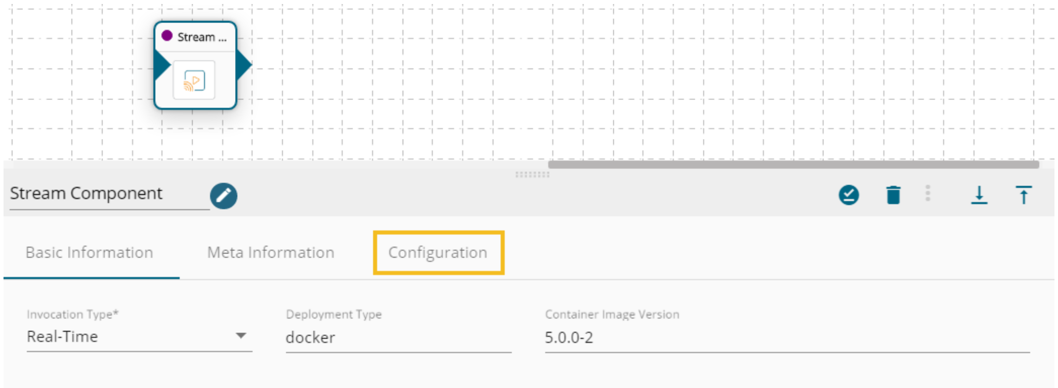
v) Click the '**Save Component in Storage**' ✓ icon.

vi) Update the Pipeline by clicking the '**Update Pipeline**' icon.



Component properties saved.

vii) Open the Stream Component again to see the **Configuration tab** below.



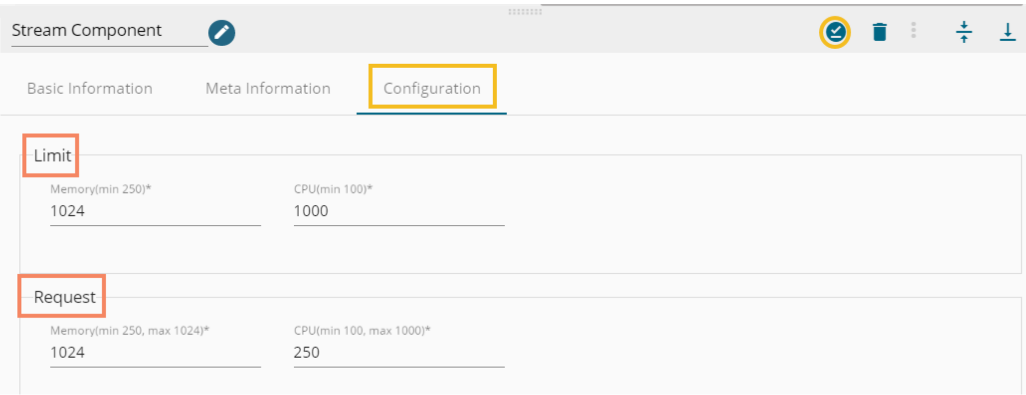
Invocation Type*: Real-Time

Deployment Type: docker

Container Image Version: 5.0.0-2

viii) Modify the values for the Configuration tab.

ix) Click the '**Save Component in Storage**' ✓ icon.



Limit

Memory(min 250)*: 1024

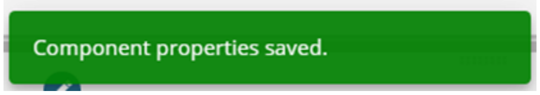
CPU(min 100)*: 1000

Request

Memory(min 250, max 1024)*: 1024

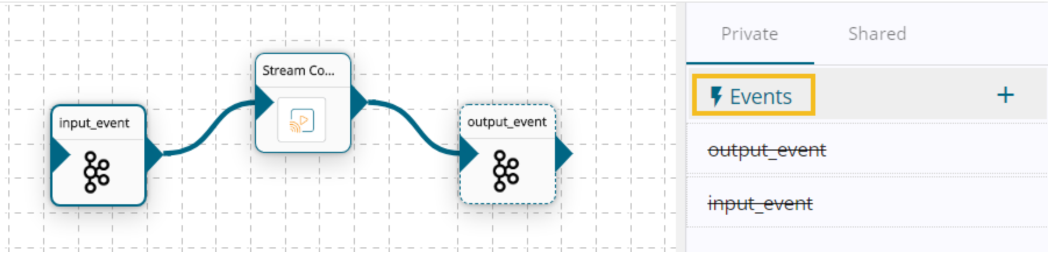
CPU(min 100, max 1000)*: 250

x) A message appears to notify the action.



Component properties saved.

xi) The Stream Component is ready to be used in the Pipeline workflow. It can read data from an input event and pass the processed streaming data to an output event, so accordingly connect the required event components to create a pipeline workflow.



Note:

a. The user can add new field and new operation by clicking the '**Add New Field'** and '**Add New Operation**' options respectively.

b. Windowing: Windowing is an operation in spark structured streaming where the stream of data gets divided into a small amount of window and then performs aggregation operations on the data.

For detailed information, please visit this link:

https://spark.apache.org/docs/2.3.0/structured-streaming-programming-guide.html#window

operations-on-event-time

8.3.11. Flatten JSON

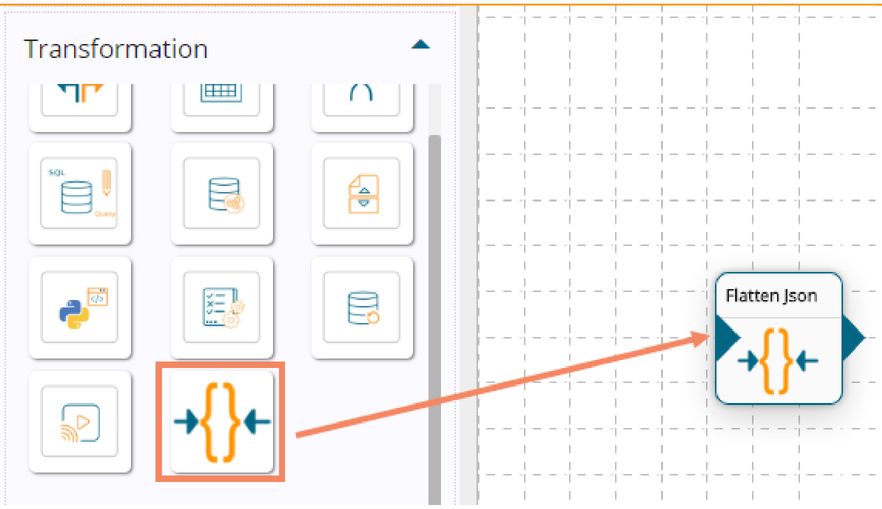
BDB Pipeline provides the Flatten JSON component for flatting a nested JSON as well as filtering a data from a nested JSON. It can filter and remove the unnecessary dumping of data on a Kafka topic.

It can run in the following scenarios:

1. Real-Time
2. Batch Process

The steps to configure the Flatten JSON component are described in this section.

i) Drag and drop to the Flatten Json component to the pipeline Workflow Editor.



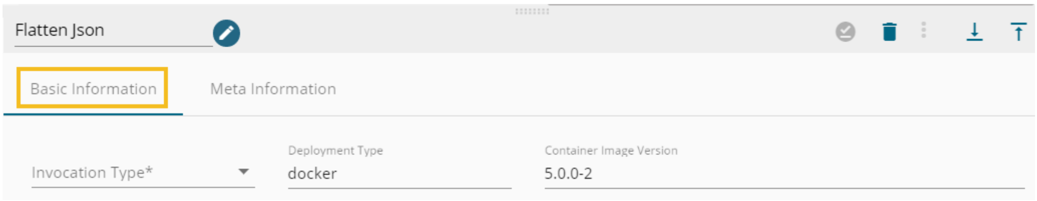
ii) Click the dragged Flatten JSON component to get the Configuration fields:

iii) The **Basic Information** tab opens by default.

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

b. Deployment Type: It comes preselected based on the component.

c. Container Image Version: It comes preselected based on the component.



Flatten Json

Basic Information | Meta Information

Invocation Type*

Deployment Type: docker

Container Image Version: 5.0.0-2

iv) Open the **Meta Information** tab to provide the connection-specific details.

a. Name: The user must provide the full path of data by using dot(.) which he wants to retrieve from a nested json.

For instance - We have a nested JSON of the following format, and we want to retrieve device type of the Device in dashboard_1 we must give the name as **dashboard_1.Device.devicetype.**

```
    "dashboard_1": {
        "Device": [
          {
            "devicetype": "Motor-N1",
            "Vibration": 53,
            "Temperature": 23
          },
          {
            "devicetype": "Motor-N2",
            "Vibration": 48,
            "Temperature": 40
          },
          {
            "devicetype": "Motor-S1",
            "Vibration": 98,
            "Temperature": 33
          },
          {
            "devicetype": "Motor-S2",
            "Vibration": 77,
            "Temperature": 37
          },
          {
            "devicetype": "Gearbox-LS",
            "Vibration": 56,
            "Temperature": 43
          },
          {
            "devicetype": "Gearbox-SS",
            "Vibration": 97,
            "Temperature": 50
          }
        ],
        "Brake": [
          {
            "devicetype": "Brake-N",
            "Thickness": 19,
            "Temperature": 414
          },
          {
            "devicetype": "Brake-S",
            "Thickness": 11,
            "Temperature": 397
          }
        ],
        "RPM": 675,
        "LOAD(T)": 40,
        "MTBF": 562,
        "MTTR": 62,
        "RUN": 3609,
        "RTBM": 317
      },
      "dashboard_2": {
        "Cast_Spd_St1": 0.013,
        "Cast_Spd_St2": -0.01,
```

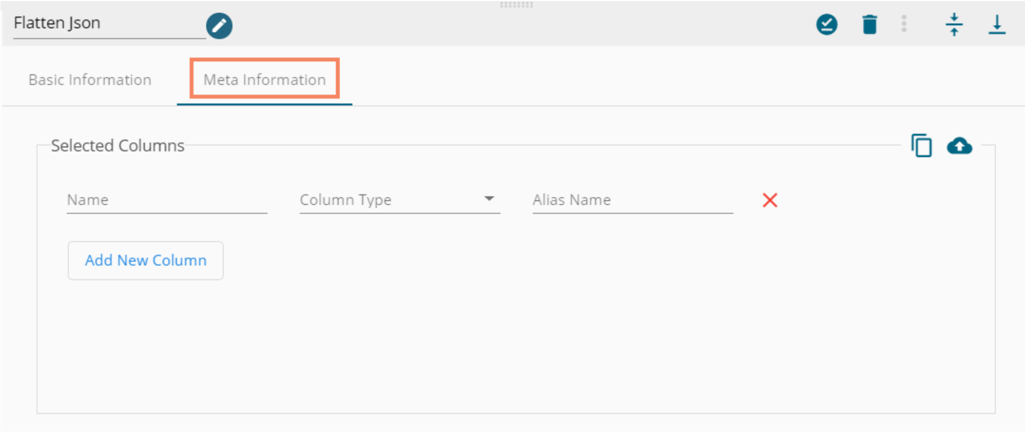
v) Selected **Columns**: The users can select some specific columns from the table to read data instead of selecting a complete table; this can be achieved via the ‘Selected Columns’ section. Select the columns which you want to read and if you want to change the name of the column, then put that name in alias name section otherwise **keep alias name same as of column name** and then select a Column Type from the drop-down menu.

or

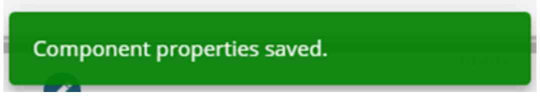
Use '**Download Data**' and '**Upload File**' options to select the desired columns.

1. Upload File: The user can upload the existing system files (CSV, JSON) using the '**Upload File**' icon (file size must be less than 2 MB).
2. Download Data (Schema): Users can download the schema structure in JSON format by using the '**Download Data**' icon.

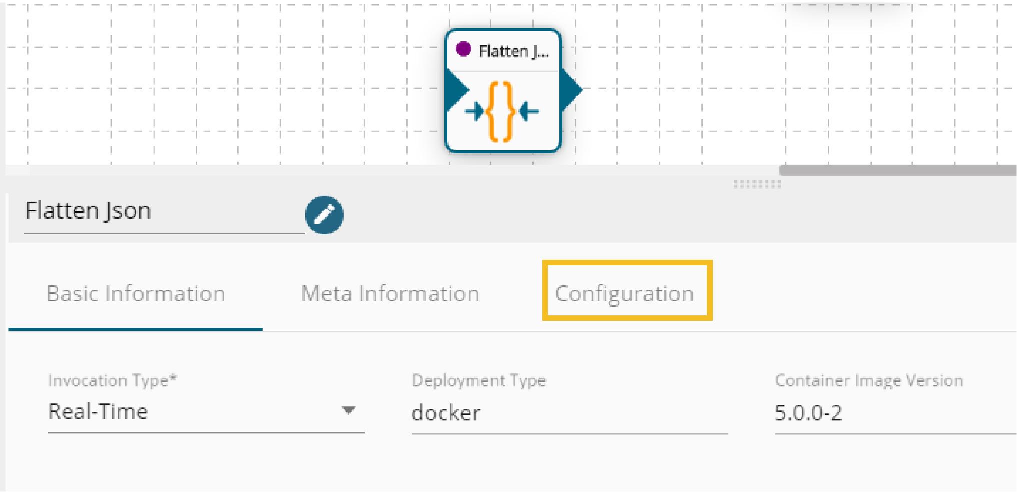
vi) Click the '**Save Component in Storage'** icon to save the configuration.



vii) Update the Pipeline by clicking the '**Update Pipeline**' icon.


Component properties saved.

viii) Open the Flatten JSON component again to see the Configuration tab below.



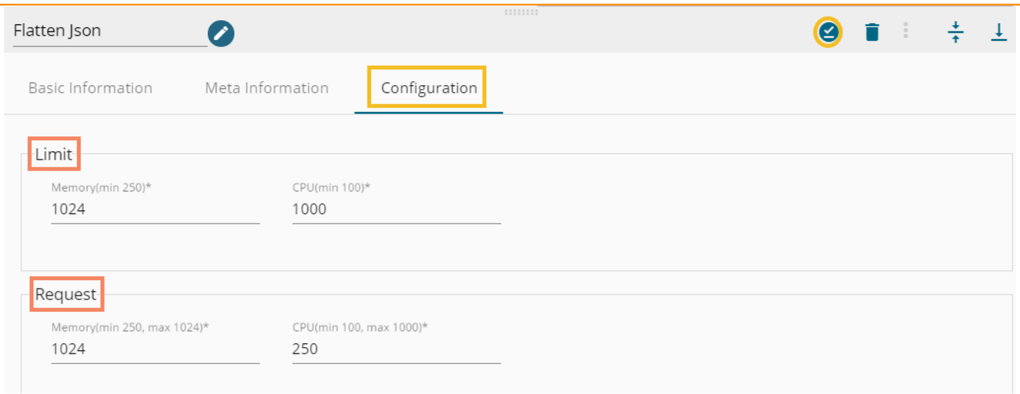
Invocation Type* Real-Time

Deployment Type docker

Container Image Version 5.0.0-2

ix) Modify the values for the Configuration tab.

x) Click the '**Save Component in Storage**' icon.


Flatten Json

Basic Information | Meta Information | Configuration

Limit

Memory(min 250)*: 1024

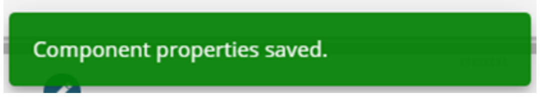
CPU(min 100)*: 1000

Request

Memory(min 250, max 1024)*: 1024

CPU(min 100, max 1000)*: 250

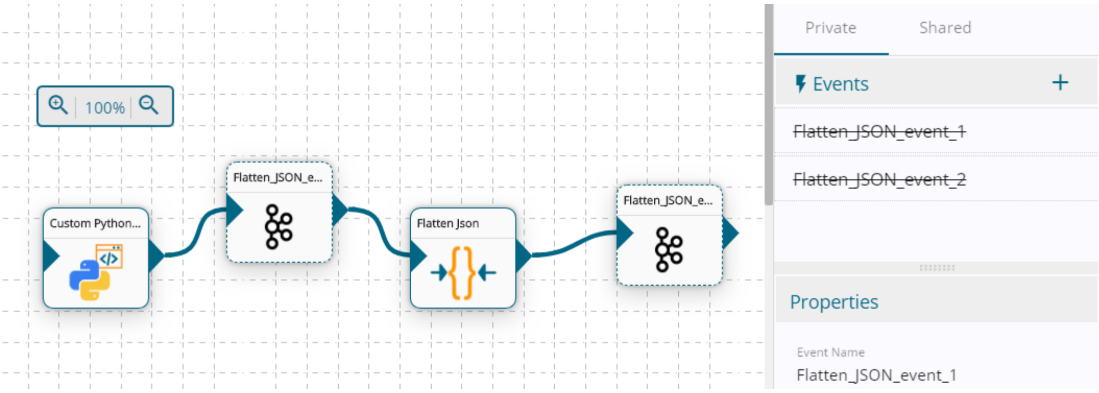
xi) A message appears to notify the action.


Component properties saved.

xii) The Flatten JSON is ready to be used in the Pipeline workflow

Note:

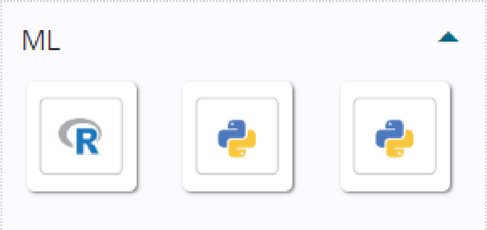
a. The Flatten JSON component cannot generate data in rows.
b. The Flatten JSON component must have an in-event and out-event in the workflow, as displayed below:



Event Name: Flatten_JSON_event_1

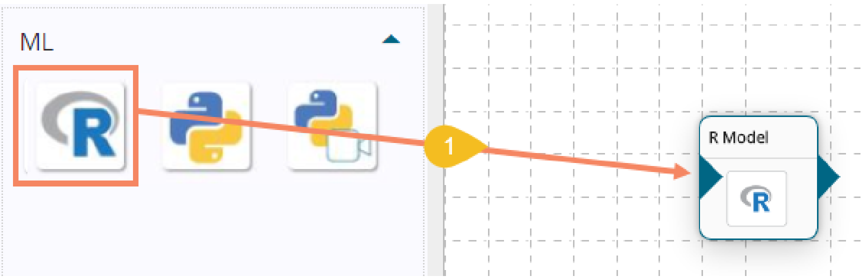
8.4. ML

ML Model Runner Components allow us to use the models created on R and Python Workspaces of the Data Science Workbench inside the Data Pipeline.



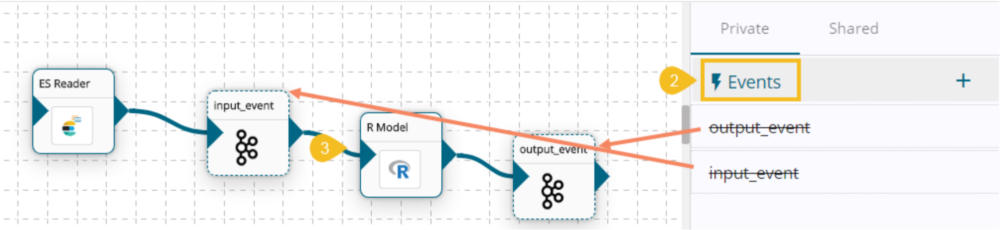
8.4.1. R Model Runner

i) Drag and Drop R Model (runner) component to the Workflow Editor.



ii) Model Runner requires data input from an Event and sends the processed data to another Event, so create two Events.

iii) Drag them to the workflow editor and connect to the dragged R Model (runner) component as displayed below:
Note: The data in Input Event can come from any Ingestion, Reader or shared events.



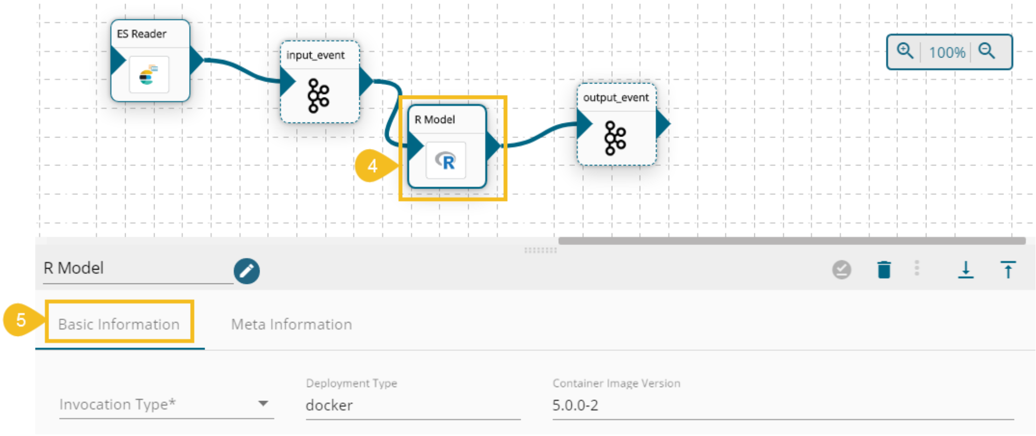
iv) Click on the R Model Component to get the Component Properties tabs.

v) **Basic Information**: It is the default tab to open for the component.

a. Select an Invocation type from the drop-down menu to confirm the running mode of the reader component. Select ‘**Real-Time**’ or ‘**Batch**’ from the drop-down menu.

b. Deployment Type: It displays the deployment type for the component. This field comes pre-selected.

c. Container Image Version: It displays the image version for the docker container. This field comes pre-selected.



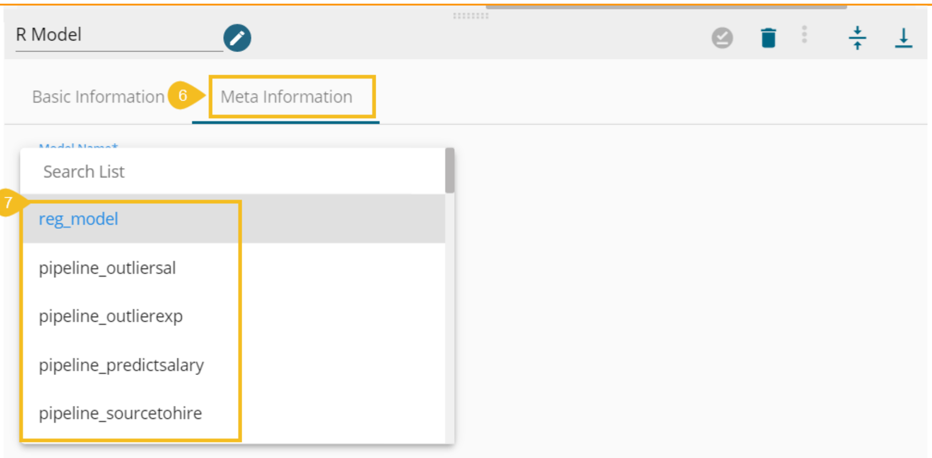
Invocation Type*

Deployment Type: docker

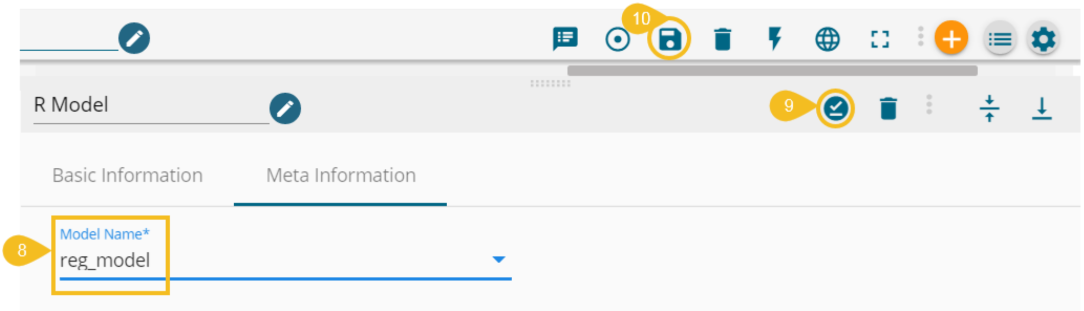
Container Image Version: 5.0.0-2

vi) Click the **Meta Information tab** to open.

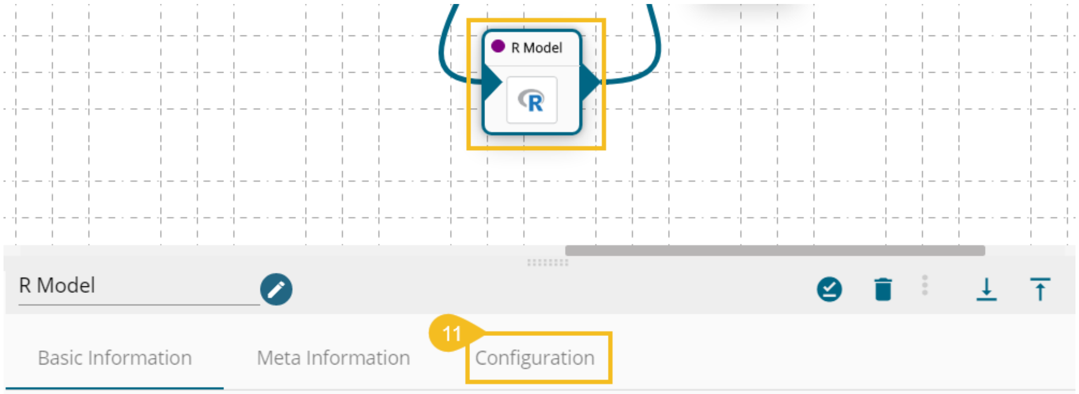
vii) All the exported R models appear under the ‘**Model Name**’ menu.



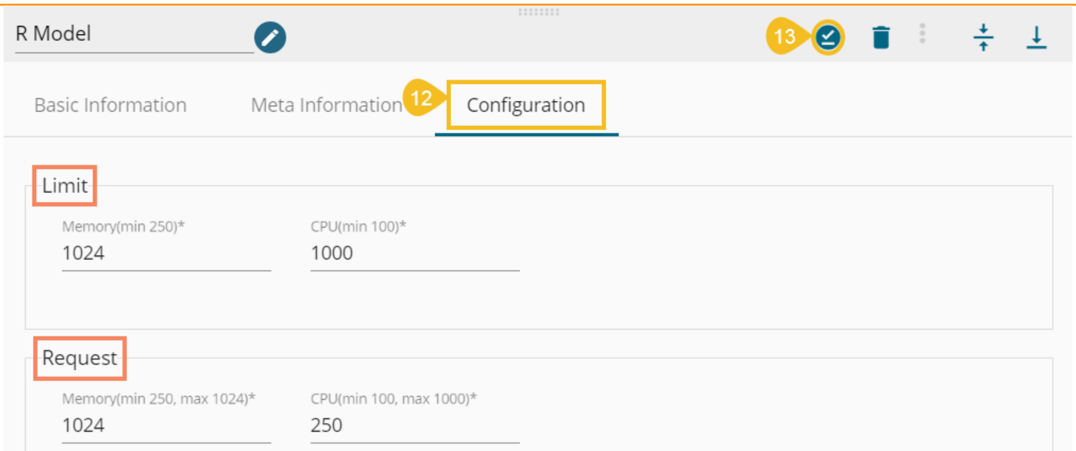
viii) Search and select the model you want to use.
ix) Save the R Model component (A message appears to confirm that the component properties have been saved).
x) Save the pipeline by using the '**Update Pipeline**' icon.



xi) Open the R model again to see the **Configuration tab** below.



xii) Modify the Configuration tab values (if needed).
xiii) Click the '**Save Component in Storage**' icon to save the component.


R Model

Basic Information | Meta Information | Configuration

Limit

Memory(min 250)*: 1024

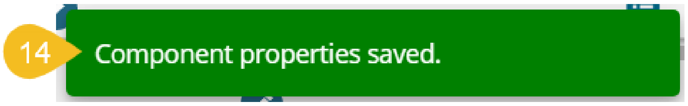
CPU(min 100)*: 1000

Request

Memory(min 250, max 1024)*: 1024

CPU(min 100, max 1000)*: 250

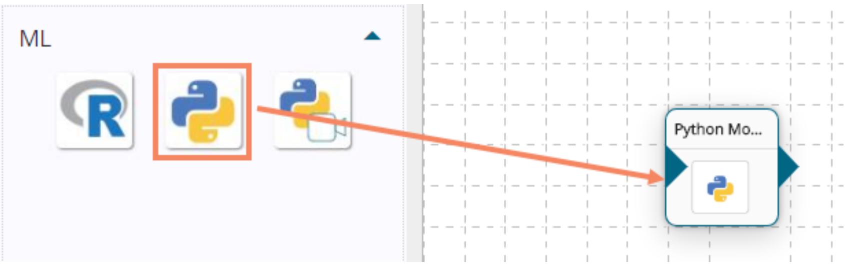
xiv) A message appears to confirm the action.


Component properties saved.

xv) The R Model runner is configured. The component reads to read the data coming to an input event, runs the model, and gives the output data with processed columns to the output event.

8.4.2. Python Model Runner

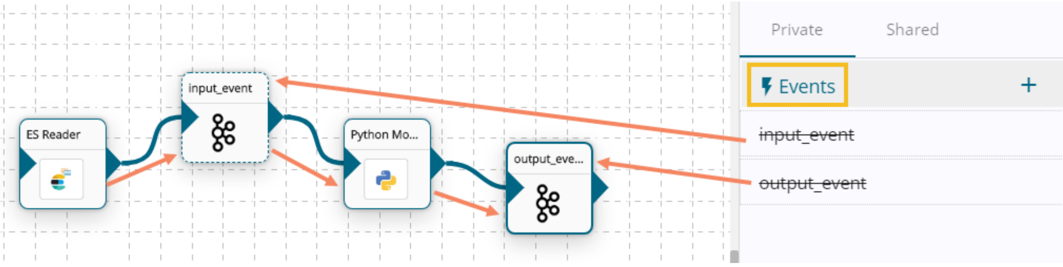
i) Drag and drop the Python Model (runner) component to the Workflow Editor.



ii) The Python Model runner requires input data from an Event and sends the processed data to another Event. So, create two Events and drag them onto the Workspace.

iii) Connect the input and output events with the Python Model runner component as displayed below.

iv) The data in the input event can come from any Ingestion, Readers or shared events.



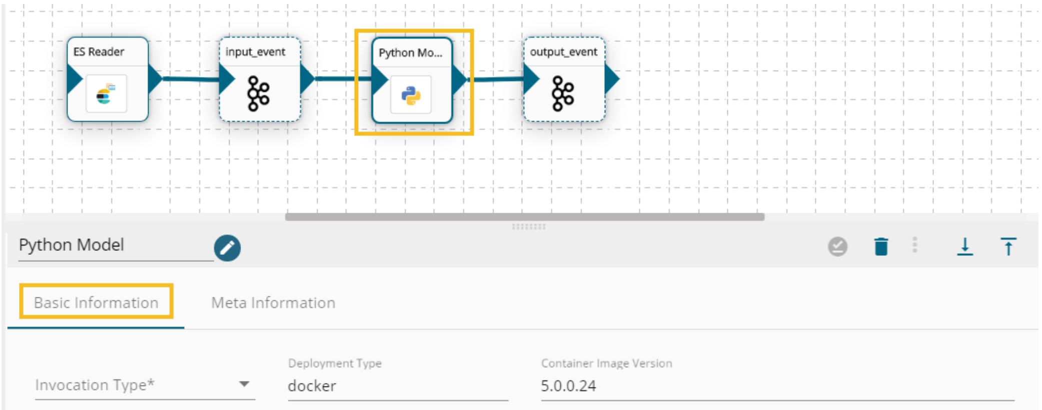
v) Click the Python Model runner component to get the component properties tabs below.

vi) **Basic Information**: It is the default tab to open for the component.

a. Select an Invocation type from the drop-down menu to confirm the running mode of the reader component. Select '**Real-Time**' or '**Batch**' from the drop-down menu.

b. Deployment Type: It displays the deployment type for the component. This field comes pre-selected.

c. Container Image Version: It displays the image version for the docker container. This field comes pre-selected.



Invocation Type*

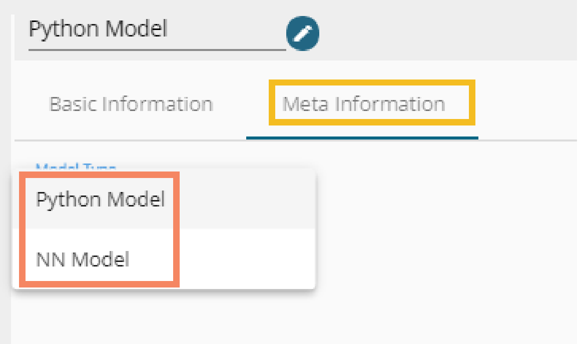
Deployment Type: docker

Container Image Version: 5.0.0.24

vii) Open the **Meta Information tab** and click on the Model name field.

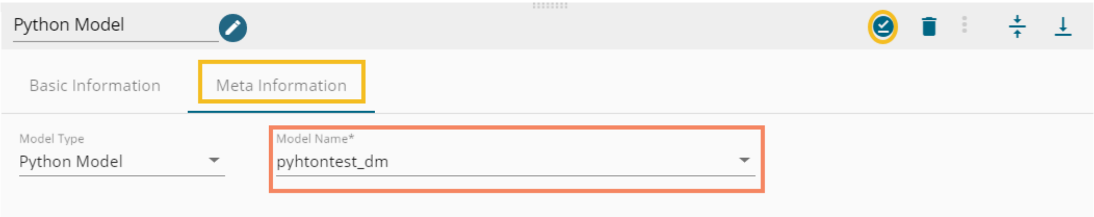
viii) The model type displays the Python Model and NN Model option.

ix) Select a model Type option.



x) Search the model you want to use and select the model.

xi) Save the Python Model runner component.

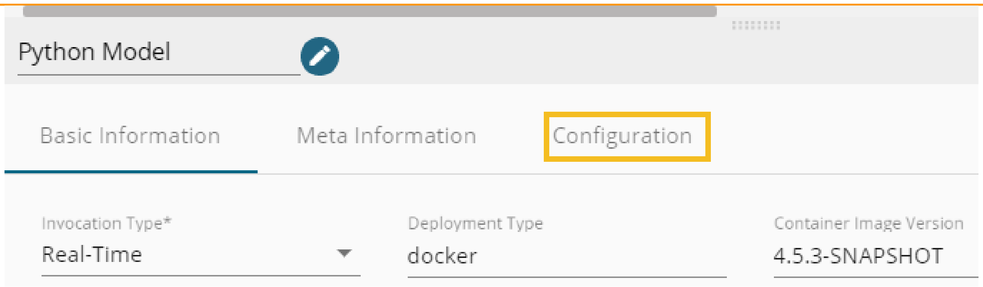


Model Type: Python Model

Model Name*: pyhtontest_dm

xii) Click the '**Update Pipeline**' icon (A notification message appears to confirm the action).

xiii) Open the Python Model runner component again to see the **Configuration tab** below.


Python Model

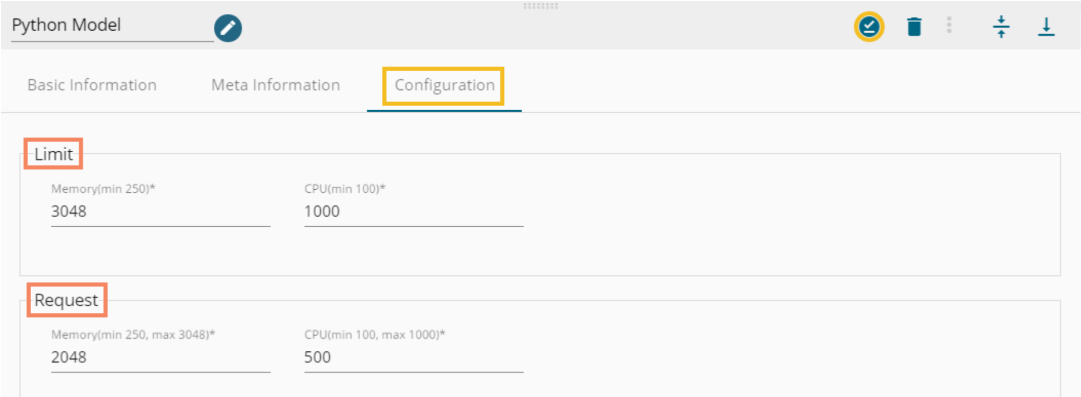
Invocation Type* Real-Time

Deployment Type docker

Container Image Version 4.5.3-SNAPSHOT

xiv) Modify the Configuration values (if needed).

xv) Click the '**Save Component in Storage**' ✓ icon.


Python Model

Limit

Memory(min 250)* 3048

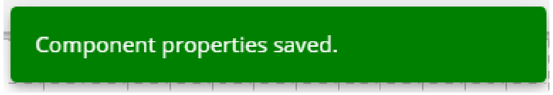
CPU(min 100)* 1000

Request

Memory(min 250, max 3048)* 2048

CPU(min 100, max 1000)* 500

xvi) A notification message appears.


Component properties saved.

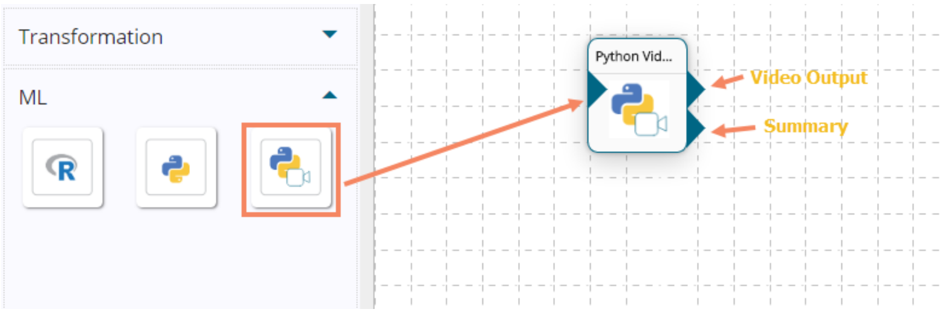
xvii) The Python Model runner component reads the data coming to the input event, runs the model, and gives the output data with predicted columns to the output event.

8.4.3. Python Video Model Runner

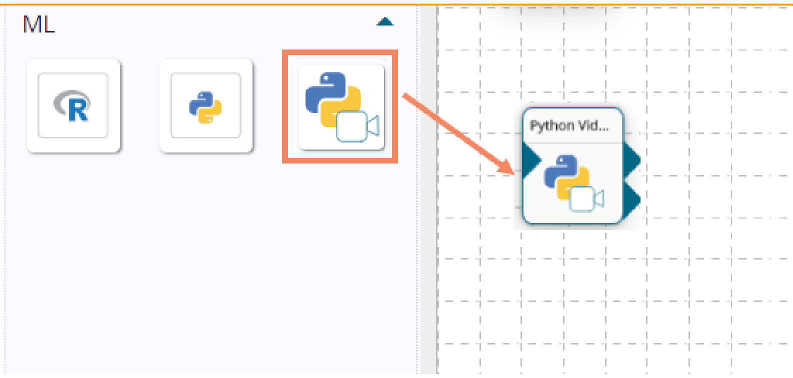
This component is responsible for applying the machine learning model created in the data science workbench module on the video ingested in the pipeline.

The Python Video Model gives two outputs:

1. Video output with classification and labeling.
2. Summary of the applied model on the video.



i) Drag and drop Python Video Model Component from the ML section to the Workflow Editor.



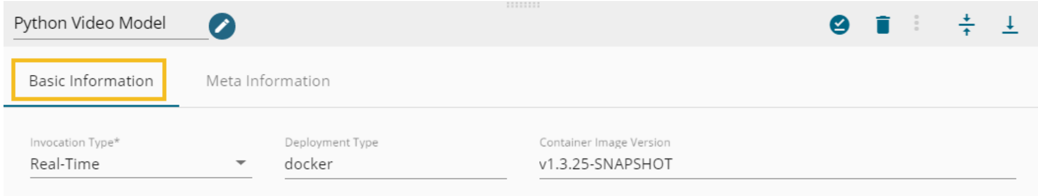
ii) Click on the Python Video Model component to see the component properties below.

iii) **Basic Information**: It is the default tab to open for the component.

a. Select an Invocation type from the drop-down menu to confirm the running mode of the reader component. Select '**Real-Time**' or '**Batch**' from the drop-down menu.

b. Deployment Type: It displays the deployment type for the component. This field comes pre-selected.

c. Container Image Version: It displays the image version for the docker container. This field comes pre-selected.



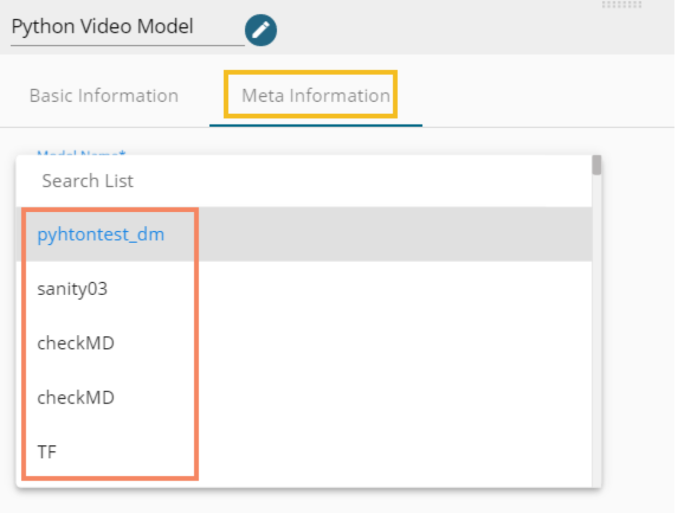
Invocation Type*: Real-Time

Deployment Type: docker

Container Image Version: v1.3.25-SNAPSHOT

iv) Open the **Meta Information tab** and click on the Model Name field.

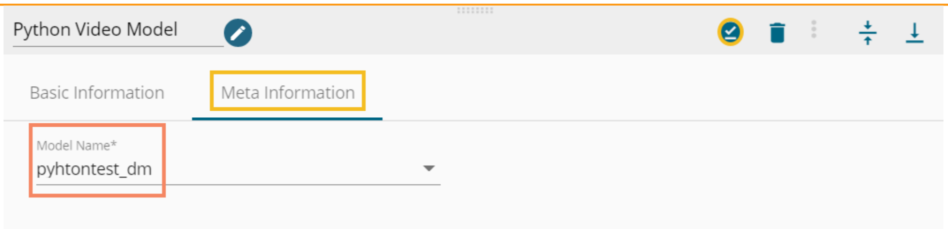
v) All the deployed python video models get listed.



pyhtontest_dm
sanity03
checkMD
checkMD
TF

vi) Select a model from the Model Name drop-down menu.

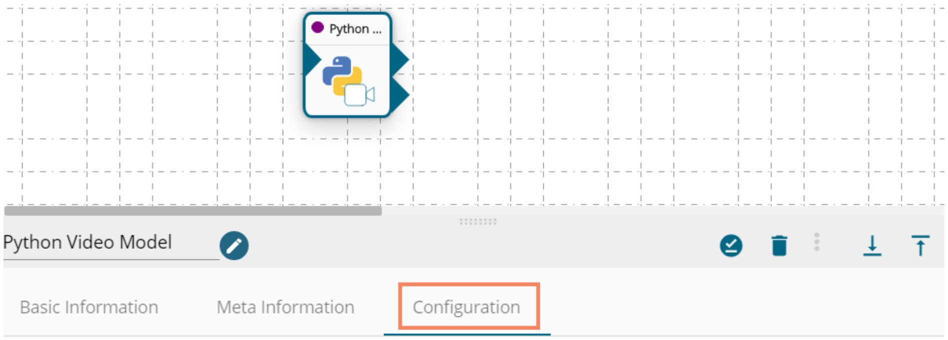
vii) Save the component (A notification message appears to assure the action).



Model Name*
pyhtontest_dm

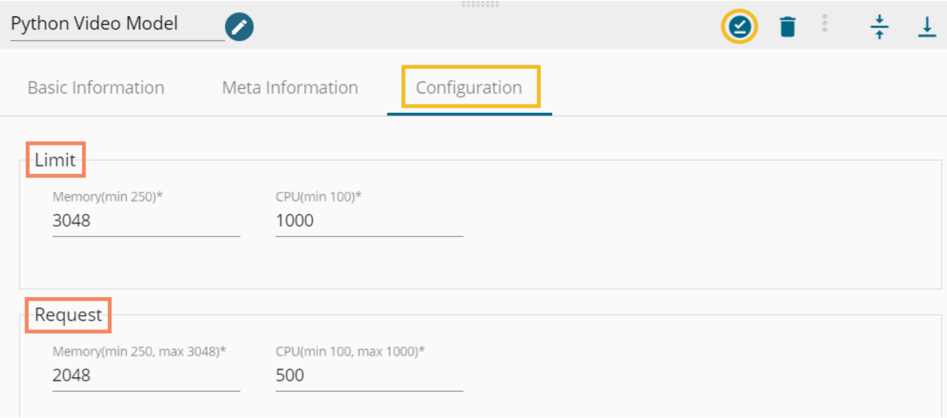
viii) Click the ‘**Update Pipeline**’ icon. (A notification message appears to assure the action).

ix) Open the Python Video Model runner component again to see the **Configuration tab** below.



x) Modify the Configuration values (if needed)

xi) Click the ‘**Save Component in Storage**’ icon.



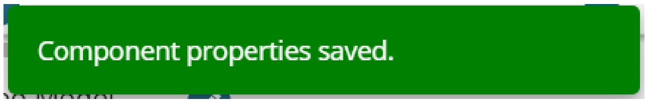
Limit

Memory(min 250)*: 3048
CPU(min 100)*: 1000

Request

Memory(min 250, max 3048)*: 2048
CPU(min 100, max 1000)*: 500

xii) A notification message appears to inform about the completion of the action.



Component properties saved.

xiii) The Python Video Model runner component gets configured to be used in the Pipeline Workflow.

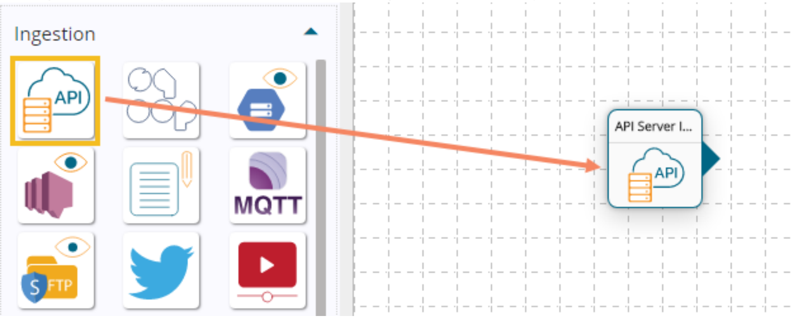
Note: The user can connect the Video writer component to the first output nodes of the Python Video Model runner to writer the output to an SFTP location. The second output is the textual summary which can be parked through using any data writer component based on the user requirement.

8.5. Ingestion

Ingestion components allow the users to ingest data in the pipeline from outside the pipeline as in files from some SFTP location or manual ingestion of data using a service in real-time.

8.5.1. API Server Ingestion

i) Drag and Drop the **API Server Ingestion** component to the Workflow Editor.



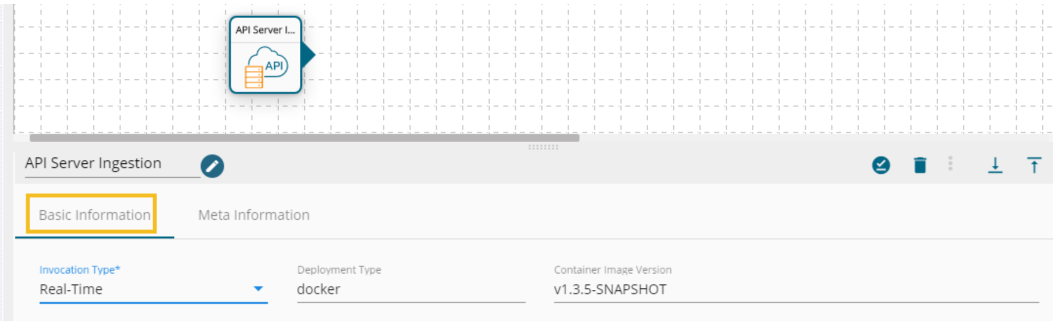
ii) Click on the dragged ingestion component to get the component properties tabs.

iii) Configure the **Basic Information tab**.

a. Select an Invocation type from the drop-down menu to confirm the running mode of the reader component. The supported invocation type is ‘**Real-Time**’.

b. Deployment Type: It displays the deployment type for the component. This field comes pre-selected.

c. Container Image Version: It displays the image version for the docker container. This field comes pre-selected.



API Server Ingestion

Basic Information | Meta Information

Invocation Type*: Real-Time

Deployment Type: docker

Container Image Version: v1.3.5-SNAPSHOT

iv) Open the ‘**Meta Information**’ **tab.**

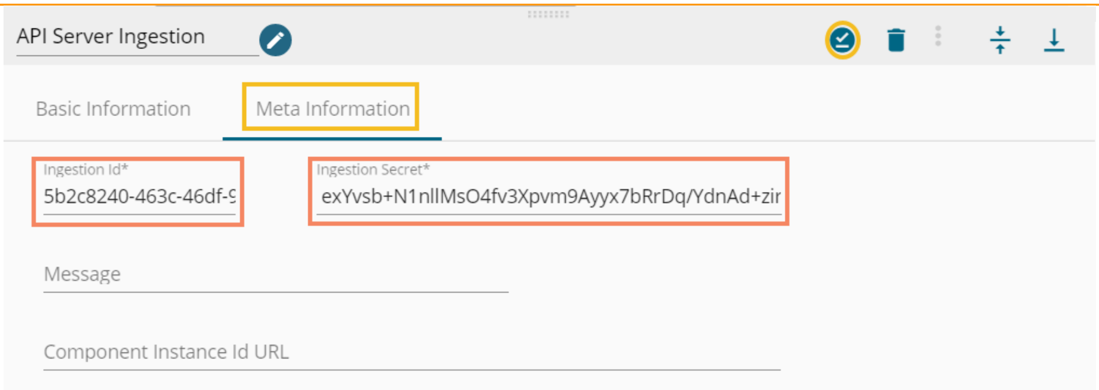
a. Provide the Ingestion Id

b. Provide the Ingestion Secret (Key)

c. Write a Message (optional)

d. The Component Instance Id URL generates (after providing the Ingestion Id and Secret Key).

v) Click the ‘**Save Component in Storage**’ ✓ icon to save the API Server Ingestion.



API Server Ingestion

Ingestion Id*
5b2c8240-463c-46df-9

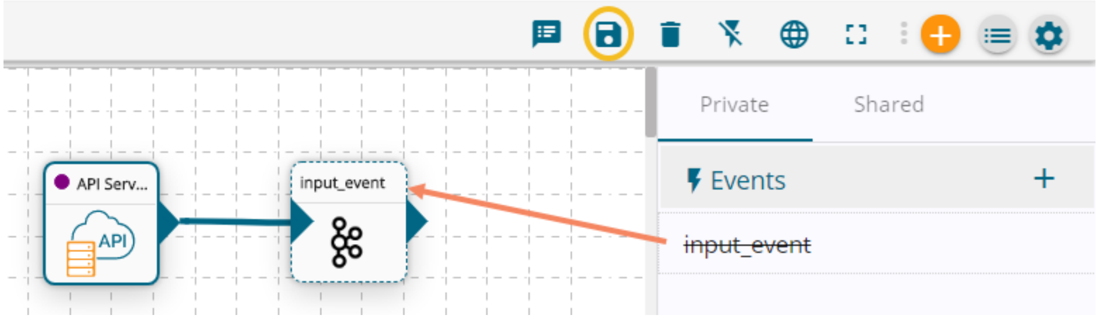
Ingestion Secret*
exYvsb+N1nllMsO4fv3Xpvm9Ayyx7bRrDq/YdnAd+zir

Message

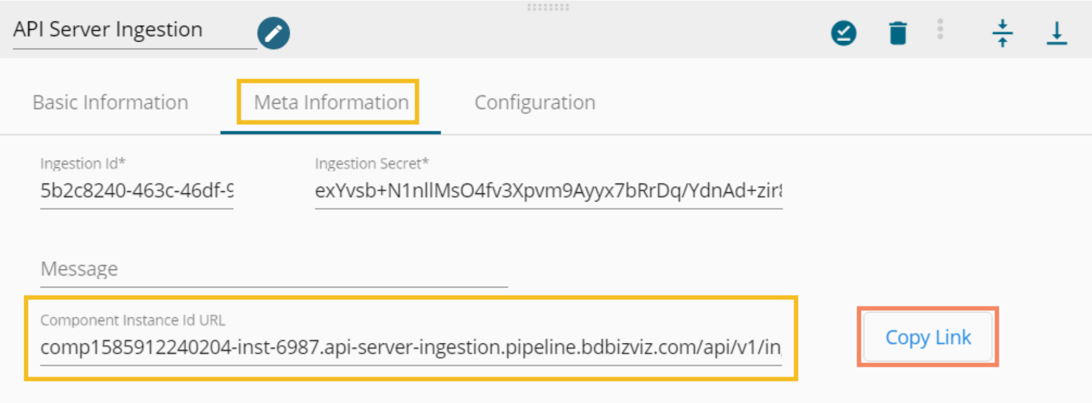
Component Instance Id URL

vi) A notification message appears to inform that the component has been saved.

vii) Connect the API Ingestion component to an Event and save the Pipeline.



viii) As we save the pipeline, an auto-generated **Component Ingestion URL** appears inside the Meta Information tab to ingest the data.



API Server Ingestion

Ingestion Id*
5b2c8240-463c-46df-9

Ingestion Secret*
exYvsb+N1nllMsO4fv3Xpvm9Ayyx7bRrDq/YdnAd+zir

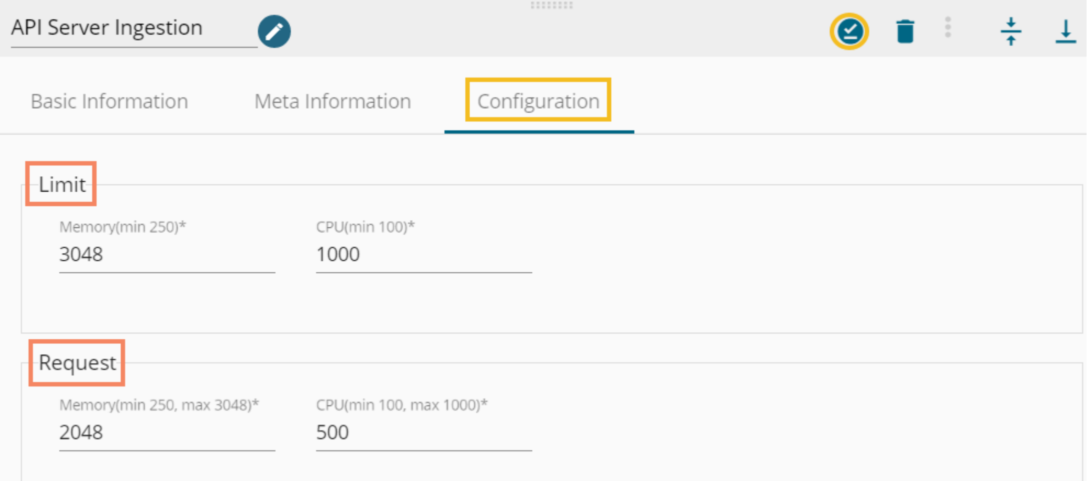
Message

Component Instance Id URL
comp1585912240204-inst-6987.api-server-ingestion.pipeline.bdbizviz.com/api/v1/in

ix) The **Configuration tab** also appears after saving the Ingestion component and updating the pipeline.

x) Modify the Configuration values (if needed).

xi) Click the '**Save Component in Storage**' icon.



Limit

Memory(min 250)*: 3048

CPU(min 100)*: 1000

Request

Memory(min 250, max 3048)*: 2048

CPU(min 100, max 1000)*: 500

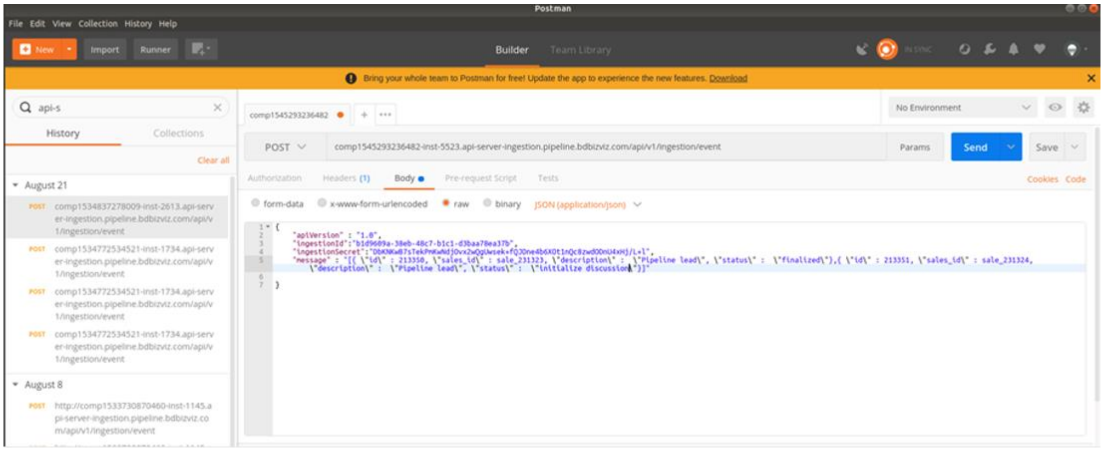
xii) A notification message appears to inform about the action.

Component properties saved.

xiii) The Ingestion component can be used in a pipeline workflow as an input component after component configuration.

Note:

a. The users can use the Component Ingestion URL with the following format in the program or add anywhere in the third-party portal.



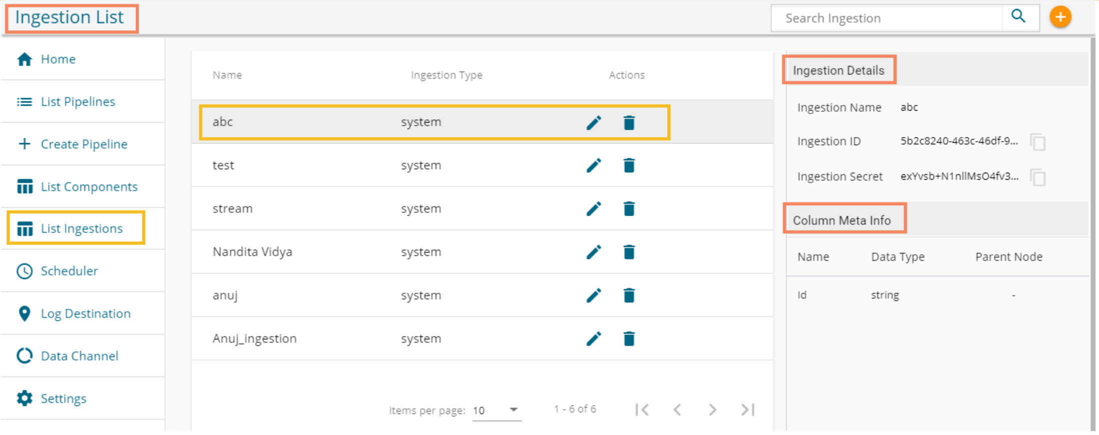
Postman
File Edit View Collection History Help
New
Import
Runner
Builder
Team Library
IN SYNC
Bring your whole team to Postman for free! Update the app to experience the new features. Download
api-s
History
Collections
Clear all
August 21
POST comp1534837278009-inst-2613.api-server-ingestion.pipeline.bdbizviz.com/api/v1/ingestion/event
POST comp1534772534521-inst-1734.api-server-ingestion.pipeline.bdbizviz.com/api/v1/ingestion/event
POST comp1534772534521-inst-1734.api-server-ingestion.pipeline.bdbizviz.com/api/v1/ingestion/event
POST comp1534772534521-inst-1734.api-server-ingestion.pipeline.bdbizviz.com/api/v1/ingestion/event
August 8
POST http://comp1533730870460-inst-1145.api-server-ingestion.pipeline.bdbizviz.com/api/v1/ingestion/event
comp1545293236482
No Environment
Params
Send
Save
Authorization
Headers (1)
Body
Pre-request Script
Tests
Cookies
Code
○ form-data
○ x-www-form-urlencoded
● raw
○ binary
JSON (application/json)

POST comp1545293236482-inst-5523.api-server-ingestion.pipeline.bdbizviz.com/api/v1/ingestion/event

```
{
    "apiVersion" : "1.0",
    "ingestionId":"b1d9609a-38eb-48c7-b1c1-d3baa78ea37b",
    "ingestionSecret":"DbKNKwB7sTekPmKwNdjOvx2wQgUwsek+fQJOne4b6XOt1nQc8zwdODnU4xHj/L+l",
    "message" : "[{ \"id\" : 213350, \"sales_id\" : sale_231323, \"description\" :  \"Pipeline lead\", \"status\" :  \"finalized\"},{ \"id\" : 213351, \"sales_id\" : sale_231324,
        \"description\" :  \"Pipeline lead\", \"status\" :  \"initialize discussion\"}]"

}
```

b. Click the List Ingestions option to open the list of ingestions. Select an Ingestion from the list. The Ingestion Details opens on the top-right side of the page with Ingestion ID and Ingestion Secret (Key).



Name	Ingestion Type	Actions
abc	system	
test	system	
stream	system	
Nandita Vidya	system	
anuj	system	
Anuj_ingestion	system	

Ingestion Details

Ingestion Name abc

Ingestion ID 5b2c8240-463c-46df-9...

Ingestion Secret exYvsb+N1nllMsO4fv3...

Column Meta Info

Name	Data Type	Parent Node
Id	string	-

Service Info :

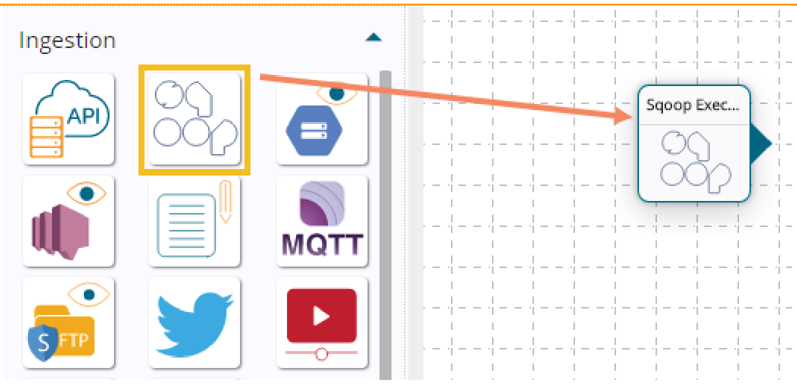
- IngestionID and Ingestion Secret (key) must same as fill in a pipeline component
- A message accepting data in Array as String format appears.
- In the message, all the objects must follow the same column info as declared in the settings page at the time ingestion create.

URL → comp1545293236482-inst-5523.api-server-ingestion.pipeline.bdbizviz.com/api/v1/ingestion/event (will be Auto-generated Component Ingestion URL)
Type → POST
Content-Type → application/json
Body →
{
 "apiVersion": "1.0",
 "ingestionId": "b1d9609a-38eb-48c7-b1c1-d3baa78ea37b",
 "ingestionSecret":
"DbKNKwB7sTekPmKwNdjOvx2wQgUwsek+fQJDne4b6XOt1nQc8zwdODnU4xHj/L+l",
 "message": "[{ \"id\" : 213350, \"sales_id\" : sale_231323, \"description\" : \"Pipeline lead\", \"status\" : \"finalized\"},{ \"id\" : 213351, \"sales_id\" : sale_231324, \"description\" : \"Pipeline lead\", \"status\" : \"initialize discussion\"}]"
}

8.5.2. Sqoop Executer

The Sqoop component in BDB Pipeline helps the users to transfer tables and databases from RDBMS (supported by Sqoop) to HDFS (Hadoop System Distributed File).

i) Drag and drop the Sqoop component to the pipeline Workflow Editor.



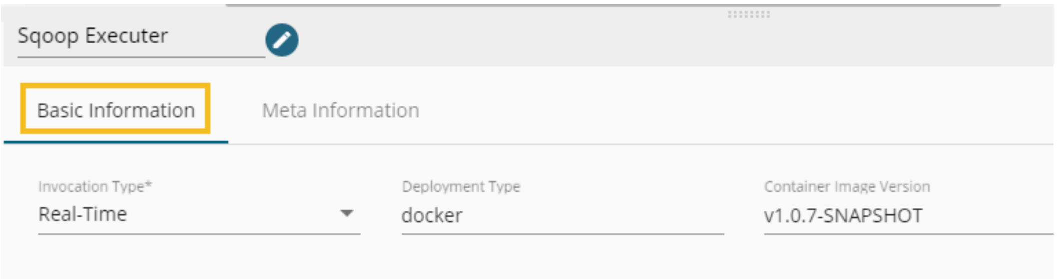
ii) Click the Sqoop Executer to see the component properties tabs below.

iii) The **Basic Information tab** opens by default by clicking the Sqoop Executer component.

a. Select an Invocation type from the drop-down menu to confirm the running mode of the reader component. The supported invocation type is ‘**Real-Time**’.

b. Deployment Type: It displays the deployment type for the component. This field comes pre-selected.

c. Container Image Version: It displays the image version for the docker container. This field comes pre-selected.



Sqoop Executer

Basic Information | Meta Information

Invocation Type*: Real-Time

Deployment Type: docker

Container Image Version: v1.0.7-SNAPSHOT

iv) Open the **Meta Information tab** and provide the following information:

a. Username: Provide the username.

b. Host – Provide a host or IP address.

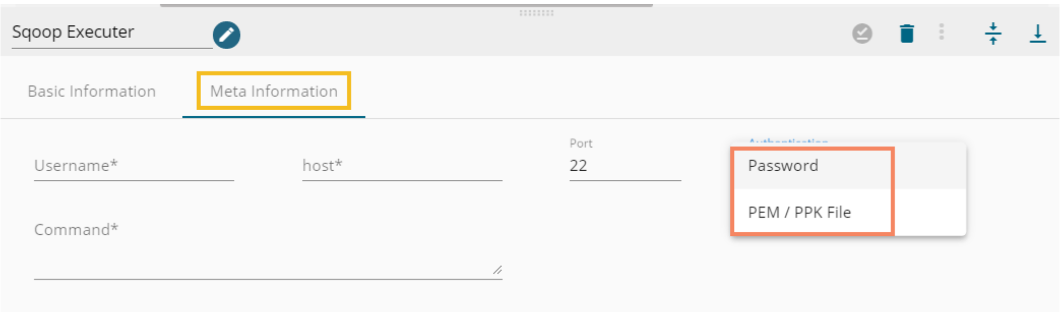
c. Port- Provide a Port number (the default number for these fields is 22).

d. Authentication- Select an authentication type from the drop-down menu out of the given choices

i. Password- provide correct password for this authentication option

ii. PEM/PPK File-choose a file and provide the file name if the user selects this authentication option.

e. Command-Enter the relevant Sqoop command.



Sqoop Executer

Basic Information | Meta Information

Username*

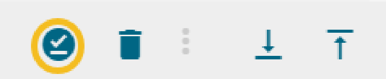
host*

Port: 22

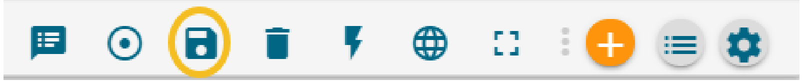
Authentication: Password / PEM / PPK File

Command*

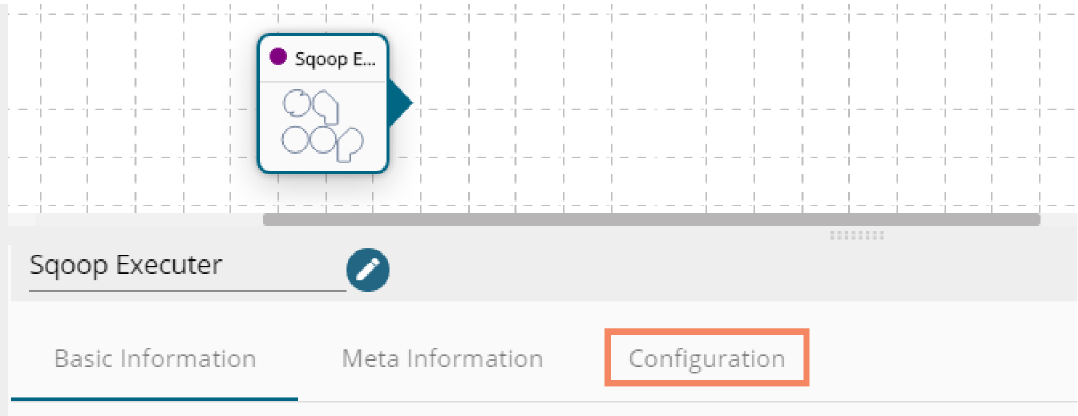
v) Click the '**Save Component in Storage**' icon to save the component (A notification message appears to inform the completion of the action).



vi) Click the '**Update Pipeline**' icon. (A notification message appears to inform that the pipeline has been updated).

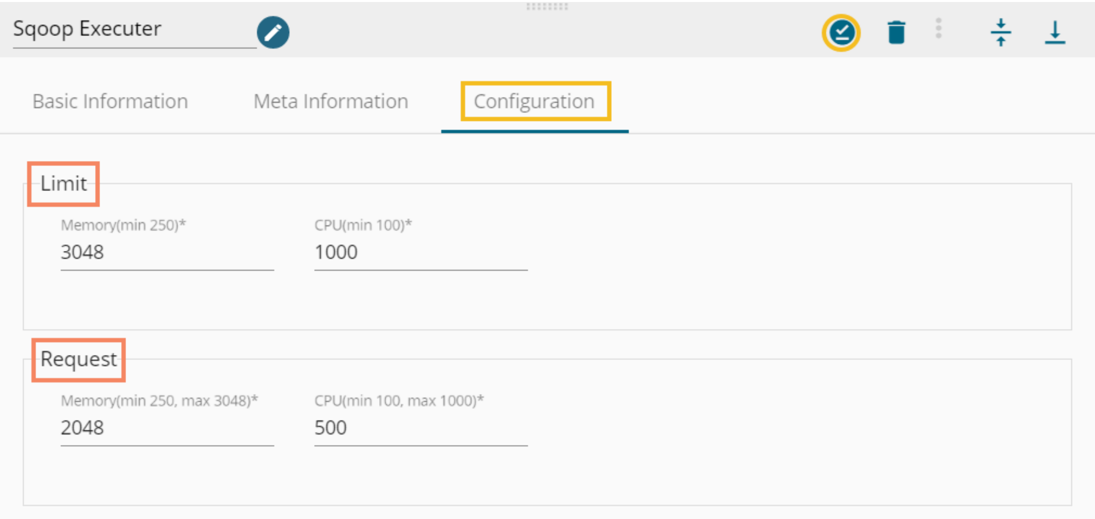


vii) Open the Sqoop component again to see the **Configuration tab** below.



viii) Modify the values of the Configuration tab (if needed).

ix) Click the '**Save Component in Storage**' icon to save the component (A notification message appears to inform the completion of the action).



Limit

Memory(min 250)*: 3048

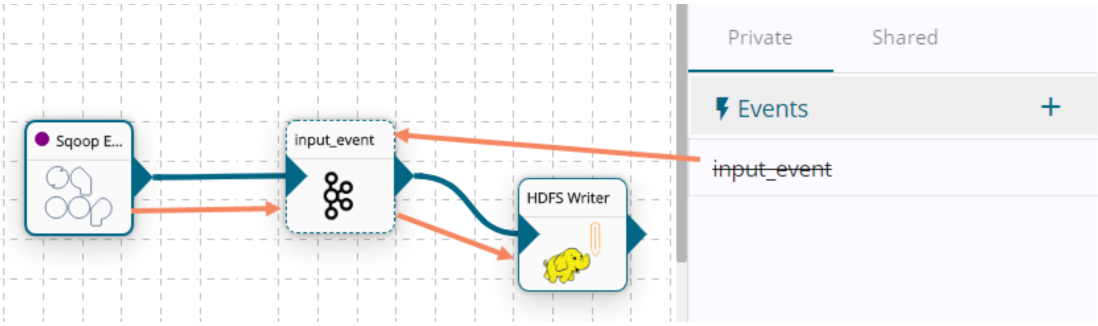
CPU(min 100)*: 1000

Request

Memory(min 250, max 3048)*: 2048

CPU(min 100, max 1000)*: 500

x) Create one Event and connect it to the Sqoop component for receiving the success message. Connect an HDFS writer to the input event to create a workflow. Once the data gets transferred to the HDFS, the user can use it for further processing in the pipeline.



8.5.3. GCS Monitor

8.5.3.1. Creating GCS(Google Cloud Storage) Credential

The users require Google credentials to access the GCS bucket via the GCS monitor. The steps to create a service account key are described below:

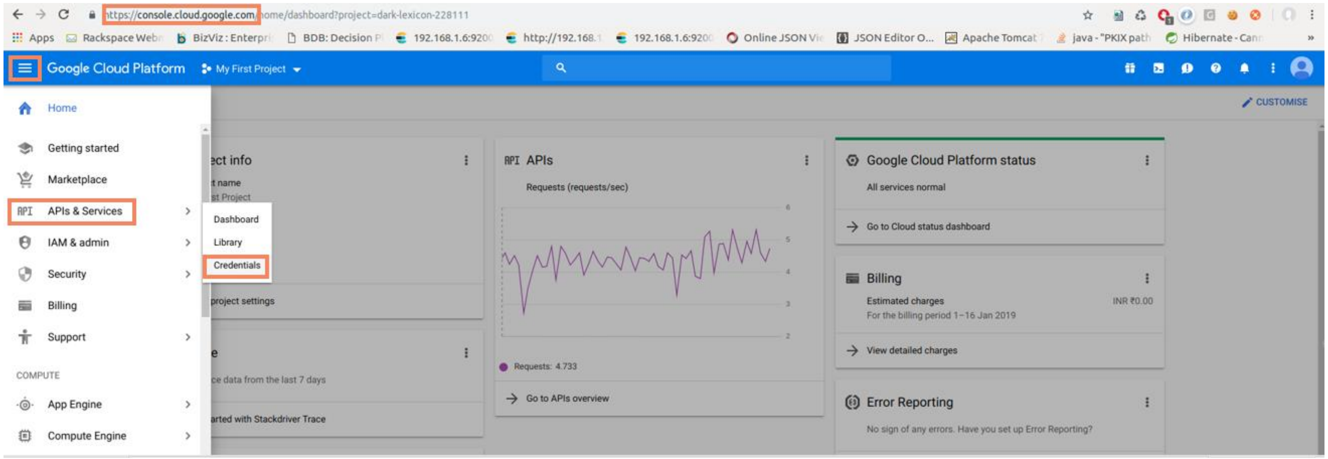
i) Open **Google Cloud Console** or open the link- https://console.cloud.google.com/home

ii) Click on the menu button (from the top left side) to open the list of options

iii) Select and click the **APIs & Services** option from the menu list

iv) A context menu opens as displayed in the image below

v) Select the **Credentials** option from the context menu



Note: The user can ignore the above-given steps to create credentials if a Service Account key is already created.

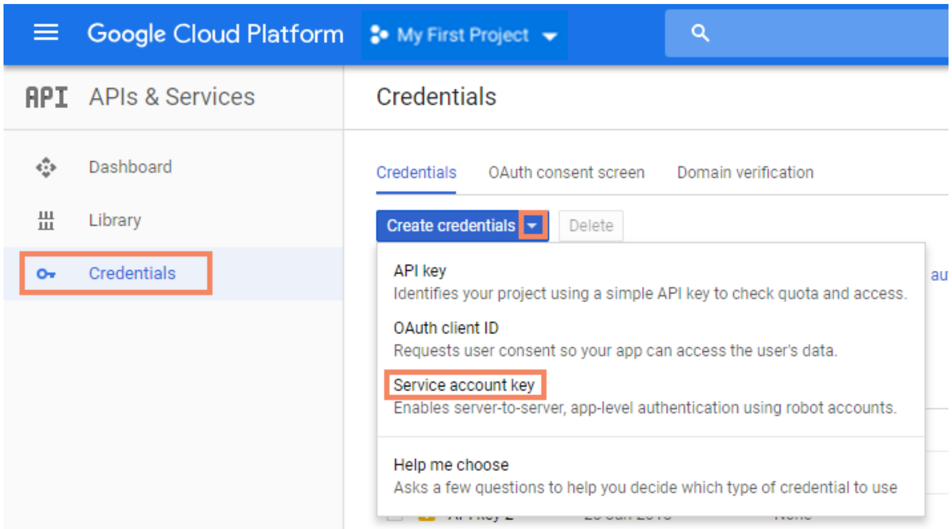
vi) The ‘**Credentials**’ opens as displayed in the below image.

vii) Click the ‘**Create Credentials’** drop-down option.

viii) **A menu opens with 3 types of authorization options:**

1. API Key
2. OAuth Client ID
3. Service account key

ix) Click the ‘**Service Account Key’** authorization option.



API key
Identifies your project using a simple API key to check quota and access.

OAuth client ID
Requests user consent so your app can access the user's data.

Service account key
Enables server-to-server, app-level authentication using robot accounts.

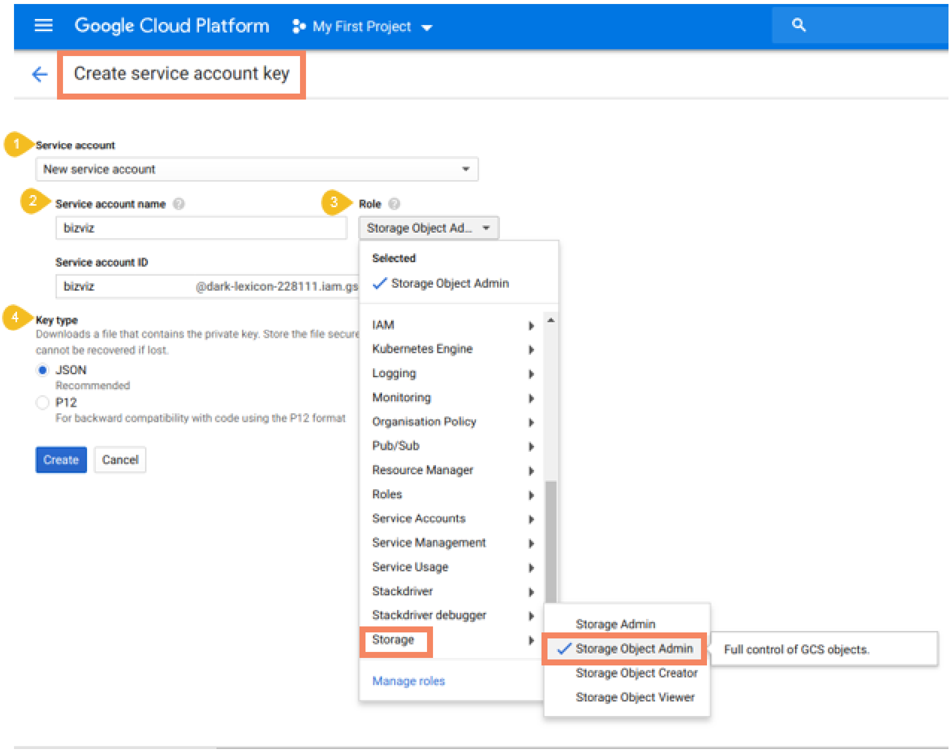
Help me choose
Asks a few questions to help you decide which type of credential to use

x) The '**Create Service Account Key**' page opens.

xi) The users need to provide the following parameters:

1. Services account: Select 'New Service Account' from the drop-down menu.
2. Service account name: Provide a user-defined name for the Service Account. (E.g., 'bizviz' in the below given image)
3. Role: Select the '**Storage'** option from the drop-down menu.
   a. A new context menu opens.
   b. Select **'Storage Object Admin'** from the context menu.
4. Key type: Select JSON by choosing the checkmark provided next to the option.

xii) The Service Account Key gets created.



Service account: New service account

Service account name: bizviz

Role: Storage Object Ad...

✓ Storage Object Admin

Service account ID: bizviz @dark-lexicon-228111.iam.gs

Key type
Downloads a file that contains the private key. Store the file secure cannot be recovered if lost.

● JSON
Recommended

○ P12
For backward compatibility with code using the P12 format

✓ Storage Object Admin — Full control of GCS objects.

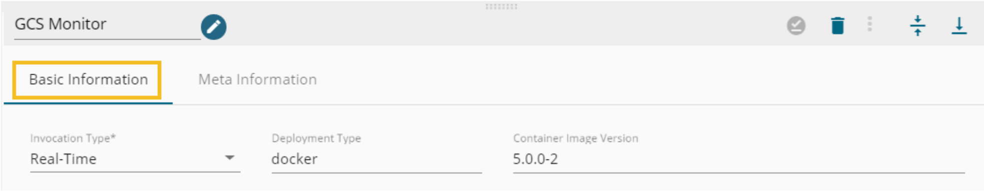
Note: After the successful creation of the Service Account, the users should get Credential JSON file name according to their project name it indicates that the Service Account Key file got successfully created.

8.5.3.2. GCS Monitor in Data Pipeline

GCS Monitor monitoring one folder all the time. If any file has been uploaded, the GCS Monitor reads the file name and sends it to the event. The GCS Monitor also copies that file to the copied (defined) location and deletes from the monitoring folder. It repeats the same process for all the files.

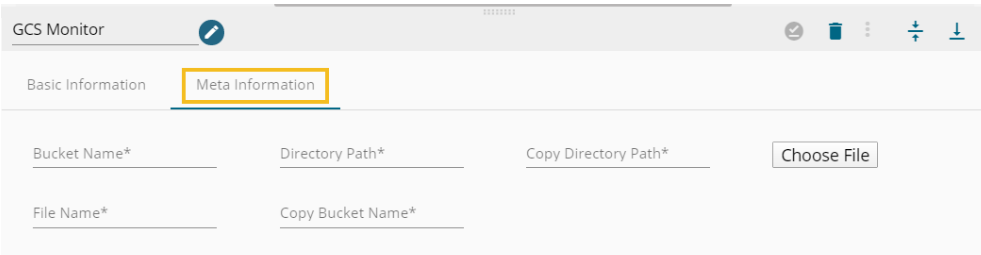
Steps to create one pipeline using the GCS Monitor component.

i) Drag and drop the GCS Monitor component to the Workflow Editor.
ii) Click the dragged GCS Monitor component to access the component properties.
iii) The **Basic Information tab** opens by default.
   a. Select an Invocation type from the drop-down menu to confirm the running mode of the reader component. The supported invocation type is ‘**Real-Time**’.
   b. Deployment Type: It displays the deployment type for the component. This field comes pre-selected.
   c. Container Image Version: It displays the image version for the docker container. This field comes pre-selected.

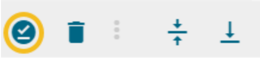


Invocation Type*: Real-Time
Deployment Type: docker
Container Image Version: 5.0.0-2

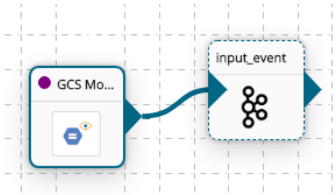
iv) Open the **Meta Information tab** and configure it.
1. Bucket Name: Enter the source bucket name
2. Directory Path: Fill monitor folder path using forward-slash (/). E.g., **monitor/**
3. Copy Directory Path: Fill the copy folder name where you want to copy the uploaded file. E.g., **monitor_copy/**
4. Choose file: Upload a **Service Account Key(s)** file.
5. File Name: After the Service Account Key file is uploaded, the file name gets auto-generated based on the uploaded file.
6. Copy Bucket Name: Fill the destination bucket name where you need to copy.



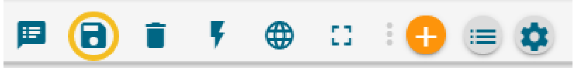
v) Click the ‘**Save Component in Storage’** icon to save all the Meta Information for the GCS Monitor component (A message appears to notify completion of the action).



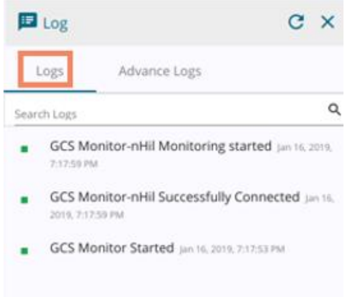
vi) The users need to create an event and connect it with the GCS Monitor component.
(Monitor sends filename to the event when any file gets uploaded on monitor folder.)



vii) Click the Update Pipeline icon (A message appears to notify completion of the action).

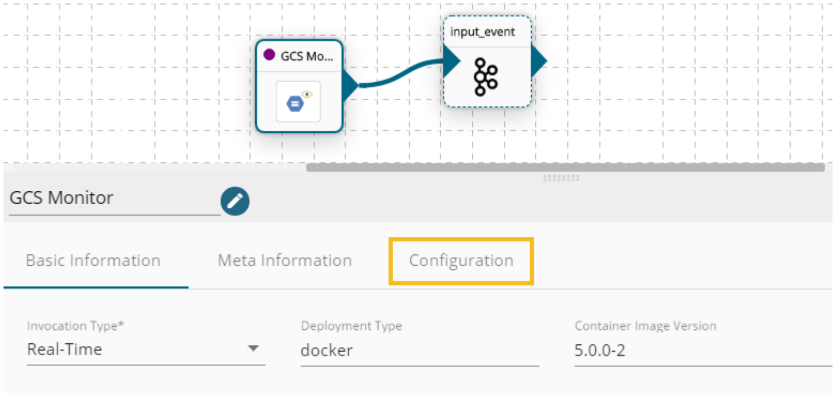


viii) Open the Log toggle panel to see the process of the GCS Monitor.



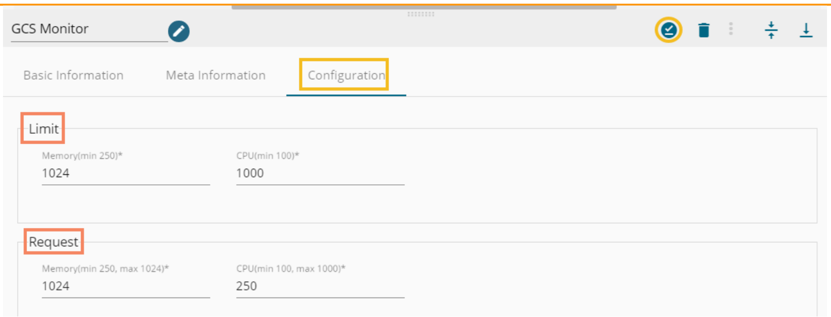
```
GCS Monitor-nHil Monitoring started Jan 16, 2019, 7:17:59 PM
GCS Monitor-nHil Successfully Connected Jan 16, 2019, 7:17:59 PM
GCS Monitor Started Jan 16, 2019, 7:17:53 PM
```

ix) Open the GCS Monitor component after updating the pipeline.
x) The Configuration tab appears below.



Invocation Type*: Real-Time
Deployment Type: docker
Container Image Version: 5.0.0-2

xi) Modify the configuration values (if needed).
xii) Click the '**Save Component in Storage**' icon.



Limit

Memory(min 250)*: 1024

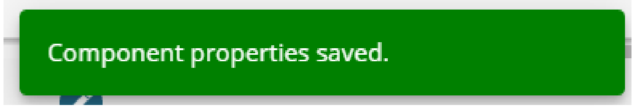
CPU(min 100)*: 1000

Request

Memory(min 250, max 1024)*: 1024

CPU(min 100, max 1000)*: 250

xiii) The GCS Monitor component gets configured and a notification message appears to confirm the action.



Component properties saved.

xiv) The GCS Monitor component can be used in a pipeline workflow.

Note: The GCS monitor can work with GCS Reader. BDB Pipeline supports CSV and JSON files at present.

8.5.4. AWS SNS Monitor

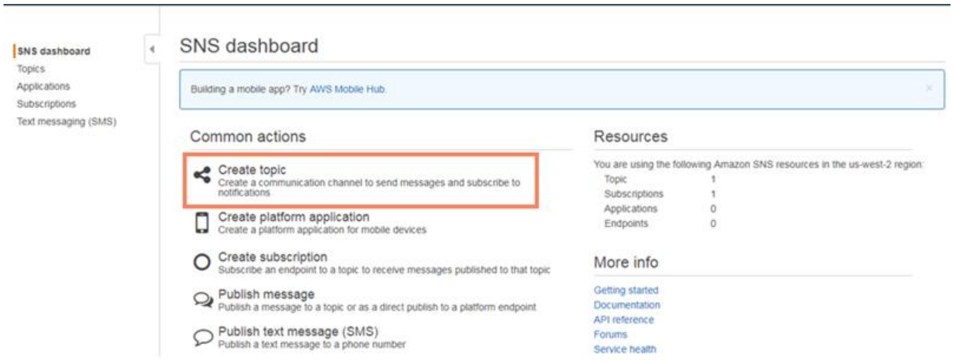
The user needs to create an SNS topic as a precondition for SNS Monitor in BDB Data Pipeline.

8.5.4.1. Steps to Configure SNS and SQS for SNS Monitor

i) Log in to an AWS account.

ii) Navigate to the SNS console using the below given link:
https://us-west-2.console.aws.amazon.com/sns/v2/home?region=us-west-2#/home

iii) Select the '**Create topic**' option.



SNS dashboard

Building a mobile app? Try AWS Mobile Hub.

Common actions

Create topic
Create a communication channel to send messages and subscribe to notifications

Create platform application
Create a platform application for mobile devices

Create subscription
Subscribe an endpoint to a topic to receive messages published to that topic

Publish message
Publish a message to a topic or as a direct publish to a platform endpoint

Publish text message (SMS)
Publish a text message to a phone number

Resources

You are using the following Amazon SNS resources in the us-west-2 region:

Topic	1
Subscriptions	1
Applications	0
Endpoints	0

More info

Getting started
Documentation
API reference
Forums
Service health

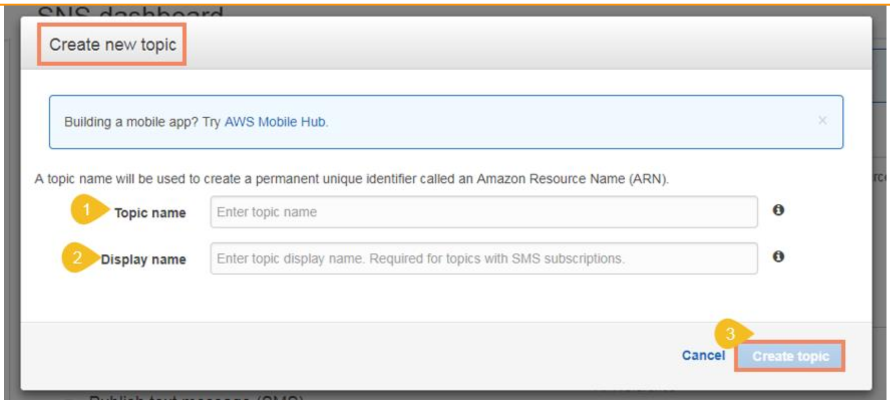
iv) The '**Create new topic**' window opens.

a. Provide the following information:

i. Topic Name: Enter a name for the topic

ii. Display Name: Enter the topic name (Required for topics with SMS subscription)

iii. Click the '**Create topic**' option.



Building a mobile app? Try AWS Mobile Hub.

A topic name will be used to create a permanent unique identifier called an Amazon Resource Name (ARN).

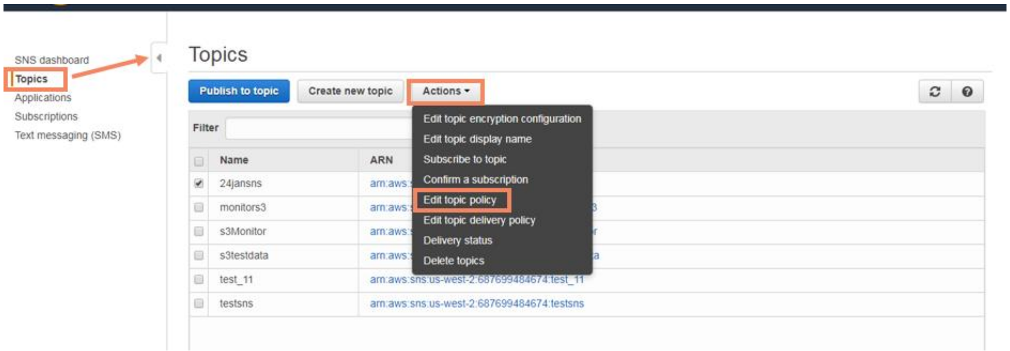
- Topic name: Enter topic name
- Display name: Enter topic display name. Required for topics with SMS subscriptions.

v) Assign permission after the topic gets created.

vi) Navigate to the ‘**Topics**’ page.

vii) Click the ‘**Actions**’ drop-down menu.

viii) Select the ‘**Edit topic policy**’ option from the menu.

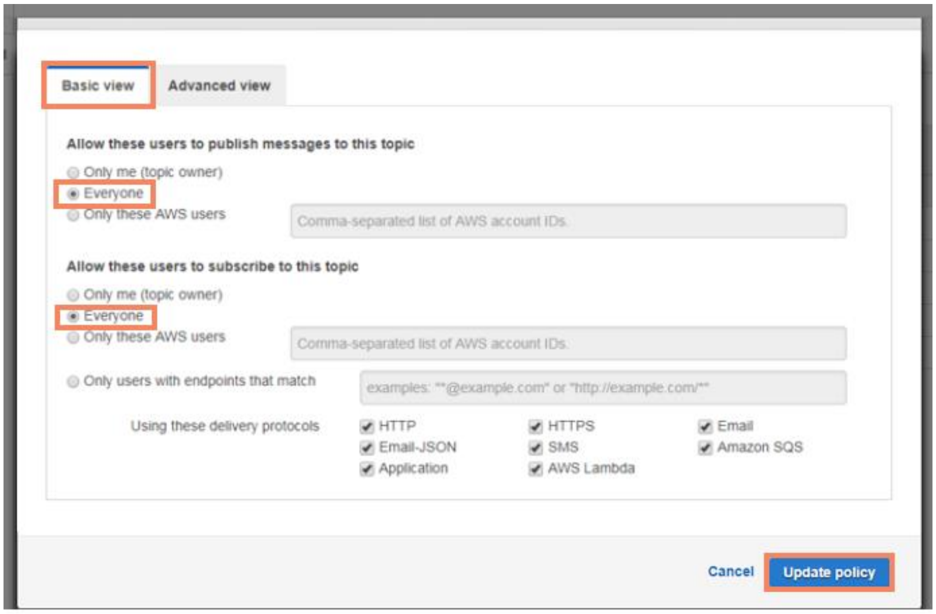


	Name	ARN
☑	24jansns	arn:aws:
□	monitors3	arn:aws:
□	s3Monitor	arn:aws:
□	s3testdata	arn:aws:
□	test_11	arn:aws:sns:us-west-2:687699484674:test_11
□	testsns	arn:aws:sns:us-west-2:687699484674:testsns

ix) The ‘**Basic view**’ tab opens.

x) Update the policy from ‘**Only me**’ to ‘**Everyone**’ using the radio button.

xi) Click the ‘**Update policy**’ option.



Allow these users to publish messages to this topic

○ Only me (topic owner)
● Everyone
○ Only these AWS users: Comma-separated list of AWS account IDs.

Allow these users to subscribe to this topic

○ Only me (topic owner)
● Everyone
○ Only these AWS users: Comma-separated list of AWS account IDs.
○ Only users with endpoints that match: examples: "*@example.com" or "http://example.com/*"

Using these delivery protocols: ☑ HTTP ☑ HTTPS ☑ Email ☑ Email-JSON ☑ SMS ☑ Amazon SQS ☑ Application ☑ AWS Lambda

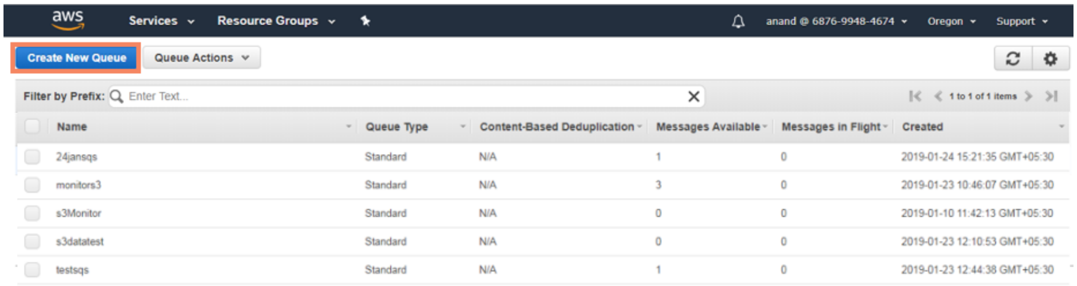
8.5.4.2. Create the SQS Queue

The steps to create an SQS Queue are described below:

i) Log in to an AWS account.

ii) Navigate to the SQS console.
https://console.aws.amazon.com/sqs/home?region=us-west-2#queue-browser:selected=https://sqs.us-west-2.amazonaws.com/687699484674/s3Monitor;prefix=

iii) Click the ‘**Create New Queue**’ option.



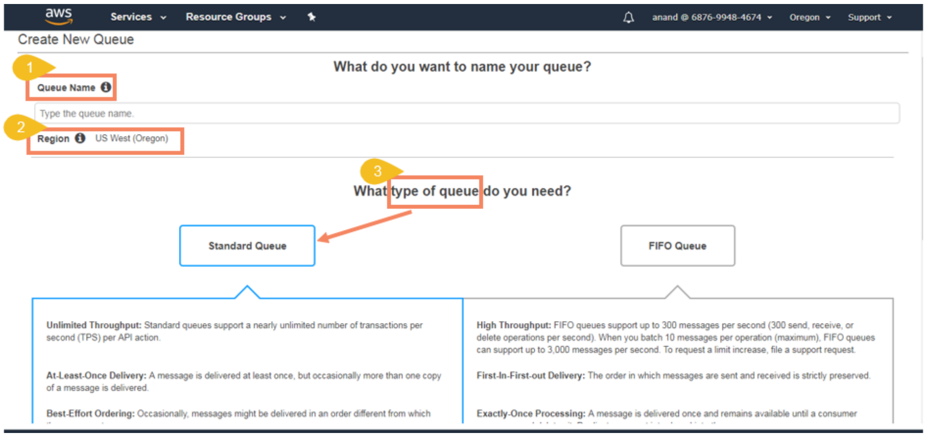
Name	Queue Type	Content-Based Deduplication	Messages Available	Messages in Flight	Created
24jansqs	Standard	N/A	1	0	2019-01-24 15:21:35 GMT+05:30
monitors3	Standard	N/A	3	0	2019-01-23 10:46:07 GMT+05:30
s3Monitor	Standard	N/A	0	0	2019-01-10 11:42:13 GMT+05:30
s3datatest	Standard	N/A	0	0	2019-01-23 12:10:53 GMT+05:30
testsqs	Standard	N/A	1	0	2019-01-23 12:44:38 GMT+05:30

iv) The ‘**Create New Queue**’ page opens.

a. Provide a ‘**Queue Name**’ in the given space.

b. Provide the ‘**Region**’.

c. Select the type of queue from the provided options: **Standard Queue** or **FIFO Queue.**



Create New Queue

What do you want to name your queue?

Queue Name: Type the queue name.

Region: US West (Oregon)

What type of queue do you need?

Unlimited Throughput: Standard queues support a nearly unlimited number of transactions per second (TPS) per API action.

At-Least-Once Delivery: A message is delivered at least once, but occasionally more than one copy of a message is delivered.

Best-Effort Ordering: Occasionally, messages might be delivered in an order different from which

High Throughput: FIFO queues support up to 300 messages per second (300 send, receive, or delete operations per second). When you batch 10 messages per operation (maximum), FIFO queues can support up to 3,000 messages per second. To request a limit increase, file a support request.

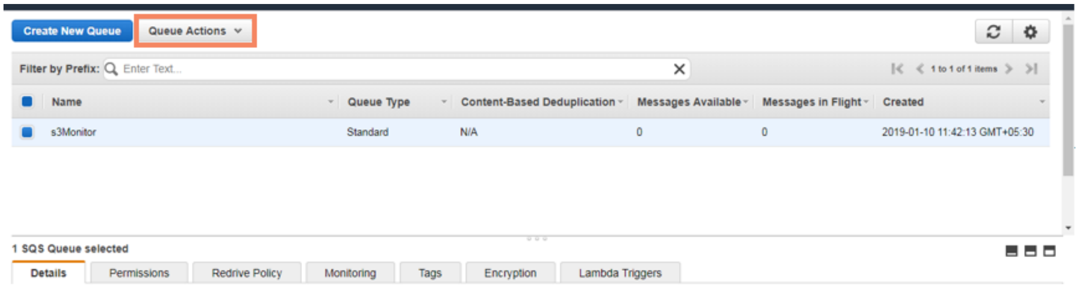
First-In-First-out Delivery: The order in which messages are sent and received is strictly preserved.

Exactly-Once Processing: A message is delivered once and remains available until a consumer

v) The users need to subscribe to the queue to an SNS topic.

vi) Navigate to the ‘**Queue Actions**’ drop-down menu.

vii) Select the ‘**Subscribe Queue to SNS topic**’ option.



Name	Queue Type	Content-Based Deduplication	Messages Available	Messages in Flight	Created
s3Monitor	Standard	N/A	0	0	2019-01-10 11:42:13 GMT+05:30

1 SQS Queue selected

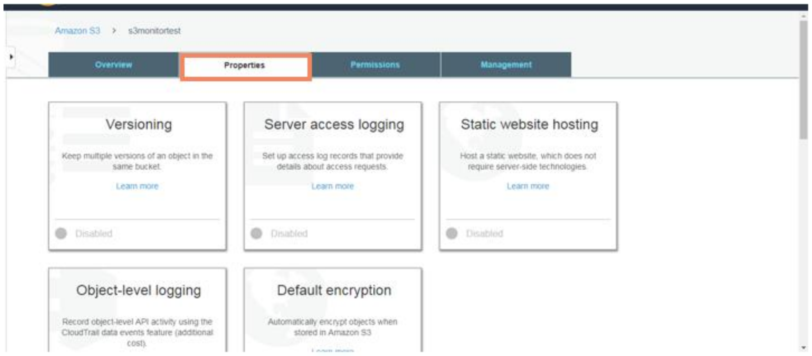
Note: Once the subscription is done with our SNS topic, the users need to create the event in S3 so that the pipeline can receive the uploaded message that comes to the S3 bucket.

8.5.4.3. Creating an S3 Event

i) Log in to an AWS account.

ii) Create the S3 bucket.

iii) Select the ‘**Properties**’ tab.



Amazon S3 > s3monitortest

Overview | Properties | Permissions | Management

Versioning
Keep multiple versions of an object in the same bucket.
Learn more
Disabled

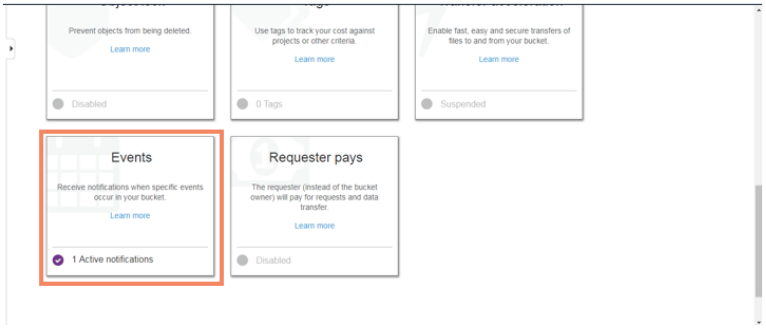
Server access logging
Set up access log records that provide details about access requests.
Learn more
Disabled

Static website hosting
Host a static website, which does not require server-side technologies.
Learn more
Disabled

Object-level logging
Record object-level API activity using the CloudTrail data events feature (additional cost).

Default encryption
Automatically encrypt objects when stored in Amazon S3

iv) Select the ‘**Events**’ option under properties.



Prevent objects from being deleted.
Learn more
Disabled

Use tags to track your cost against projects or other criteria.
Learn more
0 Tags

Enable fast, easy and secure transfers of files to and from your bucket.
Learn more
Suspended

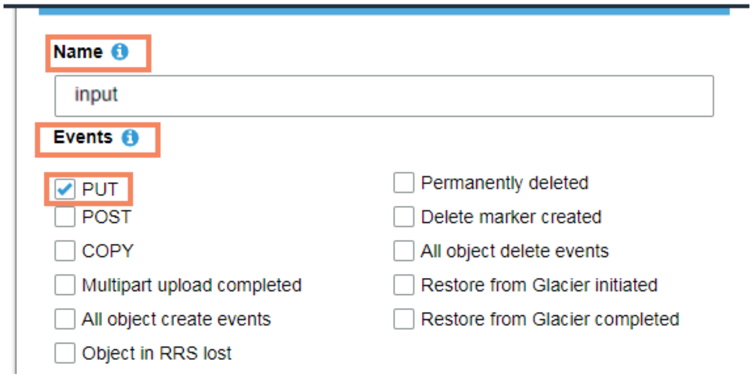
Events
Receive notifications when specific events occur in your bucket.
Learn more
1 Active notifications

Requester pays
The requester (instead of the bucket owner) will pay for requests and data transfer.
Learn more
Disabled

v) Create a new Event for the respective S3 bucket by providing the following information:

a. Provide ‘**Name**’ for the Event.

b. Select an option using the checkmark in the given box from the provided list of **Events.**



Name
input

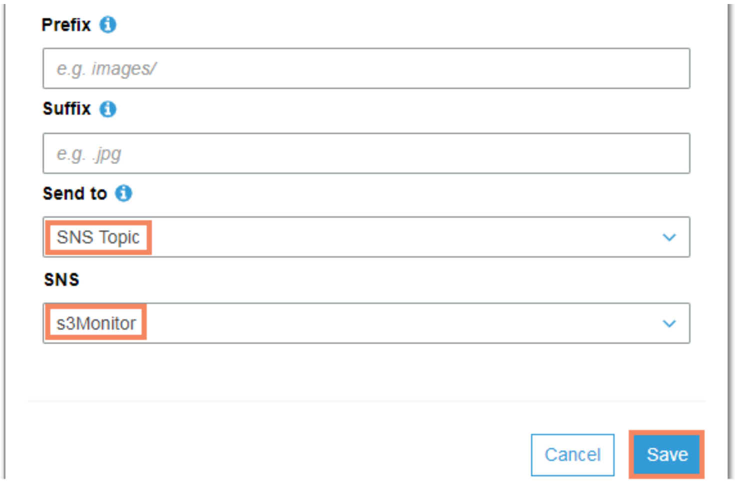
Events

☑ PUT
☐ POST
☐ COPY
☐ Multipart upload completed
☐ All object create events
☐ Object in RRS lost
☐ Permanently deleted
☐ Delete marker created
☐ All object delete events
☐ Restore from Glacier initiated
☐ Restore from Glacier completed

c. Select the **SNS topic** as an option from the ‘**Send to’** drop-down menu.

d. Select the ‘**S3 Monitor**’ from the SNS drop-down menu.

e. Click the '**Save**' option.


Prefix

e.g. images/

Suffix

e.g. .jpg

Send to

SNS Topic

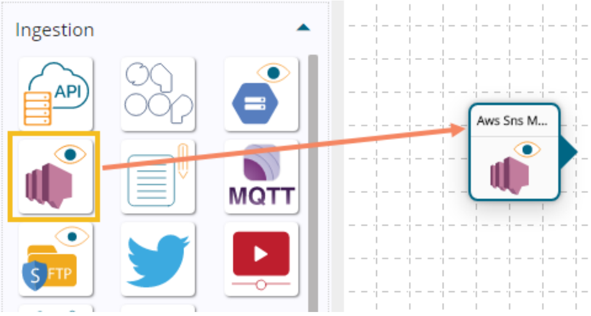
SNS

s3Monitor

After the Event gets created, the users can access the complete setup to use the Data Pipeline SNS Monitor to monitor the S3 location.

8.5.4.4. Using the AWS SNS Monitor Component in the Data Pipeline

i) Drag and drop the AWS SNS Monitor component to the Workflow Editor.



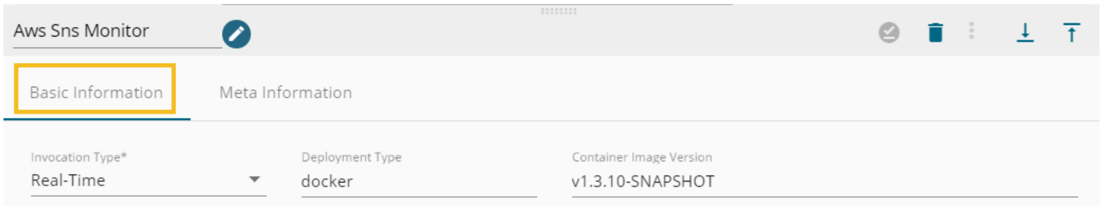
ii) Click on the component to get the component properties tabs below.

iii) The **Basic Information tab** opens by default.

a. Select an Invocation type from the drop-down menu to confirm the running mode of the reader component. The supported invocation type is '**Real-Time**'.

b. Deployment Type: It displays the deployment type for the component. This field comes pre-selected.

c. Container Image Version: It displays the image version for the docker container. This field comes pre-selected.


Aws Sns Monitor

Invocation Type*: Real-Time

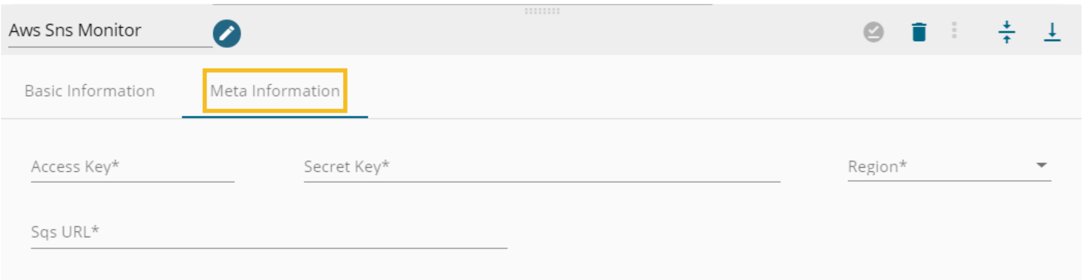
Deployment Type: docker

Container Image Version: v1.3.10-SNAPSHOT

iv) Open the '**Meta Information**' **tab** and provide all the required details:

a. Access Key

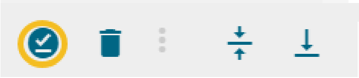
b. Secret Key
c. Region
d. SQS URL


Aws Sns Monitor

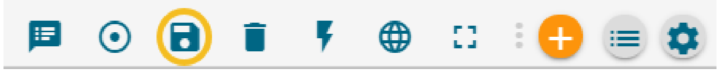
Basic Information | Meta Information

Access Key*
Secret Key*
Region*
Sqs URL*

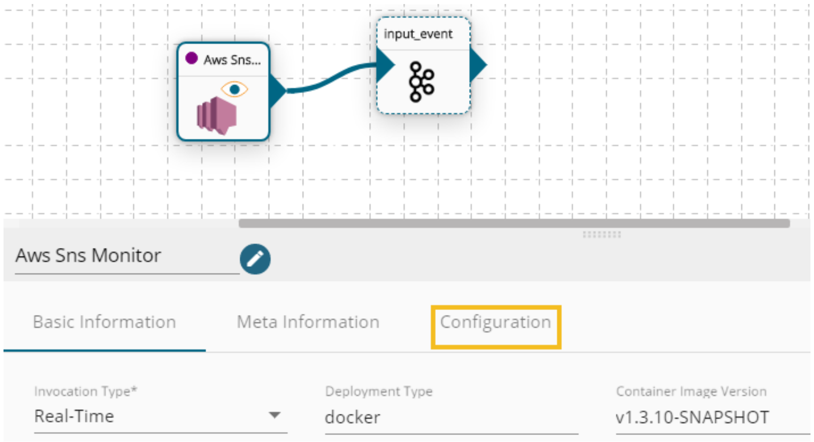
v) Click the '**Save Component in Storage**' icon to save the configuration details (A notification message appears to confirm the action).



vi) Click the '**Update Pipeline**' icon. (A notification message appears to confirm the action).



vii) Connect the AWS SNS Monitor to an input event.
viii) Open the AWS SNS Monitor component again.
ix) The **Configuration tab** appears below.

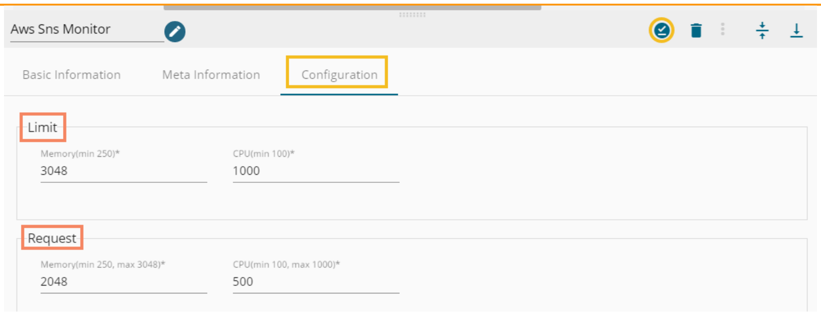


Aws Sns Monitor

Basic Information | Meta Information | Configuration

Invocation Type*: Real-Time
Deployment Type: docker
Container Image Version: v1.3.10-SNAPSHOT

x) Modify the Configuration values (if needed).
xi) Click the '**Save Component in Storage**' icon.


Aws Sns Monitor

Limit

Memory(min 250)*: 3048

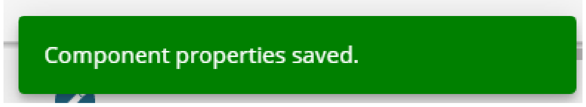
CPU(min 100)*: 1000

Request

Memory(min 250, max 3048)*: 2048

CPU(min 100, max 1000)*: 500

xii) The AWS SNS Monitor component gets configured and a notification message appears to confirm.


Component properties saved.

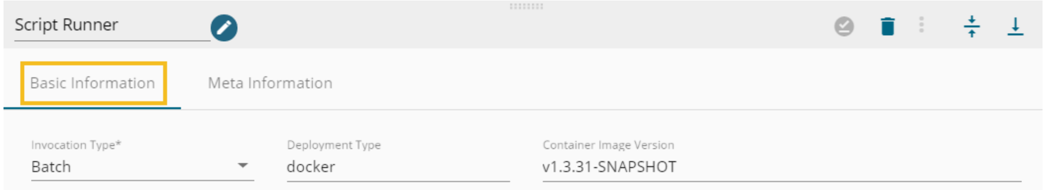
xiii) The AWS SNS Monitor component can be used in a pipeline workflow now.

8.5.5. Script Runner

This component is can be used for connecting it to a remote server/machine and running script files

present there based on some events.

i) Drag and drop Script Runner Component to the Workflow Editor.
ii) Open the dragged Script Runner component to open the component properties tabs.
iii) The **Basic Information tab** opens by default.
   a. Select an Invocation type from the drop-down menu to confirm the running mode of the reader component. The supported invocation types are ‘**Real-Time**’ and ‘**Batch**’.
   b. Deployment Type: It displays the deployment type for the component. This field comes pre-selected.
   c. Container Image Version: It displays the image version for the docker container. This field comes pre-selected.


Script Runner

Invocation Type*: Batch

Deployment Type: docker

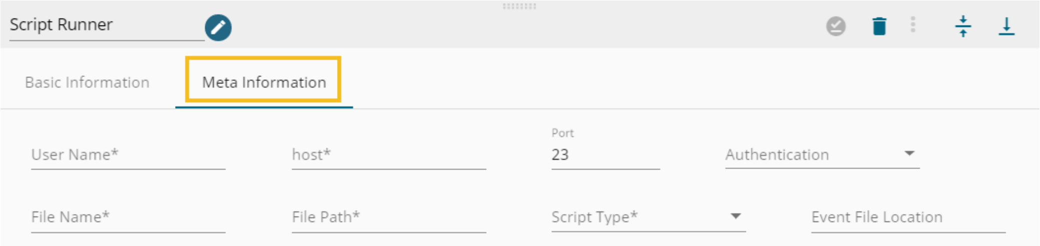
Container Image Version: v1.3.31-SNAPSHOT

iv) Open the **Meta Information tab** and configure the required information.
   a. **Host:** Host IP of the remote server/machine
   b. **Username:** Username of the remote server/machine.
   c. **Port:** Provide machine Port number.
   d. **Authentication:** Select an authentication option from the drop-down menu.
      1. Password: By selecting this option the user needs to pass the password.
      2. PEM/PPK File: By selecting this option the user needs to pass the authentication file to connect to the server.
   e. **Script type**: Choose the type of script file that You want to run out of SSH/ PERL/command options.

f. **File path:** Path of the file that is stored at the remote server.

g. **File Name:** The script file that you want to execute.

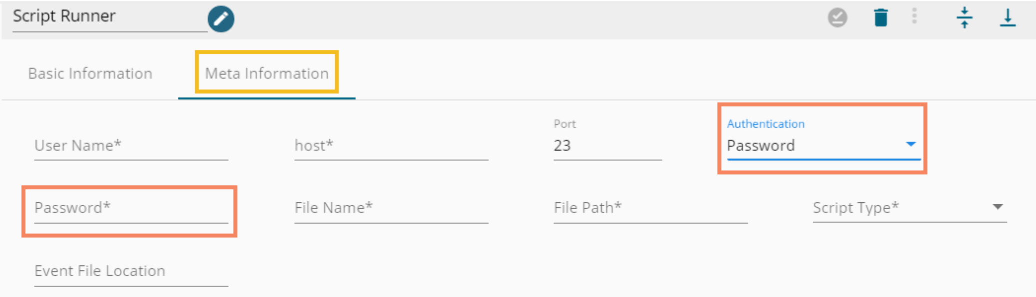
h. **Event File Location:** this is the location of the file sent through file monitor (Non-mandatory).



Port: 23

Note: The displayed fields may vary based on the selected Authentication option.

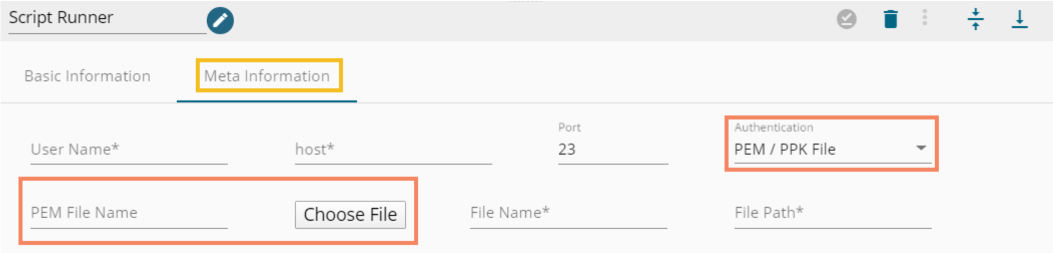
Component Properties when the Authentication option is Password.



Port: 23

Authentication: Password

Component Properties when the Authentication option is PEM/PPK File.

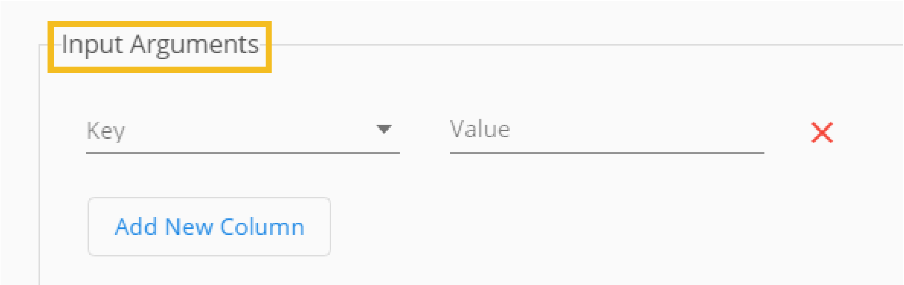


Port: 23

Authentication: PEM / PPK File

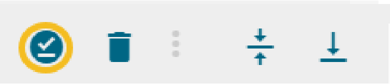
i. **Input Arguments**

1. Manual Arguments (Optional): These are the arguments to the parameter of the script that the user can provide manually.
2. Event Arguments (Optional): These are the arguments to the parameter coming from the previous event/Kafka-topic.

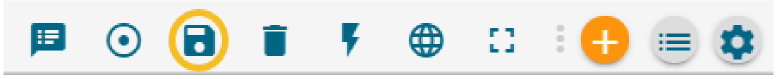


Note: The user need not do anything if the arguments to be passed are coming from the previous event. By selecting Even Arguments, the value is set to true by default.

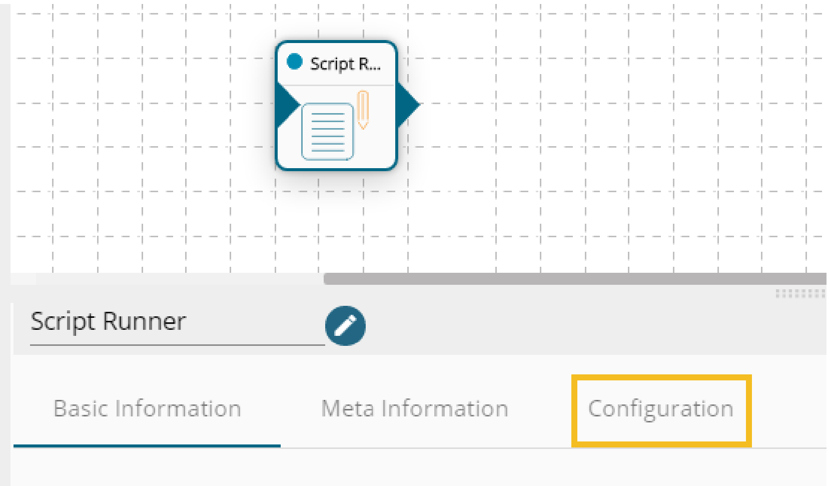
v) Click the '**Save Component in Storage**' icon (A notification message appears to confirm the action completion).



vi) Click the '**Update Pipeline**' icon (A notification message appears to confirm the action completion).

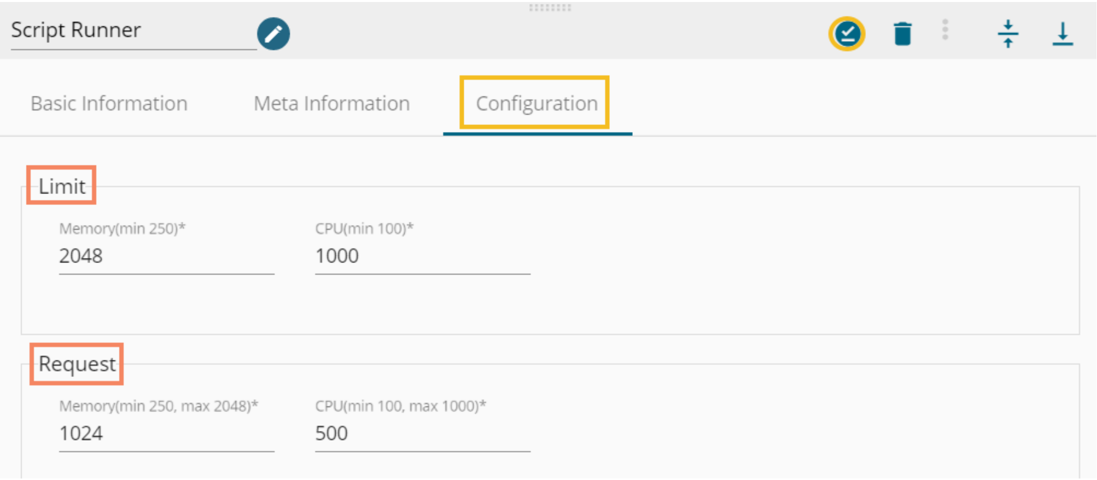


vii) Open the Script Runner component again to see the **Configuration tab** below.



viii) Modify the values for the Configuration tab (if needed).

ix) Click the '**Save Component in Storage**' icon.



Limit

Memory(min 250)*
2048

CPU(min 100)*
1000

Request

Memory(min 250, max 2048)*
1024

CPU(min 100, max 1000)*
500

x) The Script Runner component gets configured and the notification message appears to inform the same.

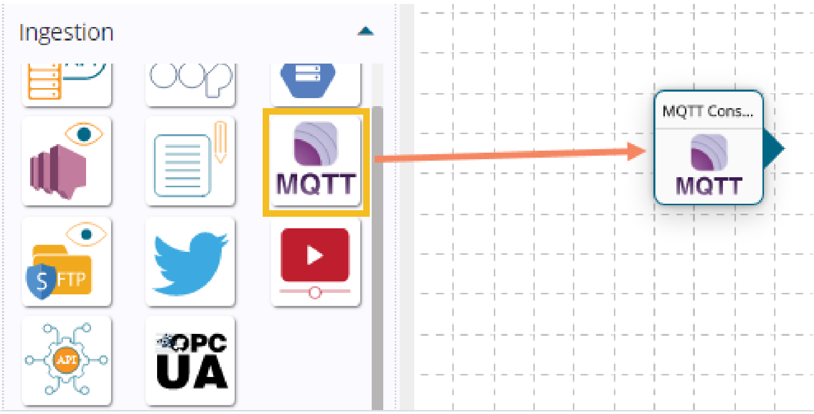
Note: The component can connect to the remote machine using the details provided. It will pick the file from the location in that machine using the file name and file path respectively and finally execute the script after passing arguments (if any).

- **Limitations**
  a. It accepts only lists as input i.e. the in-event data should be a list.
  b. It sends data on the out-event only when there is a print statement as output in the script if not there will be no data on the out event.
  c. The data produced from the script is of a list type.

8.5.6. MQTT Consumer

The MQTT consumer is instrumental in consuming data from IoT devices. It consumes data from provided MQTT broker and ingests the data in our data pipeline for further processing.

i) Drag and drop the MQTT Consumer to the Workflow Editor.



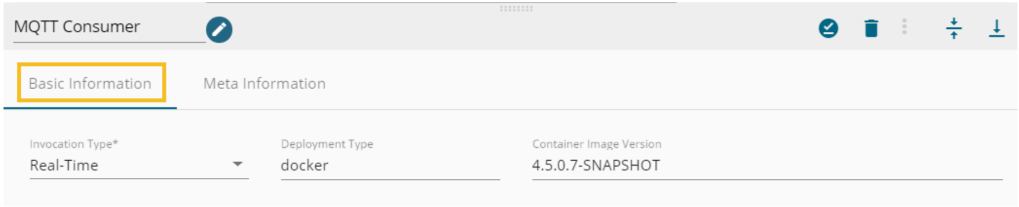
ii) Click the dragged MQTT Consumer component to get the component properties tabs.

iii) The **Basic Information tab** opens by default.

a. Select the invocation type (at present only '**Real-Time**' option is provided)

b. Deployment Type: It comes preselected based on the component.

c. Container Image Version: It comes preselected based on the component.



MQTT Consumer

Basic Information | Meta Information

Invocation Type*: Real-Time

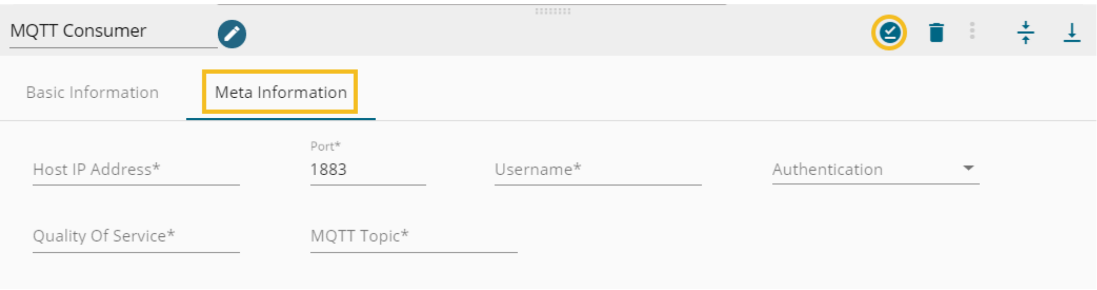
Deployment Type: docker

Container Image Version: 4.5.0.7-SNAPSHOT

iv) Click on the **Meta Information tab** to open the properties fields.

a. Host: Broker IP or URL
b. Username: If authentication required then give username
c. Broker Port Number
d. Authentication: Select an authentication option using the drop-down list.
   i. Password: Provide a password to authenticate the MQTT component.
   ii. PEM/PPK File: Choose a file to authenticate the MQTT component
e. Quality of Service: The Quality of Service (QOS) level is an agreement between the sender of a message and the receiver of a message that defines the guarantee of delivery for a specific message.
f. MQTT Topic: Provide MQTT topic name where the user is producing data.

v) Click the '**Save Component in Storage**' icon (A notification message appears to confirm the action).



MQTT Consumer

Basic Information | Meta Information

Host IP Address*

Port* 1883

Username*

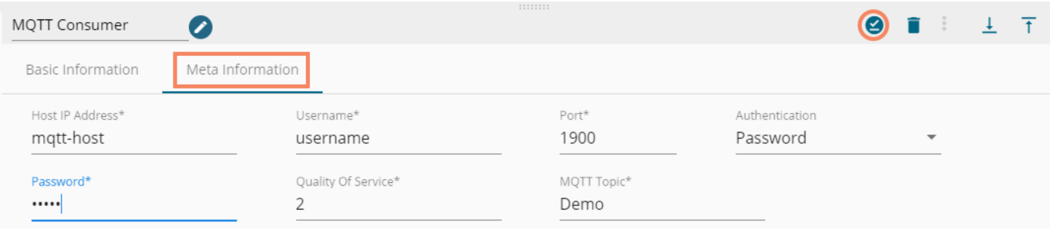
Authentication

Quality Of Service*

MQTT Topic*

Note: The displayed fields may vary based on the selected Authentication option.

Configuration Fields when the Authentication option is Password



MQTT Consumer

Basic Information | Meta Information

Host IP Address* mqtt-host

Username* username

Port* 1900

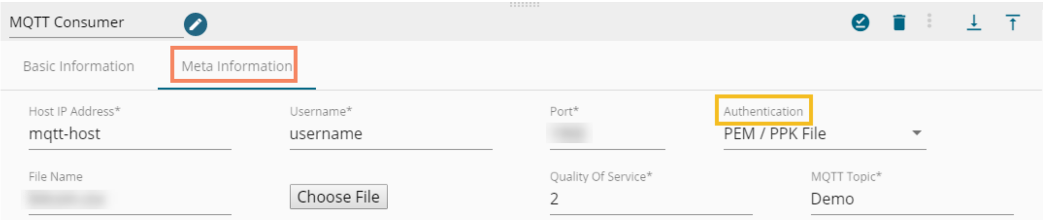
Authentication Password

Password* •••••

Quality Of Service* 2

MQTT Topic* Demo

Configuration Fields when the Authentication option is PEM/PPK File



MQTT Consumer

Basic Information | Meta Information

Host IP Address* mqtt-host

Username* username

Port* [redacted]

Authentication PEM / PPK File

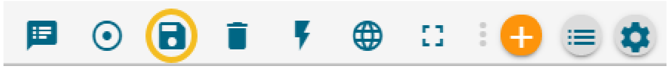
File Name [redacted]

Choose File

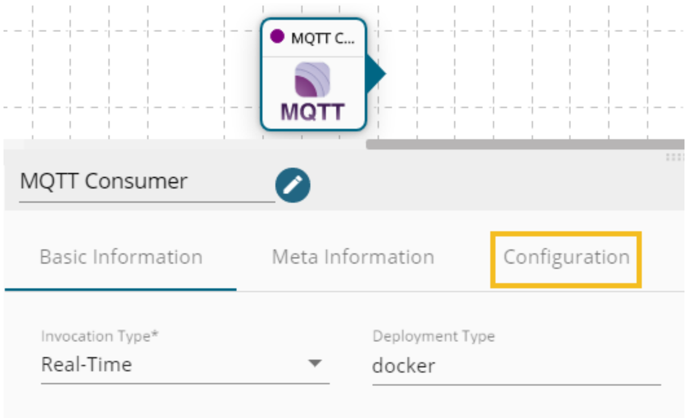
Quality Of Service* 2

MQTT Topic* Demo

vi) Click the '**Update Pipeline**' icon (A notification message appears to confirm the action completion).



vii) Open the MQTT component again to see the **Configuration tab** below.



MQTT Consumer

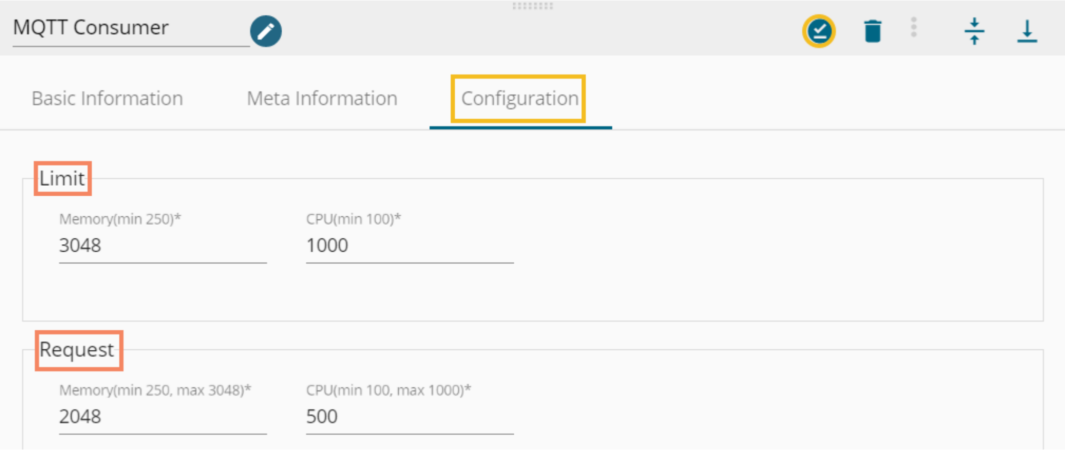
Basic Information | Meta Information | Configuration

Invocation Type* Real-Time

Deployment Type docker

viii) Modify the values for the Configuration tab (if needed).

ix) Click the '**Save Component in Storage**' icon.



MQTT Consumer

Limit

Memory(min 250)*: 3048

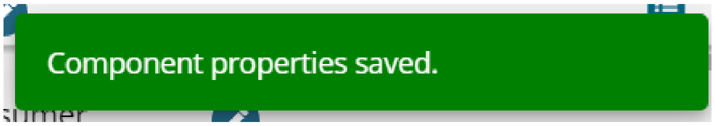
CPU(min 100)*: 1000

Request

Memory(min 250, max 3048)*: 2048

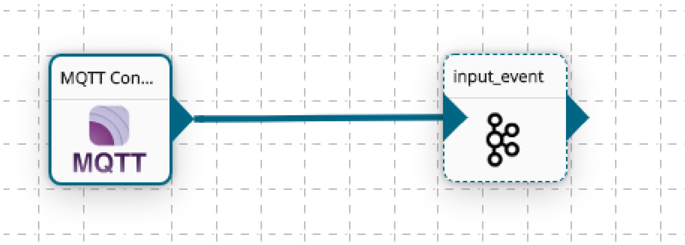
CPU(min 100, max 1000)*: 500

x) The MQTT gets configured and the notification message appears to inform the same.



Component properties saved.

Note: The MQTT Consumer should be connected to an input event to pass the data and create a Pipeline workflow.



8.5.7. FTP Python Monitor

i) Drag and Drop the SFTP Python Monitor ingestion component to the Workflow Editor.

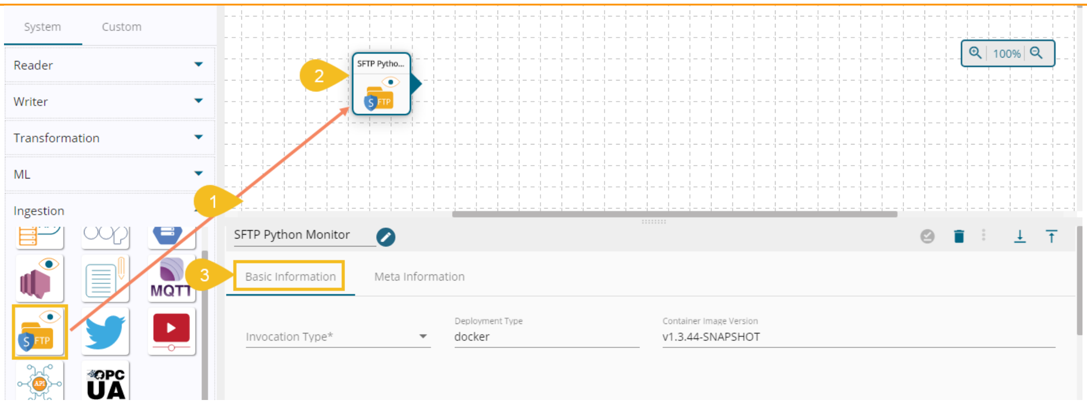
ii) Click the dragged ingestion component to get the component properties tabs.

iii) Configure the **Basic Information Tab**.

a. Select the invocation type (at present only '**Real-Time**' option is provided).

b. Deployment Type: It comes preselected based on the component.

c. Container Image Version: It comes preselected based on the component.



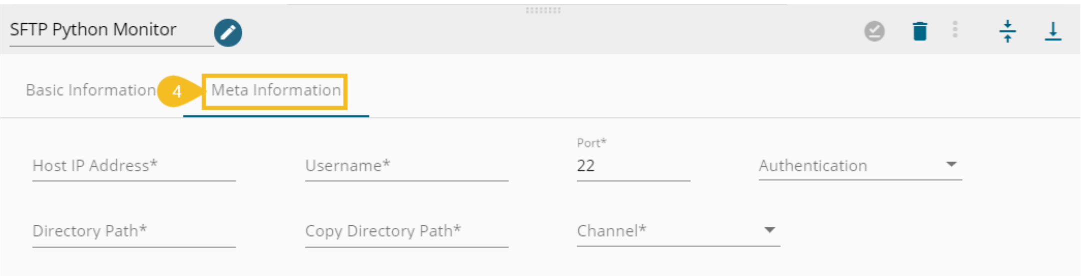
Invocation Type*

Deployment Type: docker

Container Image Version: v1.3.44-SNAPSHOT

iv) Configure the **Meta Information tab** for the dragged SFTP Python Monitor component.

a. Host: Broker IP or URL
b. Username: If authentication required then give username
c. Port: Provide the Port number
d. Authentication: Select an authentication option using the drop-down list.
    i. Password: Provide a password to authenticate the MQTT component.
    ii. PEM/PPK File: Choose a file to authenticate the MQTT component
e. Directory Path: Fill monitor folder path using forward-slash (/). E.g., **monitor/**
f. Copy Directory Path: Fill the copy folder name where you want to copy the uploaded file. E.g., **monitor_copy/**
g. Channel: Select a channel option from the drop-down menu (the supported channel is SFTP).



Host IP Address*

Username*

Port*: 22

Authentication

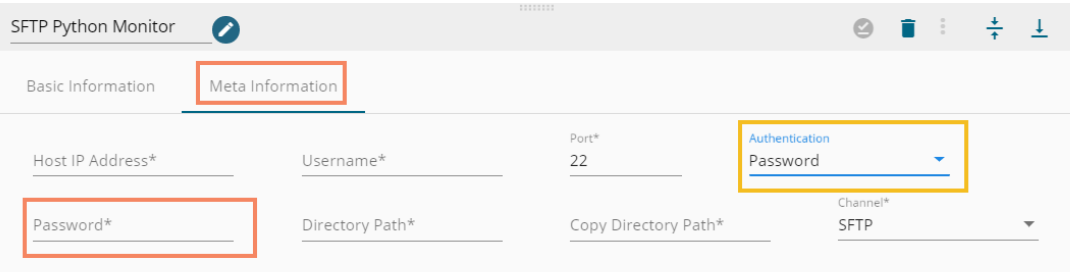
Directory Path*

Copy Directory Path*

Channel*

Note: The SFTP Python Monitor may have Meta Information fields based on the selected Authentication option.

Meta Information Fields when the Authentication option is Password



Host IP Address*

Username*

Port*: 22

Authentication: Password

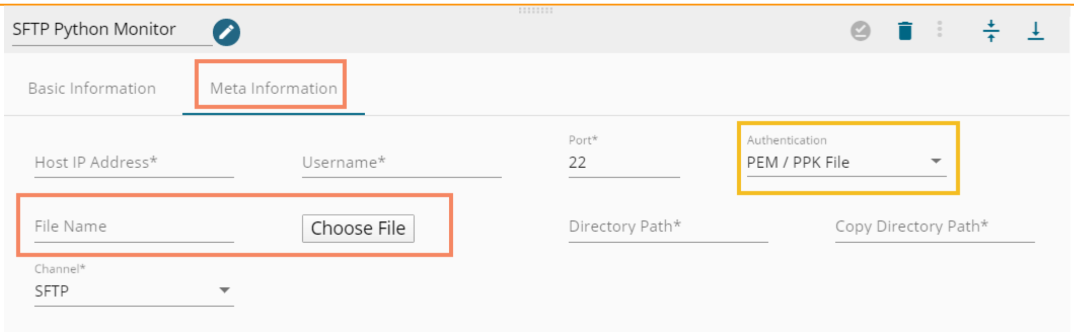
Password*

Directory Path*

Copy Directory Path*

Channel*: SFTP

Meta Information Fields when the Authentication option is PEM/PPK


SFTP Python Monitor

Host IP Address*

Username*

Port* 22

Authentication PEM / PPK File

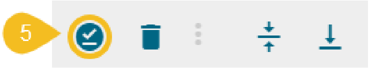
File Name Choose File

Directory Path*

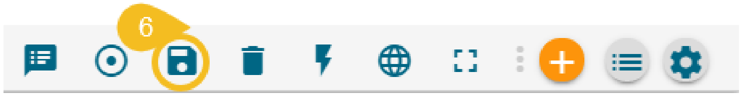
Copy Directory Path*

Channel* SFTP

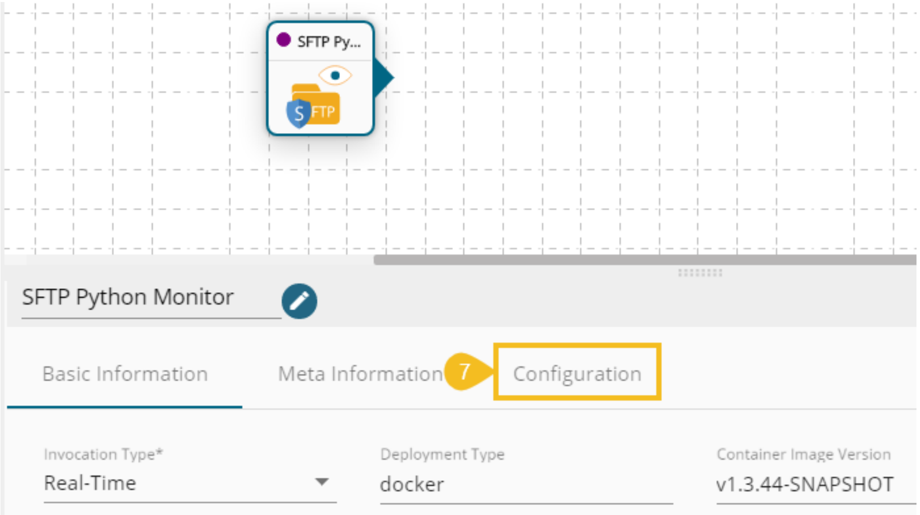
v) Save configured details of the SFTP Python Monitor component (A notification message appears to confirm the action).



vi) Click the '**Update Pipeline**' icon (A notification message appears to confirm the action).



vii) Open the SFTP Python Monitor component again to see the **Configuration tab** below.



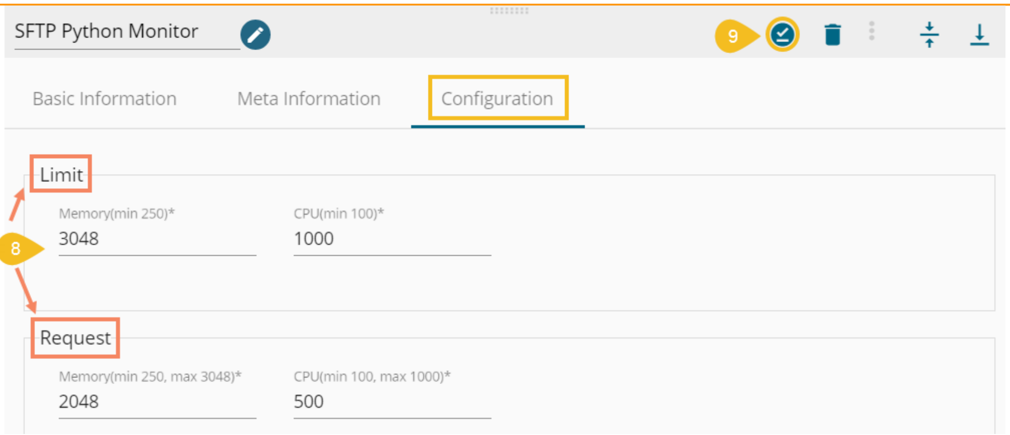
Invocation Type* Real-Time

Deployment Type docker

Container Image Version v1.3.44-SNAPSHOT

viii) Modify the Configuration values (if needed).

ix) Click the '**Save Component in Storage**' icon.


SFTP Python Monitor

Limit

Memory(min 250)*: 3048

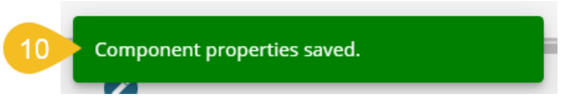
CPU(min 100)*: 1000

Request

Memory(min 250, max 3048)*: 2048

CPU(min 100, max 1000)*: 500

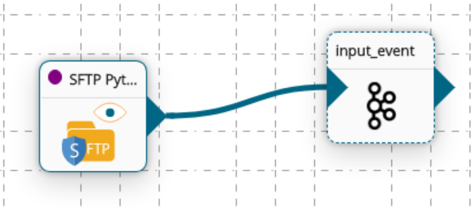
x) The SFTP Python Monitor component properties get saved and a notification message appears to confirm the same.



Component properties saved.

Note:

a. The SFTP Python Monitor component monitors the file coming to the monitored path and copies the file in Copy Path location for SFTP Reader to read.
b. The SFTP Python Monitor component requires an Event to send output.



8.5.8. Twitter Scrapper

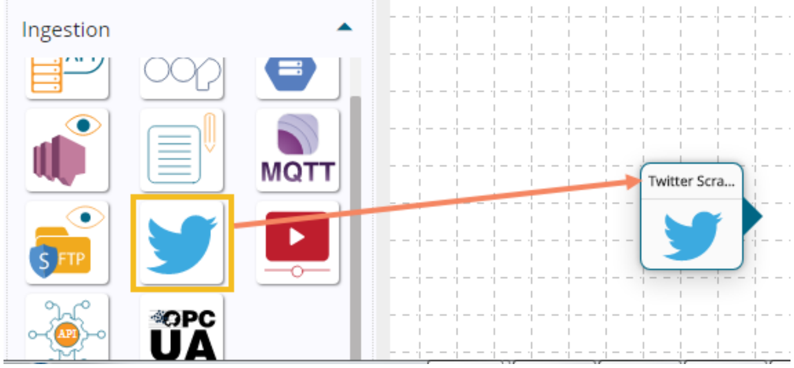
Twitter Scrapper helps in scrapping Twitter data. Twitter Scrapper provides two types of scrapping:

1. Real-Time
2. History Data

Real-Time – By using this option, the user can fetch the live Tweets by using twitter API along with specific search filters.

History Data – By using this option, the user can provide or fetch twitter data of the past 7 days (provided limit is up to 1000 tweets). It also provides us a facility to fetch data according to specific tags.

i) Drag them and drop Twitter Scrapper to the Workflow Editor.



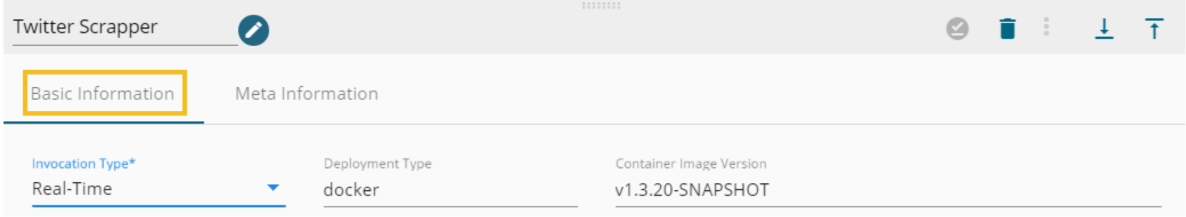
ii) Click the dragged Twitter Scrapper component to get the component properties tabs.

iii) The **Basic Information tab** opens by default.

   a. Select the invocation type (at present only '**Real-Time**' option is provided)

   b. Deployment Type: It comes preselected based on the component.

   c. Container Image Version: It comes preselected based on the component.



Invocation Type*: Real-Time
Deployment Type: docker
Container Image Version: v1.3.20-SNAPSHOT

iv) Click on the **Meta Information tab** to open the configuration fields for the selected columns.

   a. Consumer API Key: This is always is going to be unique depending upon the user's account. It is essential in both the cases of scrapping Real-time and History Data.

   b. Consumer API Secret Key: This is always is going to be unique depending upon the user's account. It is essential in both the cases of scrapping Real-time and History Data.

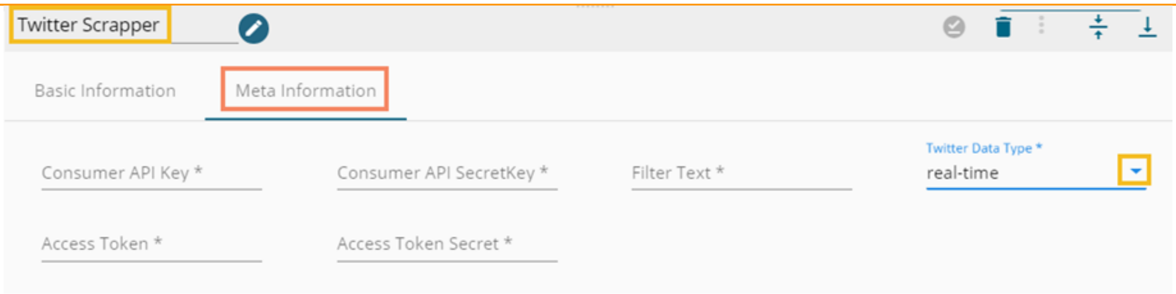
   c. Filter Text: This field is used to specify separate tags which user wants to fetch from a twitter. It provides the choice to the user for fetching multiple tags data at a time, to accomplish this user needs to provide tags name with a comma separator. For instance, #India, #Cricket

   d. Twitter Data Type: This field provides a dropdown menu from where the consumer can select the scrapping strategy(real-time/history).

   e. Access Token: This field is consumable only when the user wants to fetch real-time data from a scrapper.

   f. Access Token Secret: This field is consumable only when the user wants to fetch real-time data from a scrapper.

   Note: The Access Token and the Access Token Secret (key) fields appear only when the selected Twitter Data Type is 'real-time'.

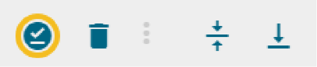

Twitter Scrapper

Basic Information | Meta Information

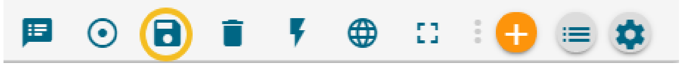
Consumer API Key * | Consumer API SecretKey * | Filter Text * | Twitter Data Type *: real-time

Access Token * | Access Token Secret *

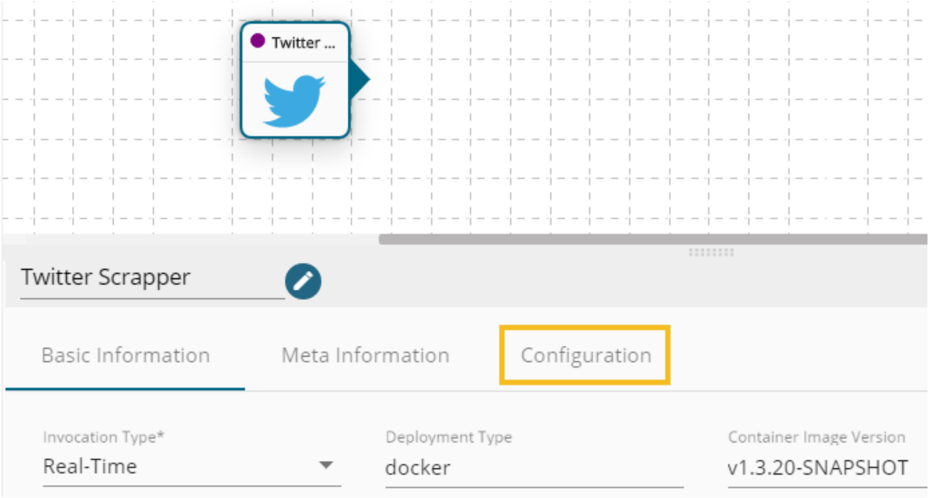
v) Save the Twitter Scrapper component (A notification message appears to inform the same).



vi) Click the '**Update Pipeline**' icon (A notification message appears to inform the same).



vii) Open the Twitter Scrapper component to see the **Configuration tab** below.



Twitter Scrapper

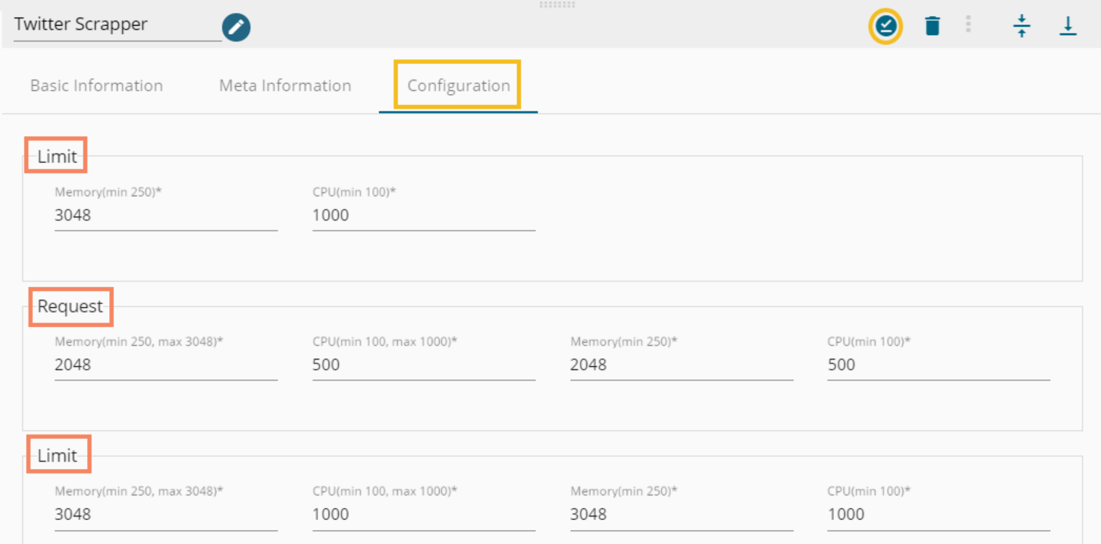
Basic Information | Meta Information | Configuration

Invocation Type*: Real-Time

Deployment Type: docker

Container Image Version: v1.3.20-SNAPSHOT

viii) Click on the Configuration tab to configure the required information.
   a. Limit: Provide Memory and CPU details to configure the limit
   b. Request: Provide Memory and CPU details to configure the request
   c. Click the '**Save Component in Storage**' ✓ icon.



Twitter Scrapper

Limit

Memory(min 250)*: 3048
CPU(min 100)*: 1000

Request

Memory(min 250, max 3048)*: 2048
CPU(min 100, max 1000)*: 500
Memory(min 250)*: 2048
CPU(min 100)*: 500

Limit

Memory(min 250, max 3048)*: 3048
CPU(min 100, max 1000)*: 1000
Memory(min 250)*: 3048
CPU(min 100)*: 1000

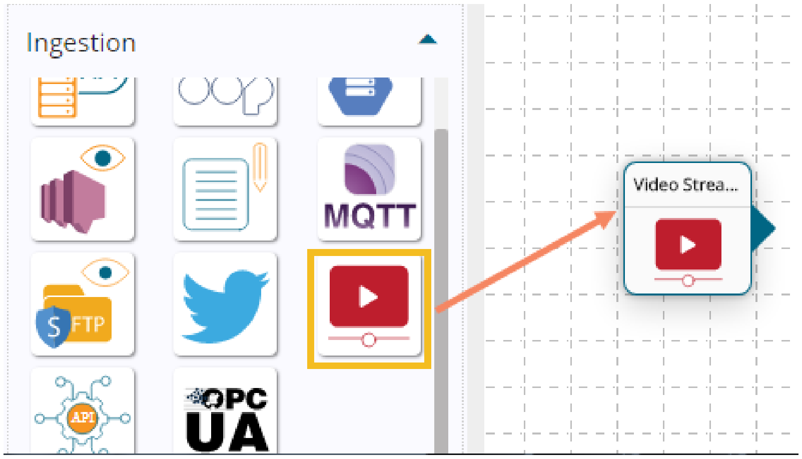
ix) The Twitter Scrapper component properties get saved and a notification message appears to inform the same.

Component properties saved.

Note: The user should know Consumer API Key, Consumer Secret Key, Access Token, Access Token Secret from a developer account of twitter.

8.5.9. Video Stream Consumer

i) Drag & drop the Video Stream Consumer component to the Workflow Editor.

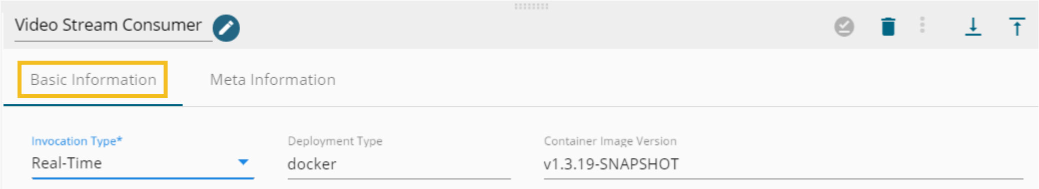


ii) Click the dragged Video Stream Consumer component to open the component properties tabs.

iii) **Basic Information**: It is the default tab to open for the component.

1) Select an Invocation type from the drop-down menu to confirm the running mode of the reader component. The supported option is ‘**Real-Time**’.
2) Deployment Type: It displays the deployment type for the component. This field comes pre-selected.

3) Container Image Version: It displays the image version for the docker container. This field comes pre-selected.

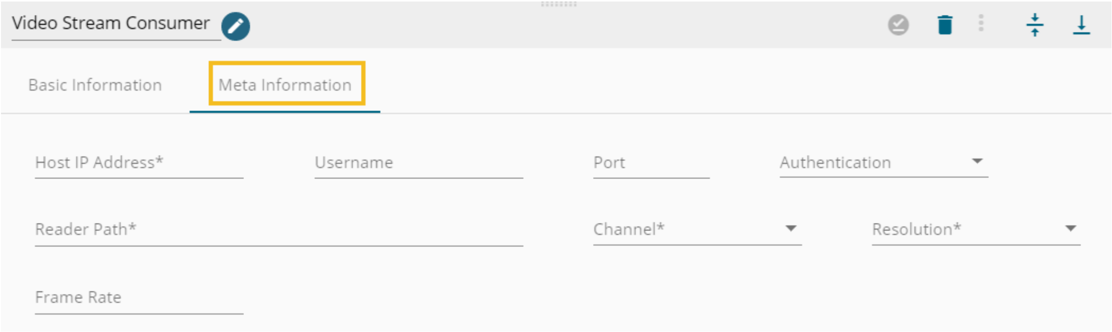

Video Stream Consumer

Basic Information | Meta Information

Invocation Type*: Real-Time
Deployment Type: docker
Container Image Version: v1.3.19-SNAPSHOT

iv) Select the **Meta Information tab** and provide the mandatory fields to configure the dragged Video Writer component.

a. Host IP Address(*)- Provide IP or URL
b. Username(*)- Provide username
c. Port (*)- Provide Port number
d. Authentication- select an authentication option out of password or PEM PPK File
e. Reader Path(*)-
f. Channel(*)- The supported channels are SFTP and URL
g. Resolution(*)- select an option defining the video resolution out of the given options
h. Frame Rate

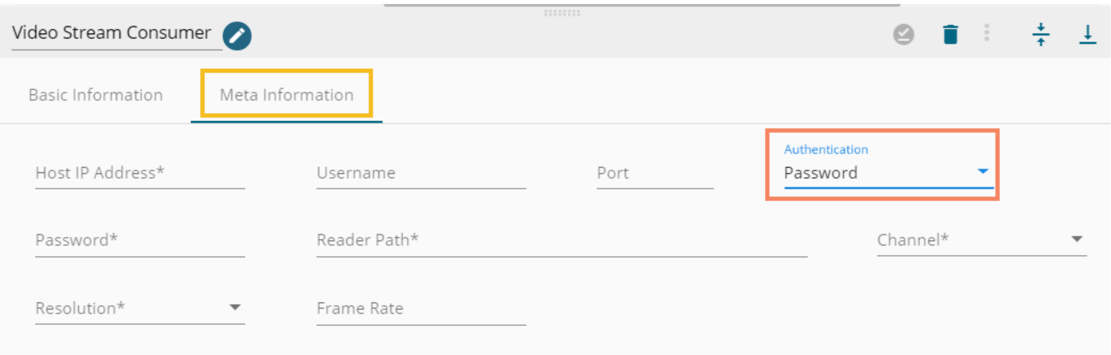

Video Stream Consumer

Basic Information | Meta Information

Host IP Address*
Username
Port
Authentication
Reader Path*
Channel*
Resolution*
Frame Rate

The fields for the Meta Information tab changes based on the selection of the Authentication option.

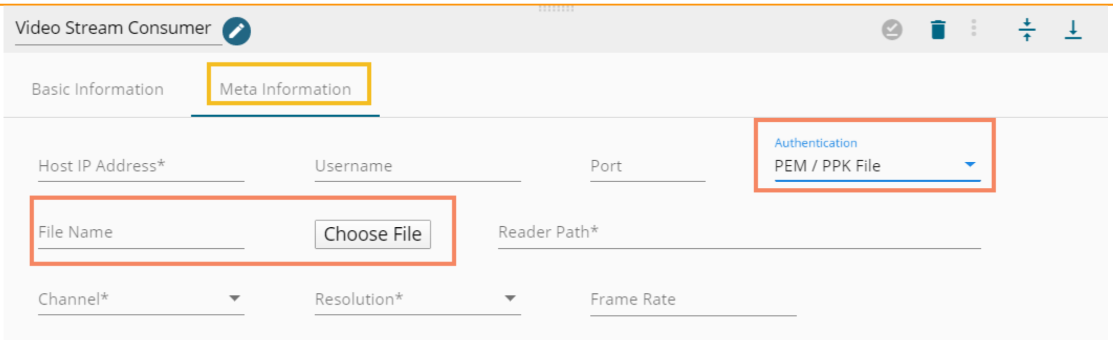
When the authentication option is Password


Video Stream Consumer

Basic Information | Meta Information

Host IP Address*
Username
Port
Authentication: Password
Password*
Reader Path*
Channel*
Resolution*
Frame Rate

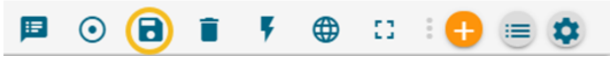
When the authentication option is PEM/PPK file



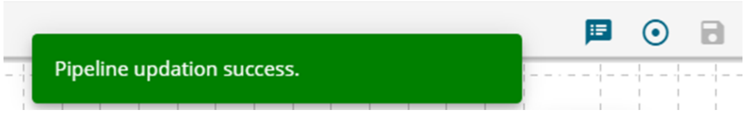
Authentication: PEM / PPK File

While choosing the PEM/PPK File authentication option, the user needs to select a file using the Choose File option.

v) Click the ‘**Save Component in Storage’** icon for the Video Writer component.
vi) A message appears to notify the same.
vii) Click the ‘**Update Pipeline**’ icon to save the changes.

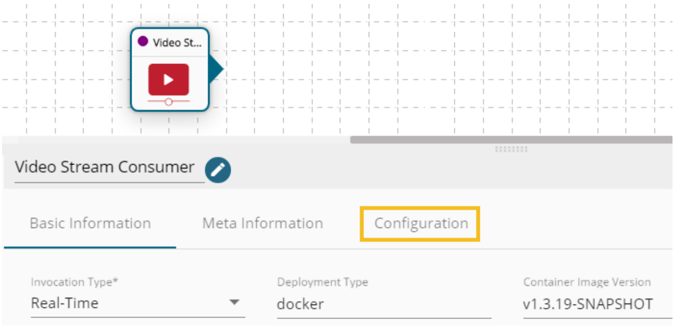


viii) A message appears to notify that the pipeline has been updated.



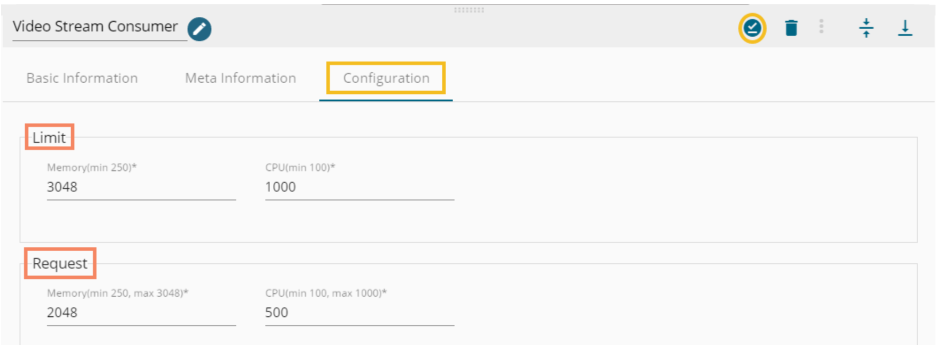
Pipeline updation success.

ix) Open the dragged Video Stream Consumer component again to see the **Configuration properties tab** below.



Invocation Type*: Real-Time
Deployment Type: docker
Container Image Version: v1.3.19-SNAPSHOT

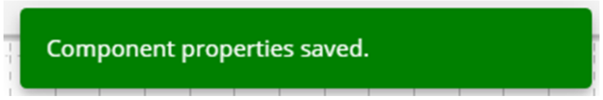
x) Modify the values of the Configuration tab and click the ‘**Save Component in Storage**’ icon.



Limit
Memory(min 250)*: 3048
CPU(min 100)*: 1000

Request
Memory(min 250, max 3048)*: 2048
CPU(min 100, max 1000)*: 500

xi) The message appears to notify that the component properties are saved.


Component properties saved.

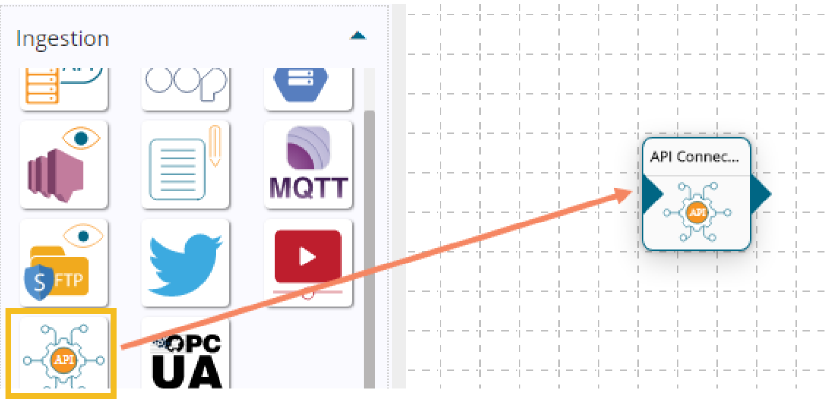
xii) The Video Steam Consumer component is ready to pass the data in the Pipeline Workflow.

Note: The Video Stream Consumer supports only the Video URL.

8.5.10. API Connector

The API connector helps to ingest data from the Google Analytics and Service Now API connectors.

i) Drag and Drop the **API Connector** component to the Workflow Editor.



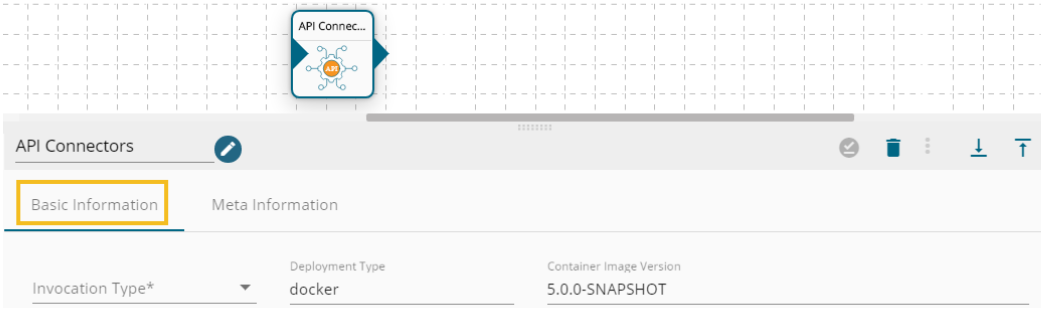
ii) Click on the dragged ingestion component to get the component properties tabs.

iii) Configure the **Basic Information** tab.

a. Select an Invocation type from the drop-down menu to confirm the running mode of the reader component. Select one option out of ‘**Real-Time**’ and ‘**Batch**’.

b. Deployment Type: It displays the deployment type for the component. This field comes pre-selected.

c. Container Image Version: It displays the image version for the docker container. This field comes pre-selected.



Invocation Type*

Deployment Type: docker

Container Image Version: 5.0.0-SNAPSHOT

iv) Click on the **Meta Information tab** to open the component properties fields.

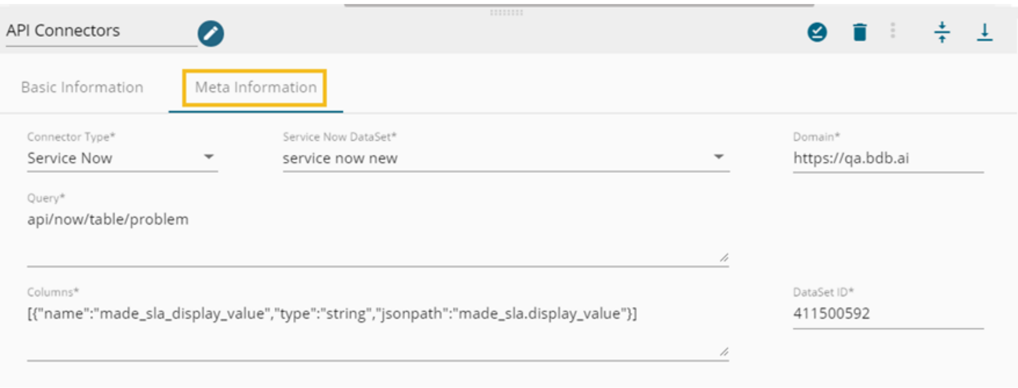
a) Customer Type: Select one option from the drop-down menu. The given choices are:

i. Service Now

ii. Google Analytics

Meta Information fields when the Connector Type is Service Now

1. Connector Type: Select the Service Now option from the drop-down menu.
2. Service Now DataSet: Select a Service Now Data Set from the drop-down menu.
3. Domain: it gets selected based on the chosen Data Set
4. Query: It gets reflected the query statement based on the selected Data Set
5. Columns: It gets reflected based on the selected Data Set
6. DataSet ID: It gets reflected based on the selected Data Set



Connector Type*: Service Now

Service Now DataSet*: service now new

Domain*: https://qa.bdb.ai

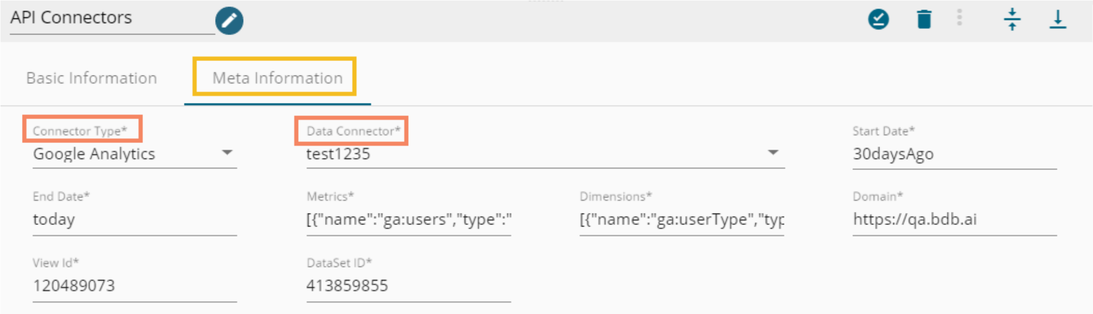
Query*: api/now/table/problem

Columns*: [{"name":"made_sla_display_value","type":"string","jsonpath":"made_sla.display_value"}]

DataSet ID*: 411500592

Meta Information fields when the Connector Type is Google Analytics

1. Connector Type: Select the Google Analytics option from the drop-down menu.
2. Data Connector: Select a Google Analytics data connector from the drop-down menu.
3. Start Date: It gets reflected based on the chosen Data Connector.
4. End Date: It gets reflected based on the chosen Data Connector.
5. Metrics: The selected metrics get reflected based on the choice of the data connector.
6. Dimensions: The selected dimensions from the data connector get reflected.
7. Domain: The domain address appears based on the selected data connector.
8. View id: It gets reflected from the selected data connector.
9. Data Set ID: It gets reflected from the selected data connector.



Connector Type*: Google Analytics

Data Connector*: test1235

Start Date*: 30daysAgo

End Date*: today

Metrics*: [{"name":"ga:users","type":"

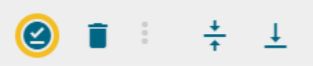
Dimensions*: [{"name":"ga:userType","typ

Domain*: https://qa.bdb.ai

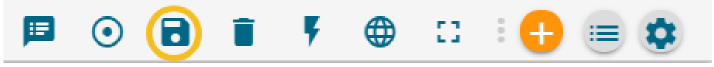
View Id*: 120489073

DataSet ID*: 413859855

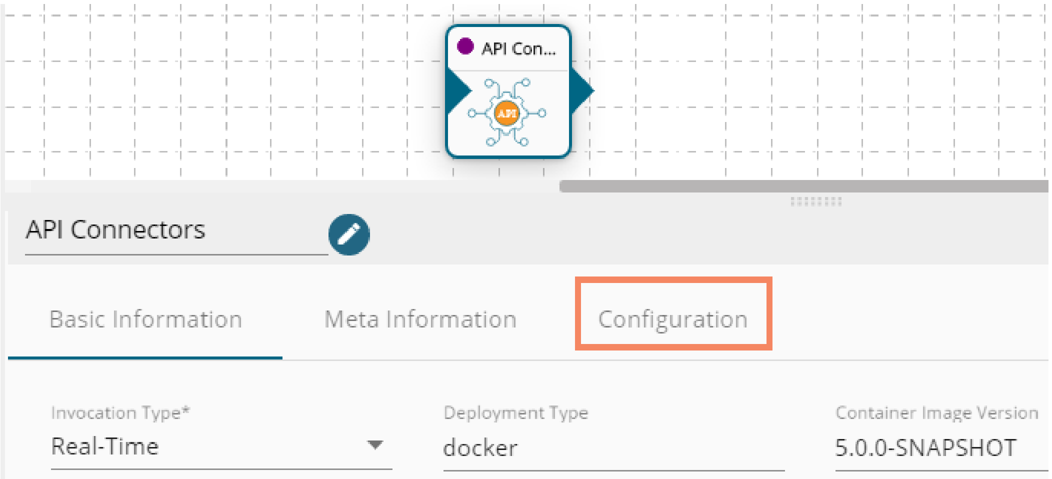
ii) Click the '**Save Component in Storage**' icon to save the component properties (A notification message appears to confirm the same).



iii) Click the '**Update Pipeline**' icon (A Notification message appears to confirm the action).



iv) Open the API Connector ingestion component again to see the **Configuration tab** below.



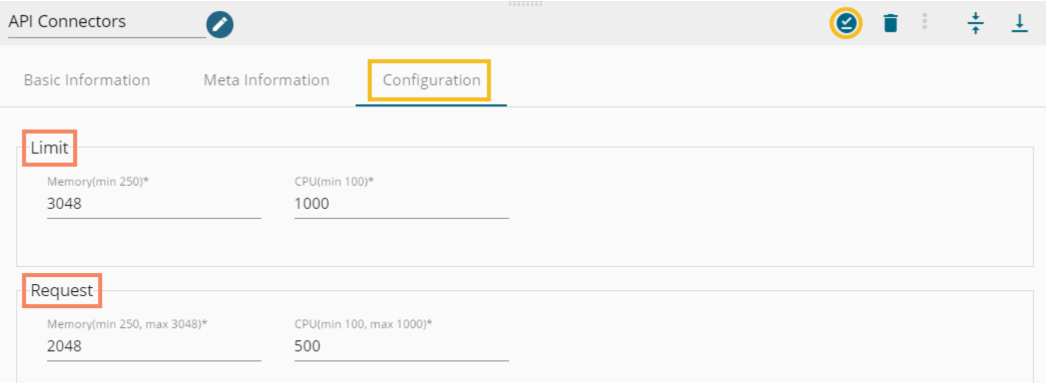
Invocation Type*: Real-Time

Deployment Type: docker

Container Image Version: 5.0.0-SNAPSHOT

v) Modify the Configuration values (if needed).

vi) Click the '**Save Component in Storage**' icon.



Limit

Memory(min 250)*: 3048

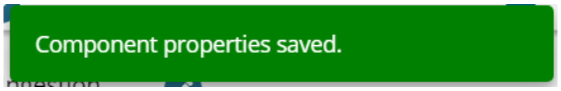
CPU(min 100)*: 1000

Request

Memory(min 250, max 3048)*: 2048

CPU(min 100, max 1000)*: 500

vii) A notification message appears to inform about the action.

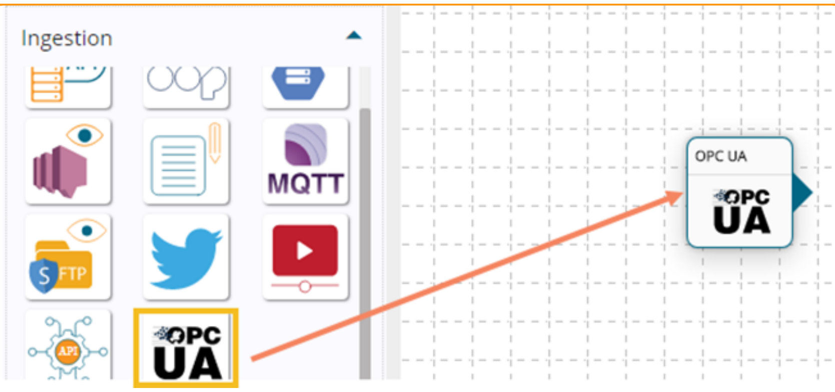


Component properties saved.

viii) The API Connector component gets configured.

8.5.11. OPC UA

i) Drag and Drop the **OPC UA** ingestion component to the Workflow Editor.



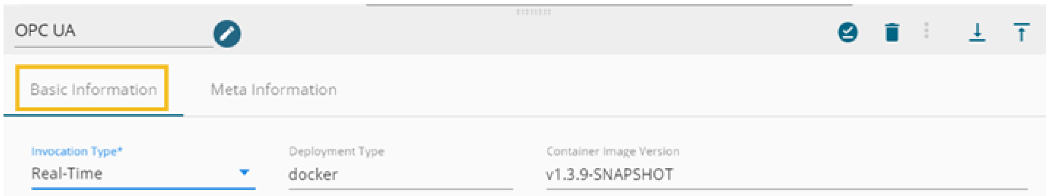
ii) Click on the dragged ingestion component to get the component properties tabs.

iii) Configure the **Basic Information tab**.

a. Select an Invocation type from the drop-down menu to confirm the running mode of the reader component. Select one option out of ‘**Real-Time**’ and ‘**Batch**’.

b. Deployment Type: It displays the deployment type for the component. This field comes pre-selected.

c. Container Image Version: It displays the image version for the docker container. This field comes pre-selected.


OPC UA

Invocation Type*: Real-Time

Deployment Type: docker

Container Image Version: v1.3.9-SNAPSHOT

iv) Configure the **Meta Information tab** by providing the required fields.

a. URL(*)- Provide URL link

b. Message Security Mode- Select a message security mode from the drop-down menu (The supported options are Sign and SignAndEncrypt)

c. Security Policy- Select a policy using the drop-down menu.

d. Certificate File Name- This name gets reflected based on the Choose File option provided for the Certificate file.

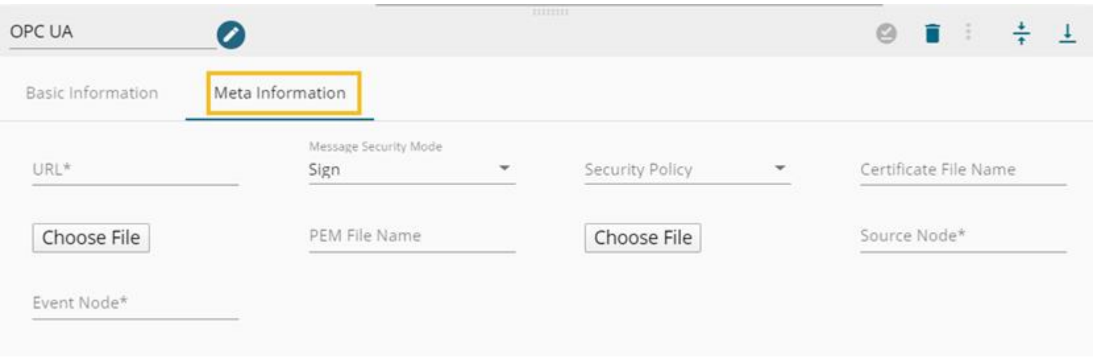
e. Choose File- Browse a certificate file by using this option.

f. PEM File Name: This name gets reflected based on the Choose File option provided for the PEM file.

g. Choose File: Browse a PEM file by using this option.

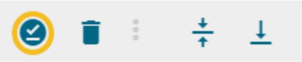
h. Source Node: Provide the source node.

i. Event Note: Provide the event node.

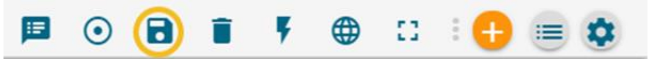

OPC UA

Message Security Mode: Sign

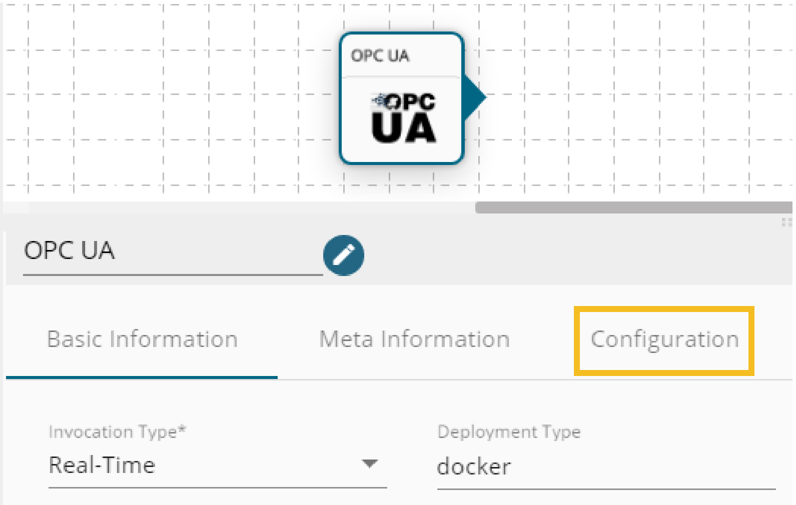
v) Click the ‘**Save Component in Storage**’ icon (A notification message appears to confirm the same).



vi) Click the ‘**Update Pipeline**’ icon (A notification message appears to confirm the same).



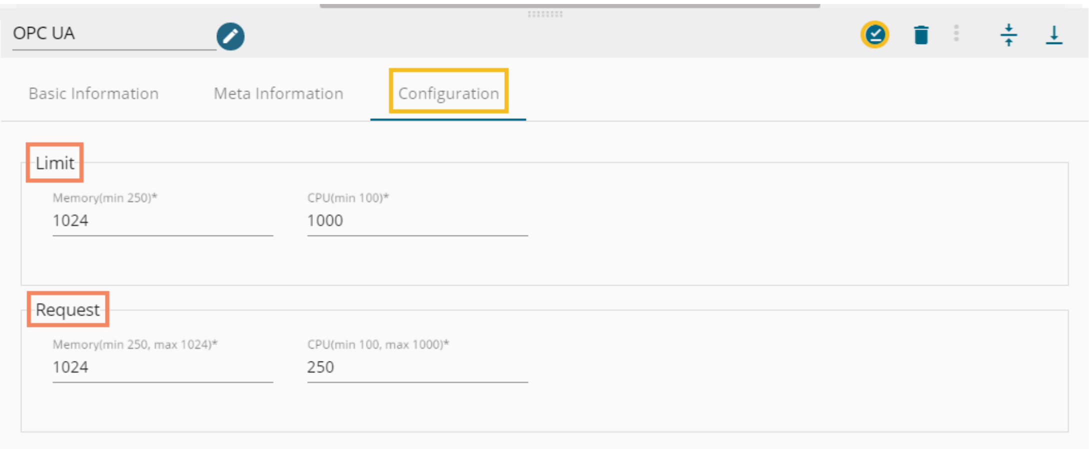
vii) Open the OPC UA component to see the **Configuration tab** below.



Invocation Type*: Real-Time

Deployment Type: docker

viii) Modify the values for the Configuration tab and click the ‘**Save Component in Storage**’ icon.



Limit

Memory(min 250)*: 1024

CPU(min 100)*: 1000

Request

Memory(min 250, max 1024)*: 1024

CPU(min 100, max 1000)*: 250

ix) The OPC UA ingestion component gets configured and a notification message appears to inform the same.

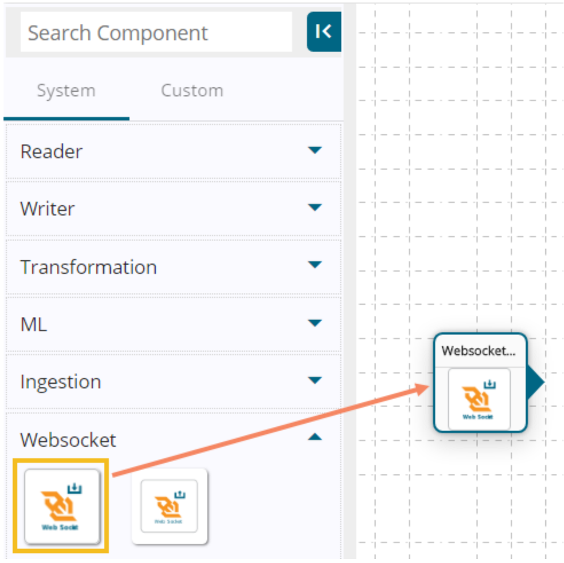
8.6. WebSocket

The Websocket component provided in the Components Pallet of the Data pipeline contains Websocket Listener and Websocket Writer components to produce and consume the live streaming data using socket.

8.6.1. Websocket Listener

Websocket Listener component listens to and accepts the incoming Websocket connection requests.

i) Drag the Websocket Listener component to the Workflow Editor.



ii) Open the Websocket Listener component to get the Component Properties tab below.

iii) The **Basic Information tab** opens by default.

a. Select an Invocation type from the drop-down menu to confirm the running mode of the reader component. The supported invocation type is ‘**Real-Time**’.

b. Deployment Type: It displays the deployment type for the component. This field comes pre-selected.

c. Container Image Version: It displays the image version for the docker container. This field comes pre-selected.

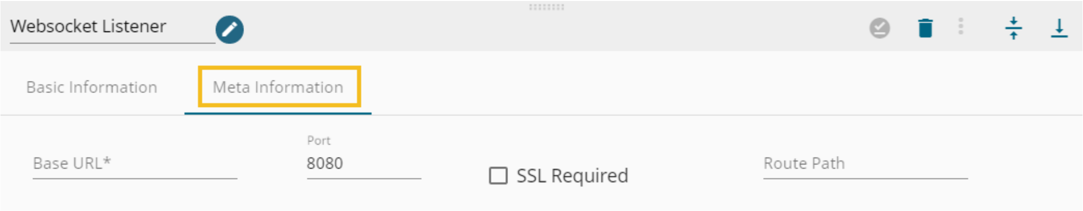
iv) Configure the **Meta information tab** by providing the required fields.

a. Base URL

b. Port

c. SSL Required- Enable the

d. Route Path



Websocket Listener

Basic Information | Meta Information

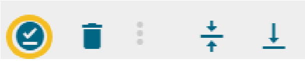
Base URL*

Port: 8080

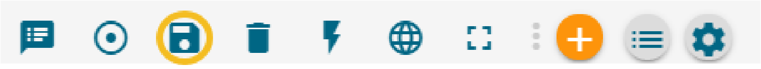
☐ SSL Required

Route Path

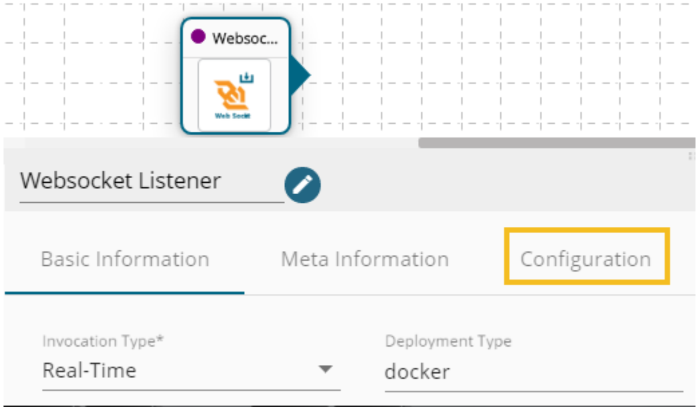
v) Click the ‘**Save Component in Storage**’ icon (A message appears to confirm the same).



vi) Click the ‘**Update Pipeline**’ icon (A message appears to confirm the same).



vii) Open the Websocket Listener component again to see the **Configuration tab** below.

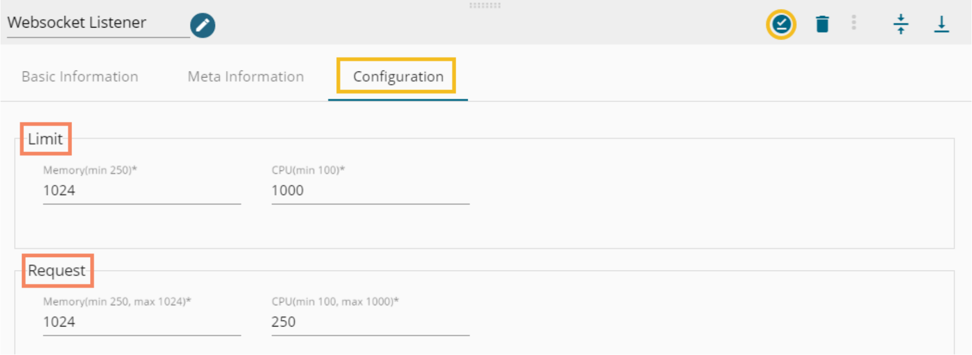


Websocket Listener

Invocation Type*: Real-Time

Deployment Type: docker

viii) Modify the values of the Configuration tab and click the '**Save Component in Storage**' icon.


Websocket Listener

Limit

Memory(min 250)*: 1024

CPU(min 100)*: 1000

Request

Memory(min 250, max 1024)*: 1024

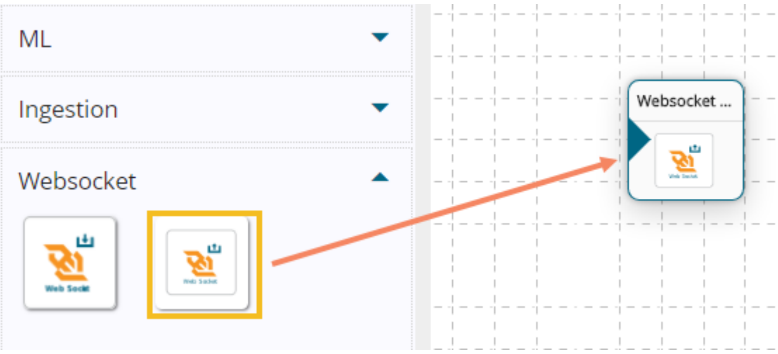
CPU(min 100, max 1000)*: 250

ix) The Websocket Listener component gets configured and a message appears to inform the same.

8.6.2. Websocket Producer

Websocket Producer helps the user to get the message received by the Kafka channel.

i) Drag the Websocket Producer component to the Workflow Editor.



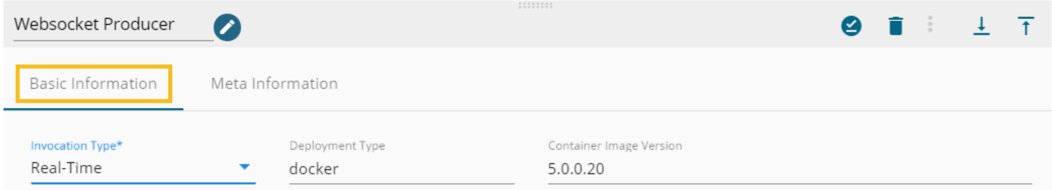
ii) Open the Websocket Producer component to get the Component Properties tab below.

iii) The **Basic Information tab** opens by default.

a. Select an Invocation type from the drop-down menu to confirm the running mode of the reader component. The supported invocation type is '**Real-Time**'.

b. Deployment Type: It displays the deployment type for the component. This field comes pre-selected.

c. Container Image Version: It displays the image version for the docker container. This field comes pre-selected.



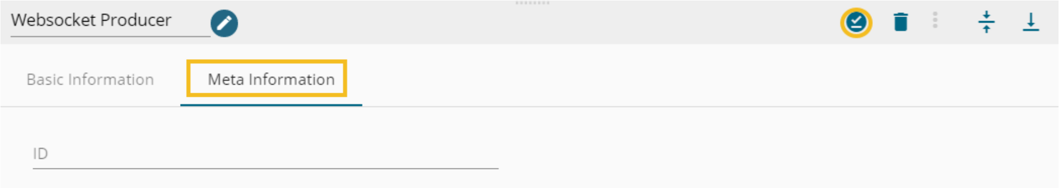
Websocket Producer

Invocation Type*: Real-Time

Deployment Type: docker

Container Image Version: 5.0.0.20

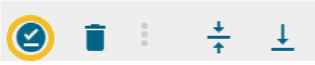
iv) Provide the Websocket ID field to configure the **Meta Information tab**.



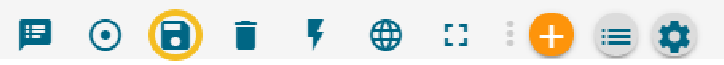
Websocket Producer

ID

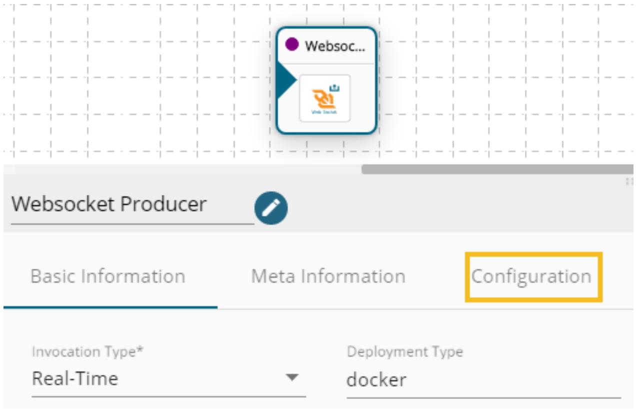
v) Click the '**Save Component in Storage**' icon (A message appears to confirm the same).



vi) Click the '**Update Pipeline**' icon (A message appears to confirm the same).



vii) Open the Websocket Producer component again to see the **Configuration tab** below.

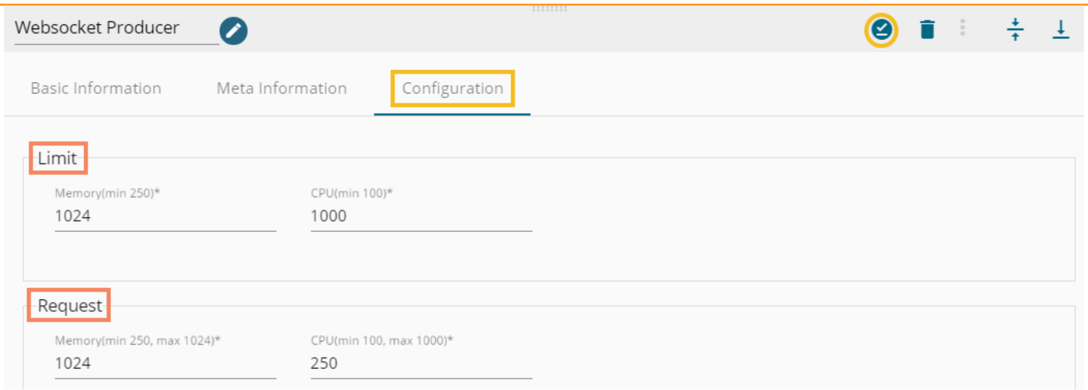


Websocket Producer

Invocation Type*: Real-Time

Deployment Type: docker

viii) Modify the values of the Configuration tab and click the '**Save Component in Storage**' icon.



Limit

Memory(min 250)*: 1024

CPU(min 100)*: 1000

Request

Memory(min 250, max 1024)*: 1024

CPU(min 100, max 1000)*: 250

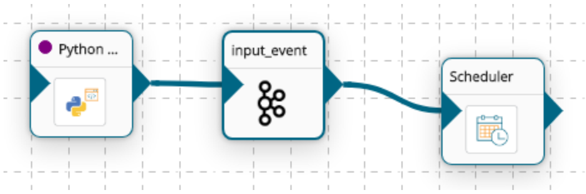
ix) The Websocket Producer component gets configured and a message appears to inform the same.

8.7. Scheduler

The scheduled pipelines are listed together with the scheduler details. It displays the meta-information filled in the scheduler component for the respective pipeline. The page also contains the information on how many times the pipeline has been triggered and when the next time the scheduled component will get deployed.

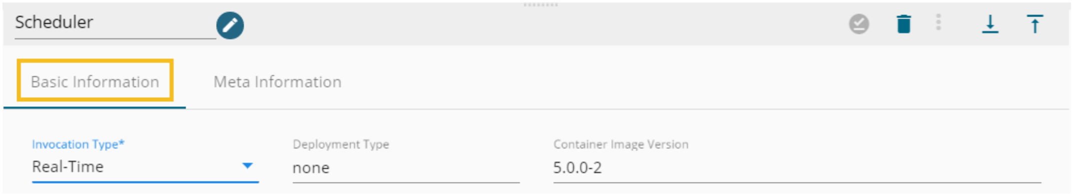
i) Drag and drop the Scheduler component to the Workflow Editor.

ii) Connect it with a reader or Data Loading component (Input event).



iii) Click on the Scheduler component to get the configuration details.

iv) The Basic tab opens by default.



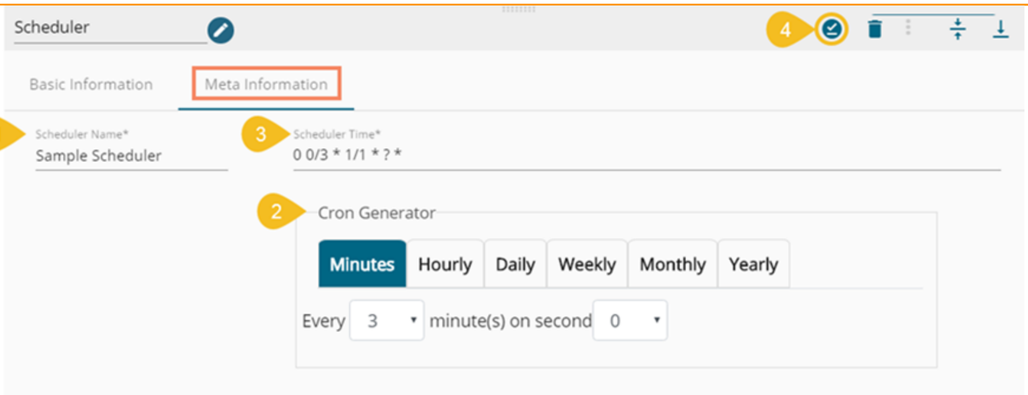
Invocation Type*: Real-Time

Deployment Type: none

Container Image Version: 5.0.0-2

v) Open the Meta Information tab and configure the required fields.

vi) Click the ‘**Save Component in Storage**’ icon to save the component configuration.

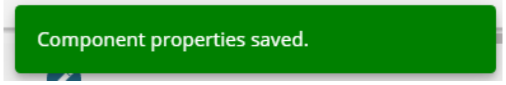


Scheduler Name*: Sample Scheduler

Scheduler Time*: 0 0/3 * 1/1 * ? *

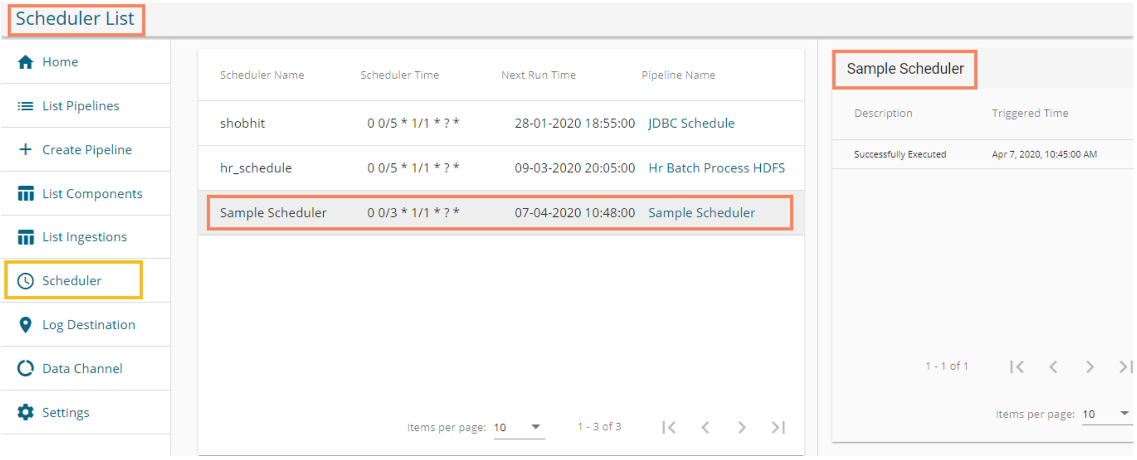
Every 3 minute(s) on second 0

vii) A notification message appears to confirm the action.



Component properties saved.

viii) Update and activate the pipeline (notification messages appear to confirm the same).
ix) Open the Scheduler tab from the landing page screen.
x) The scheduled pipeline gets listed displaying the triggered time and Next Run Time details.



Scheduler Name	Scheduler Time	Next Run Time	Pipeline Name
shobhit	0 0/5 * 1/1 * ? *	28-01-2020 18:55:00	JDBC Schedule
hr_schedule	0 0/5 * 1/1 * ? *	09-03-2020 20:05:00	Hr Batch Process HDFS
Sample Scheduler	0 0/3 * 1/1 * ? *	07-04-2020 10:48:00	Sample Scheduler

Items per page: 10 1 - 3 of 3

Sample Scheduler

Description	Triggered Time
Successfully Executed	Apr 7, 2020, 10:45:00 AM

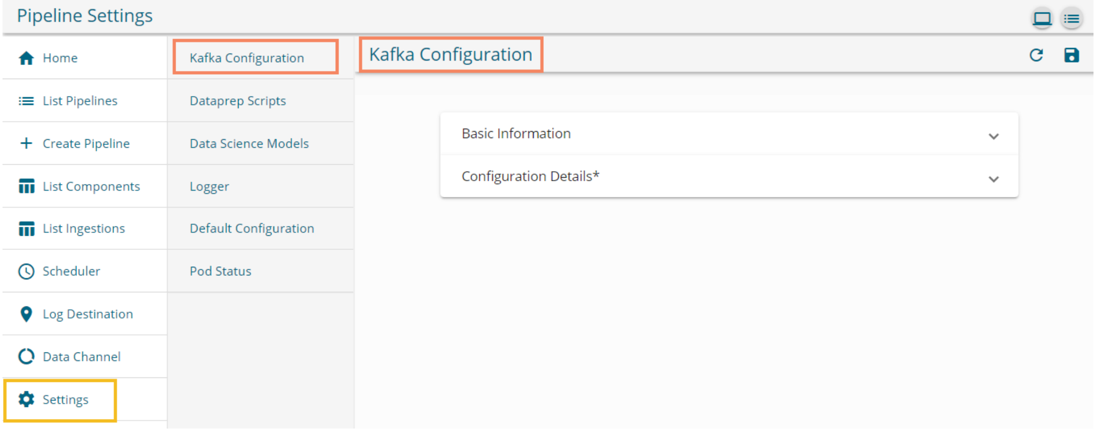
1 - 1 of 1

Items per page: 10

Note: the scheduler supports only reader components at present.

9. Settings

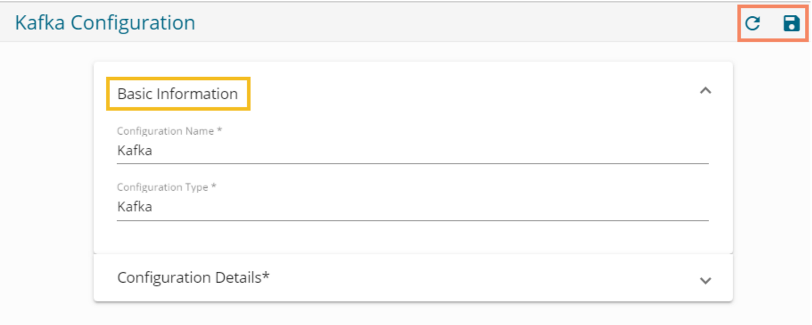
The user gets settings information about various pipeline functionalities. The Settings page provides information about Kafka Configuration, Data Prep Scripts, Data Science Models, and Logger. The Kafka Configuration details open by default while selecting the '**Settings**' page.



9.1. Kafka Configuration

The Kafka Configuration page displays editable Basic Information and Configuration Details for Kafka.

Basic Information contains Configuration Name and Configuration Type. The basic information details can be refreshed and saved by using the icons given for the same.



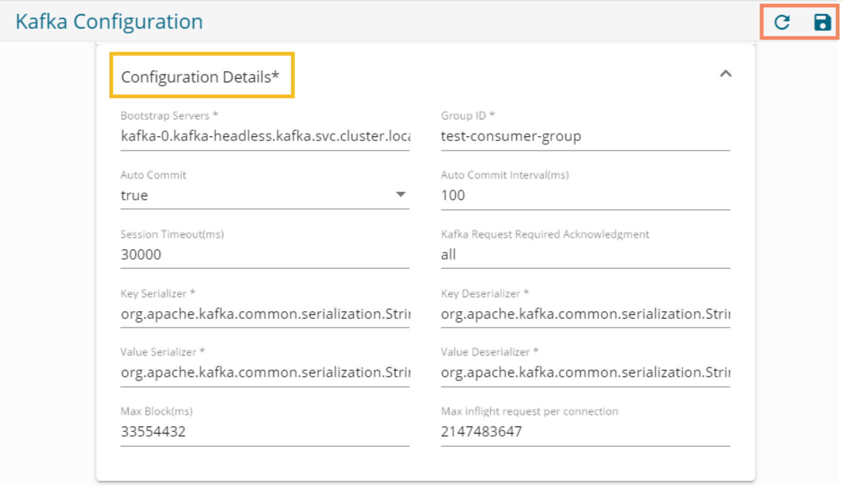
Configuration Name *
Kafka

Configuration Type *
Kafka

The Configuration Details tab for Kafka displays the following information:

a. Bootstrap Servers
b. Group ID
c. Auto Commit
d. Auto Commit Interval(ms)
e. Session Timeout(ms)
f. Kafka Request Required Acknowledgment
g. Key Serializer
h. Key Deserializer
i. Value Serializer
j. Value Deserializer
k. Max Block(ms)
l. Max Inflight request per connection

The user can refresh the configuration and save the configuration by using the provided icons for the same.



Kafka Configuration

Configuration Details*

Bootstrap Servers *
kafka-0.kafka-headless.kafka.svc.cluster.loca

Group ID *
test-consumer-group

Auto Commit
true

Auto Commit Interval(ms)
100

Session Timeout(ms)
30000

Kafka Request Required Acknowledgment
all

Key Serializer *
org.apache.kafka.common.serialization.Strii

Key Deserializer *
org.apache.kafka.common.serialization.Strii

Value Serializer *
org.apache.kafka.common.serialization.Strii

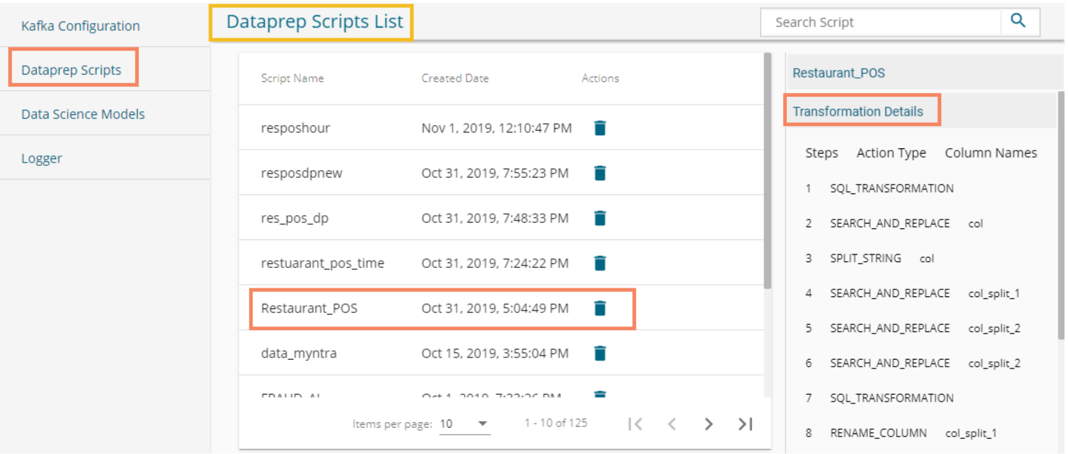
Value Deserializer *
org.apache.kafka.common.serialization.Strii

Max Block(ms)
33554432

Max inflight request per connection
2147483647

9.2. Dataprep Scripts

The user can see a list of all the exported Dataprep Scripts from the Data Preparation module under the Dataprep Scripts tab provided on the Settings page. The user can also view the Transformation Details provided with the sequence of Steps, Action Type, and Column Names for the selected Dataprep Script.



Dataprep Scripts List

Script Name	Created Date	Actions
resposhour	Nov 1, 2019, 12:10:47 PM	🗑
resposdpnew	Oct 31, 2019, 7:55:23 PM	🗑
res_pos_dp	Oct 31, 2019, 7:48:33 PM	🗑
restuarant_pos_time	Oct 31, 2019, 7:24:22 PM	🗑
Restaurant_POS	Oct 31, 2019, 5:04:49 PM	🗑
data_myntra	Oct 15, 2019, 3:55:04 PM	🗑

Items per page: 10 1 - 10 of 125

Restaurant_POS

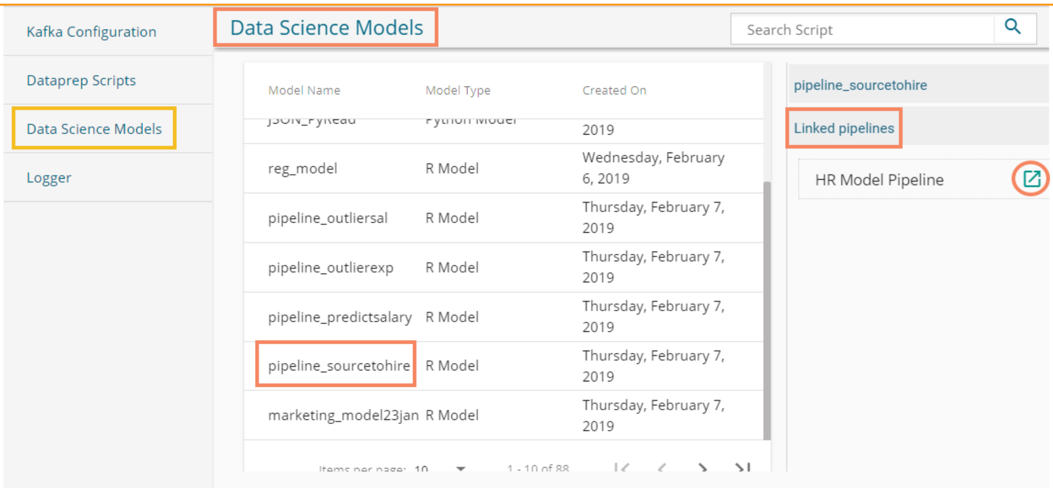
Transformation Details

Steps	Action Type	Column Names
1	SQL_TRANSFORMATION	
2	SEARCH_AND_REPLACE	col
3	SPLIT_STRING	col
4	SEARCH_AND_REPLACE	col_split_1
5	SEARCH_AND_REPLACE	col_split_2
6	SEARCH_AND_REPLACE	col_split_2
7	SQL_TRANSFORMATION	
8	RENAME_COLUMN	col_split_1

Note: The user can delete a Dataprep Script by clicking the ‘**Delete Script**’ 🗑 icon provided next to it.

9.3. Data Science Models

The exported models from Data Science Workbench gets listed on this page. The linked pipeline to the selected model is also mentioned on the right side of the page. The user can navigate to the related pipeline by clicking the ‘**Open Pipeline**’ ↗ icon from this page.



Model Name	Model Type	Created On
reg_model	R Model	Wednesday, February 6, 2019
pipeline_outliersal	R Model	Thursday, February 7, 2019
pipeline_outlierexp	R Model	Thursday, February 7, 2019
pipeline_predictsalary	R Model	Thursday, February 7, 2019
pipeline_sourcetohire	R Model	Thursday, February 7, 2019
marketing_model23jan	R Model	Thursday, February 7, 2019

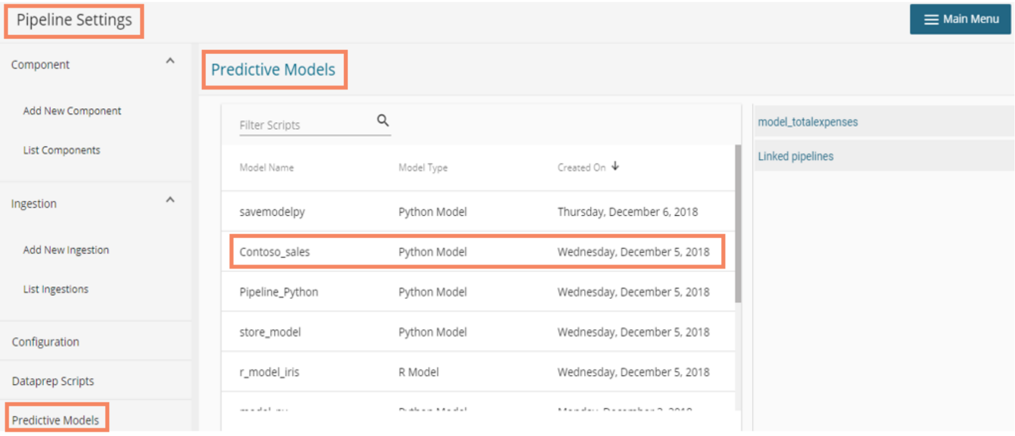
9.3.1. Using a Deployed Model in Data Pipeline

i) Navigate to the Pipeline Settings page.

ii) Select the '**Data Science Models**' option.

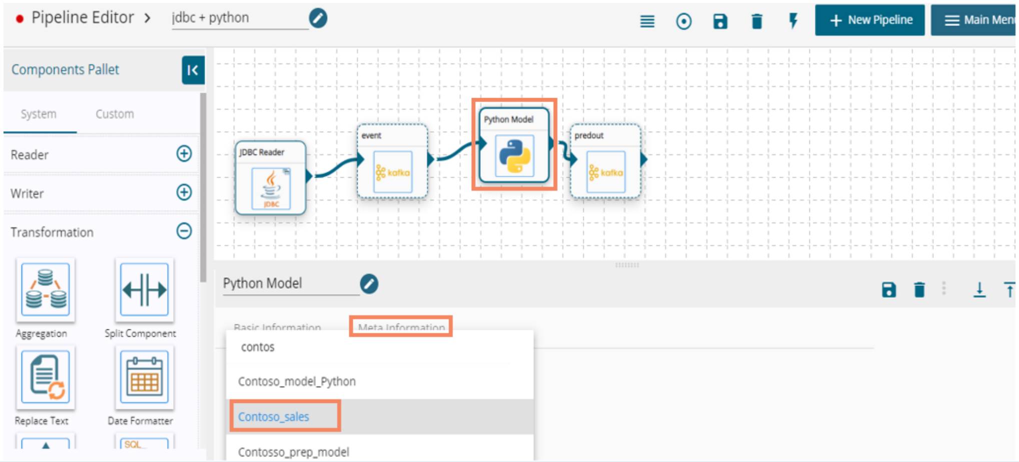
iii) The deployed Data Science Models display in a list.

iv) Select a Data Science Model from the list.



Model Name	Model Type	Created On
savemodelpy	Python Model	Thursday, December 6, 2018
Contoso_sales	Python Model	Wednesday, December 5, 2018
Pipeline_Python	Python Model	Wednesday, December 5, 2018
store_model	Python Model	Wednesday, December 5, 2018
r_model_iris	R Model	Wednesday, December 5, 2018

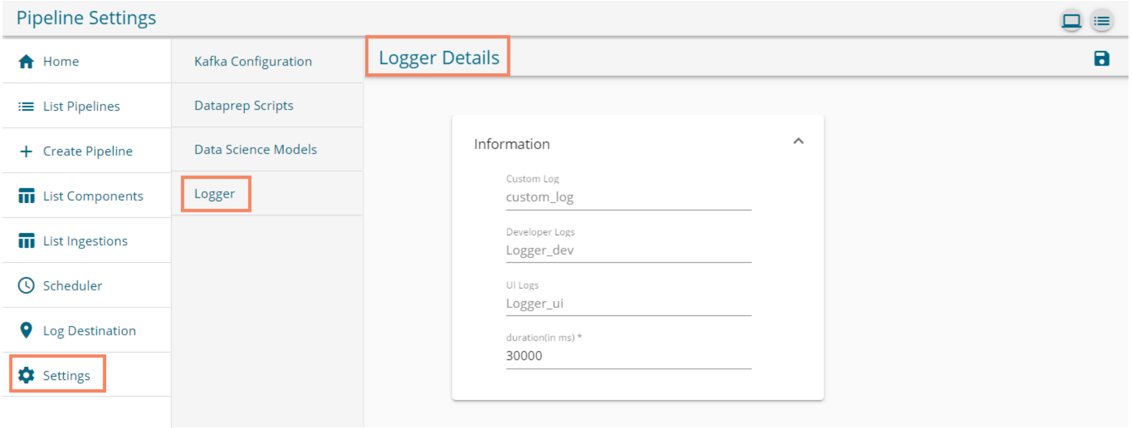
v) The user can access the deployed Data Science Model under the ML Model Runner Drop-down. (E.g., The deployed Python Model appears in the Meta Information tab for the Python Model runner pipeline component.)



contos
Contoso_model_Python
Contoso_sales
Contosso_prep_model

9.4. Logger

This page displays the Logger configuration details for the UI Logs that appear on the log panel of the workflow editor.



Custom Log
custom_log

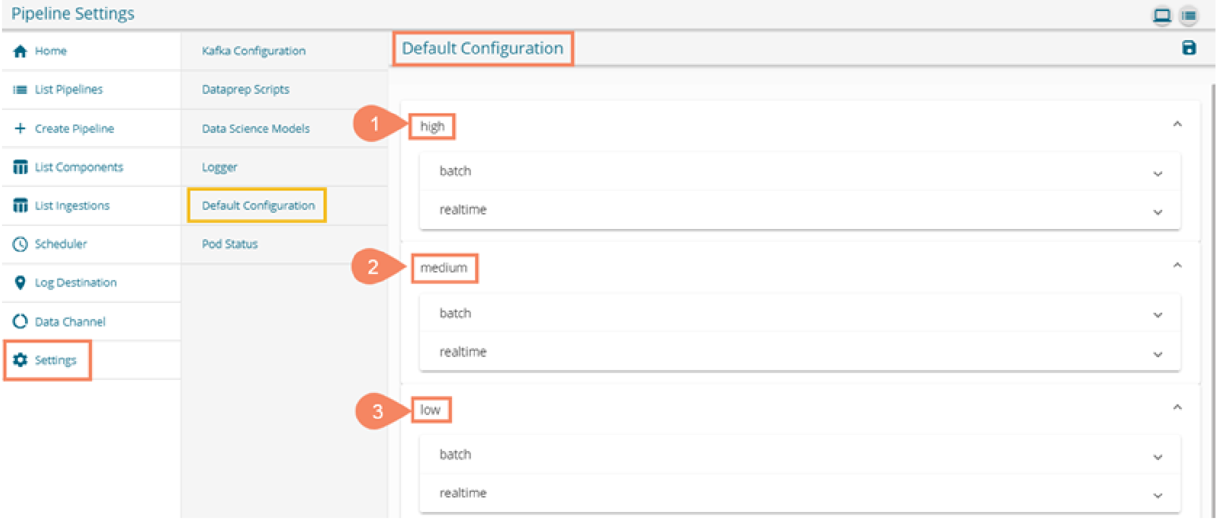
Developer Logs
Logger_dev

UI Logs
Logger_ui

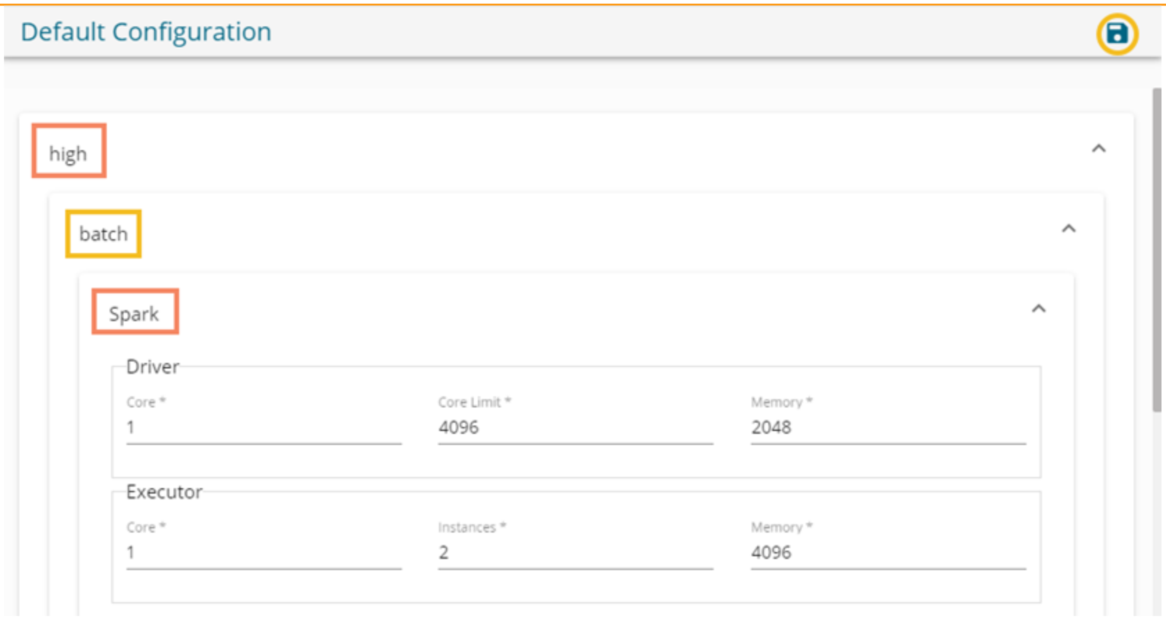
duration(in ms) *
30000

9.5. Default Configuration

The Default Configuration displays pipeline default configuration information categorized as High, Medium, and Low level.



The users can edit the information and click the ‘**Save Configuration**’ icon.



Default Configuration

high

batch

Spark

Driver

Core *: 1

Core Limit *: 4096

Memory *: 2048

Executor

Core *: 1

Instances *: 2

Memory *: 4096

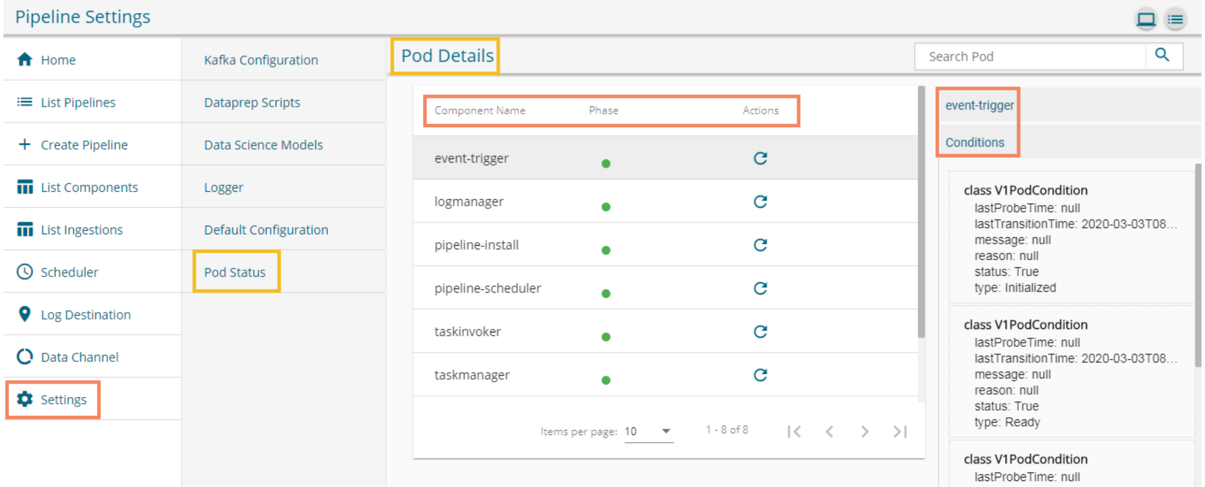
A message appears to confirm the updates.

Configuration updated successfully.

The configuration gets saved.

9.6. Pod Status

The Pod Status window displays the status of the base components pods; the user can refresh the specific Pod using this page if the Phase is RED the user should make the Component up.



Component Name	Phase	Actions
event-trigger	●	↻
logmanager	●	↻
pipeline-install	●	↻
pipeline-scheduler	●	↻
taskinvoker	●	↻
taskmanager	●	↻

```
class V1PodCondition
  lastProbeTime: null
  lastTransitionTime: 2020-03-03T08...
  message: null
  reason: null
  status: True
  type: Initialized

class V1PodCondition
  lastProbeTime: null
  lastTransitionTime: 2020-03-03T08...
  message: null
  reason: null
  status: True
  type: Ready

class V1PodCondition
  lastProbeTime: null
```

10. Third-Party Data Ingestion using Data Pipeline

The Python Script component in Data Pipeline can be used in a very flexible way as a reader, ingestion, transform component, and data writer component.

All the user needs to do is to write a custom script according to the use-case.

Below is an example of listening data from an MQTT topic and calling a machine learning model API. this API return data after applying the model to it.

10.1. Data Ingestion from an MQTT Consumer

Sample Code:

```
   from threading import Thread
   import paho.mqtt.client as mqtt
   import os,json,requests
   from confluent_kafka import Producer

 def get_out_event():
      componentInsId = os.getenv('COMPONENT_INST_ID')
      url = str(os.getenv('MDS_BASE_URL')) + "/api/v1/componentInst/" + str(componentInsId)
      uuid = os.getenv('UUID')
      if uuid is not None:
        url = url + "?uuid=" + str(uuid)
      resp = requests.get(url)
      component_json = json.loads(resp.content)
      data = component_json['data']
      meta_data = data['metaData']
      out_event = meta_data['outEvent'][0]
      return out_event

def kafka_config():
     url = str(os.getenv('MDS_BASE_URL')) + "/api/v1/configuration/kafka"
     resp = requests.get(url)
     component_json = json.loads(resp.content)
     config = component_json['data']['confDetails']['bootstrap.servers']
     return config

   config = kafka_config()
   # config = "192.168.1.32:9092"
   out_event = get_out_event()
   # out_event = "adnoc_wesocket_test"
   p = Producer({'bootstrap.servers': config})

def sent_mqtt_data(broker_address,username,password,topic):
    client = mqtt.Client("P1")  # create new instance
    #client._client_id="lens_2q4aSVgeMiSkFw6fpTUbEJx7G0B"
    client.username_pw_set(username, password)
    client.connect(broker_address,port=1883,keepalive=120)  # connect to broker
    def on_message(client, userdata,message):
      p.produce(out_event,message.payload.decode("utf-8"))
    print("connected")
    client.subscribe(topic=topic,qos=0)
    client.on_message = on_message

    client.loop_forever()
    # return client
```

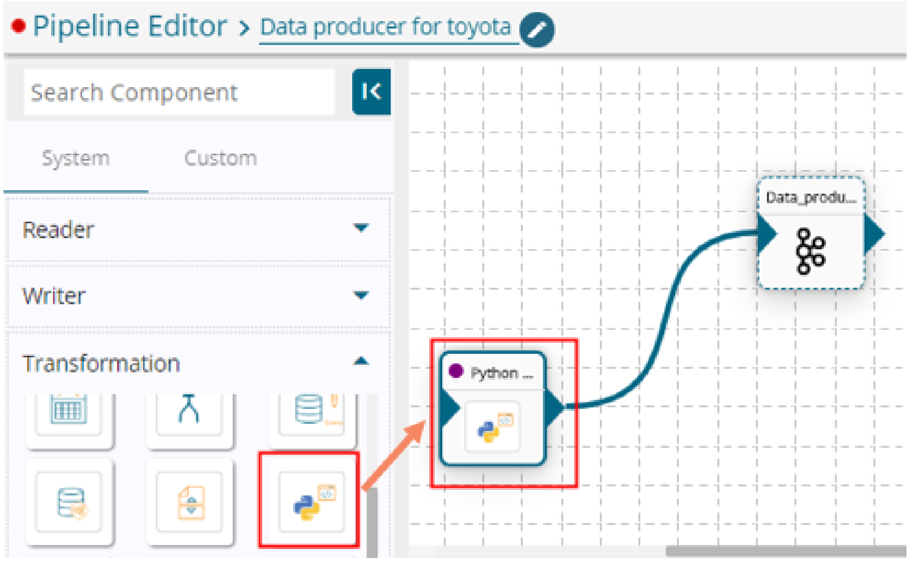
Corresponding File:

sampleMQTT.py

10.2. Configuring a Data Pipeline According to the Script

i) Navigate to the Transformation section of the component pallet.

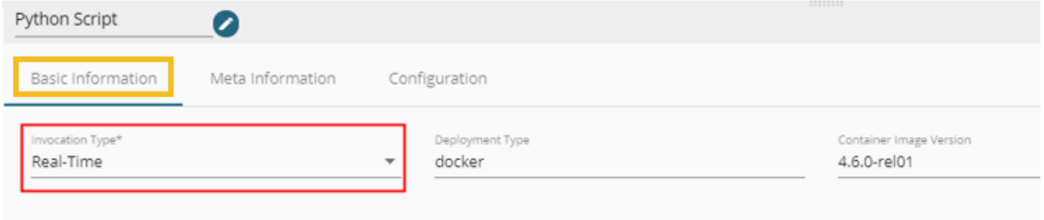
ii) Select and drag the Python Script component to the canvas (Workflow Editor).



iii) Configure the Python Script component.

iv) The '**Basic Information**' tab opens by default.

v) Select the '**Real-Time**' as invocation type so it can keep listening to the MQTT topic in real-time.



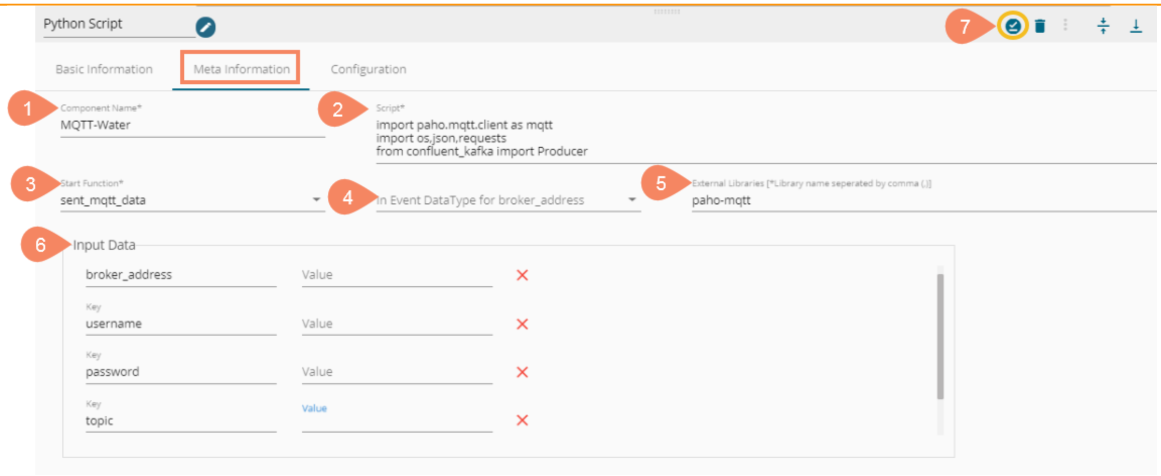
Invocation Type*: Real-Time

Deployment Type: docker

Container Image Version: 4.6.0-rel01

vi) Open the Meta Information tab and configure as displayed in the below image:

1. Component Name: Provide a name for the component.
2. Script: Insert the custom python script in this space.
3. Start Function: Activate the function to run the Python script.
4. In Event Data Type: Select a data frame or list.
5. External Libraries: Provide names of the libraries separated by a comma
6. Input Date: Fill the values of the input parameters of the start function.
7. Click the 'Save' option for the component.



Python Script

Basic Information | Meta Information | Configuration

Component Name*: MQTT-Water

Script*:
```
import paho.mqtt.client as mqtt
import os,json,requests
from confluent_kafka import Producer
```

Start Function*: sent_mqtt_data

In Event DataType for broker_address

External Libraries [*Library name seperated by comma (,)]: paho-mqtt

Input Data

Key	Value
broker_address	Value
username	Value
password	Value
topic	Value

Note : List of already installed libraries are as given below:

```
certifi==2019.6.16
chardet==3.0.4
confluent-kafka==1.1.0
idna==2.8
numpy==1.17.1
pandas==0.25.1
PyMySQL==0.9.3
python-dateutil==2.8.0
pytz==2019.2
requests==2.22.0
six==1.12.0
urllib3==1.25.3
```

10.2.1. Making a POST request along with the data

Sample code:

```
# In[]: arguments for OCR
import requests
import time


def main(df):
   try:
      print("Process Started")

      # In[]: Creating data to post

      url = 'API URL'
      response = requests.post(url,data=data_to send)

      # In[]: Reading the returned information
      res = response.json()
      # results_text = res['lttsText']
      print(res)
      return [res]
```

Corresponding File:

SampleApiPostReq.py

11. Signing Out

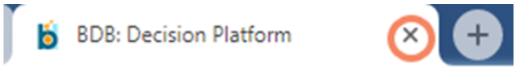
The users can Sign-out from the Data Pipeline tab at any given stage, but preferable is that the users should complete all the tasks they wish to perform and save it before closing the tab or singing out from the Platform.

The Signing Out process for the Data Pipeline has two steps:

1. Closing the BDB Data Pipeline

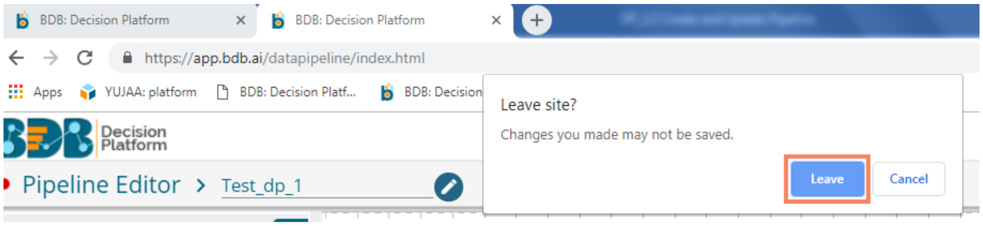
Once you have completed the Data Pipeline tasks, save your work and close the Data Pipeline tab.

i) Click the **'Close'** button (the 'X' on the right edge) from the Data Pipeline tab.



ii) A window appears to confirm the action of closing the BDB Pipeline.

iii) Click the '**Leave**' option to close the selected pipeline.



Leave site?

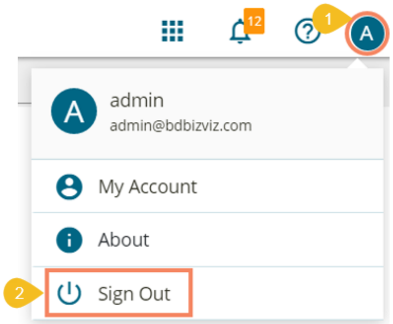
Changes you made may not be saved.

2. Sign Out from the BDB Platform

i) After closing the Data Pipeline plugin, the users get redirected to the Platform homepage.

ii) Click the '**User Profile**' icon on the Platform homepage.

iii) Click the '**Sign Out**' option.



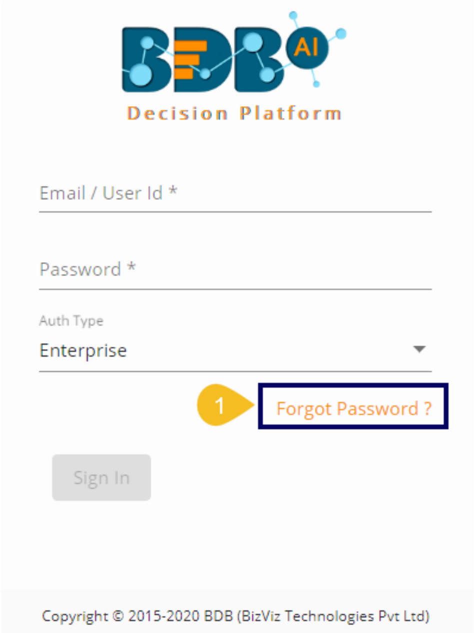
iv) The user successfully signs off from the **BDB Platform**.

Note: Clicking on the '**Sign Out**' option redirects the user back to the login page of the BDB platform.

11.1. Forgot Password Option

The users are provided with a choice to change the password on the Login page of the platform.

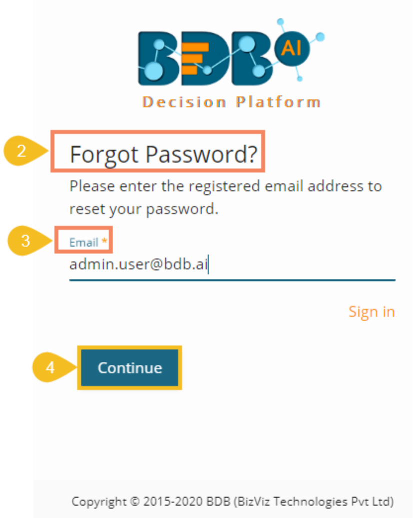
i) Click the '**Forgot Password?**' option from the Sign In page.



ii) The '**Forgot Password?**' page opens.

iii) Provide the email id that is registered with BDB to send the reset password link.

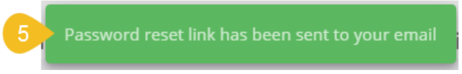
iv) Click the '**Continue**' option.



Please enter the registered email address to reset your password.

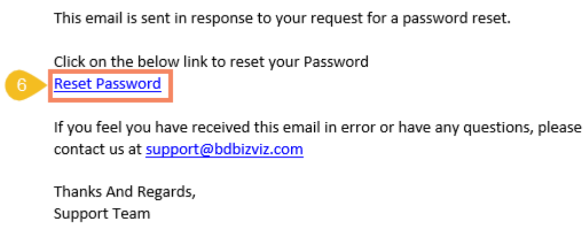
admin.user@bdb.ai

v) The user may be redirected to select a space in case of multiple spaces under one server link(The user needs to select a space and click the '**Continue**' option once again). If a user does not have multiple spaces then, a message appears to notify the user that the password reset link (The users receive the reset link via their registered email.)



Password reset link has been sent to your email

vi) Click the link from your registered email.

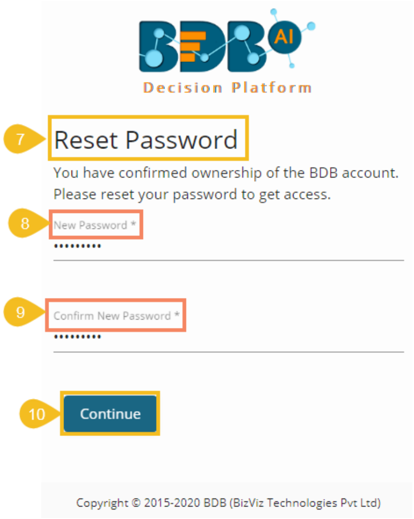

This email is sent in response to your request for a password reset.

Click on the below link to reset your Password
Reset Password

If you feel you have received this email in error or have any questions, please contact us at support@bdbizviz.com

Thanks And Regards,
Support Team

vii) The user gets redirected to the ‘**Reset Password**’ page to set a new password.
viii) Set a new password.
ix) Confirm the newly set password.
x) Click the ‘**Continue**’ option.

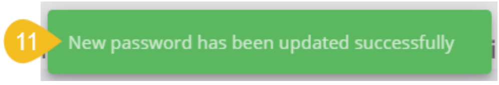


Reset Password

You have confirmed ownership of the BDB account. Please reset your password to get access.

Copyright © 2015-2020 BDB (BizViz Technologies Pvt Ltd)

xi) The password for the selected BDB account gets reset and a message appears to inform the user.



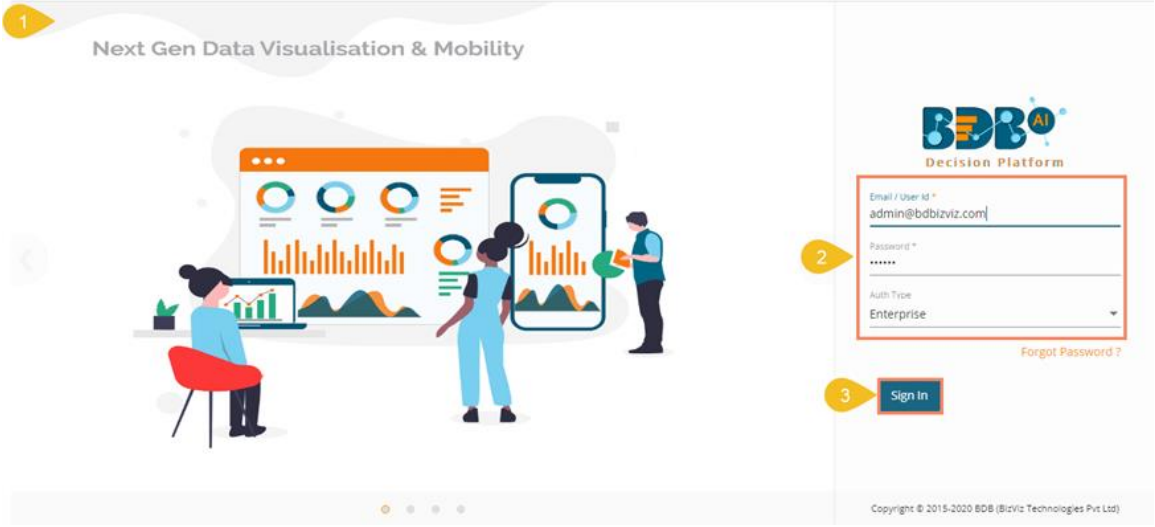
New password has been updated successfully

Note: The user gets redirected back to the Sign In page after successfully resetting the password.

11.2. Force Login

The ‘**Force Login**’ functionality has been introduced to control the number of active sessions up to three. The users can access only 3 sessions at a time when they try to access the 4th session, a warning message displays to inform that the user has consumed the permitted sessions and, a click on the ‘**Force Login**’ would kill all those active sessions.

i) Navigate to the BDB Platform Login page.
ii) Enter the valid credentials to log in.
iii) Click the ‘**Sign In**’ option.



Next Gen Data Visualisation & Mobility

Decision Platform

Email / User Id *
admin@bdbizviz.com

Password *
••••••

Auth Type
Enterprise

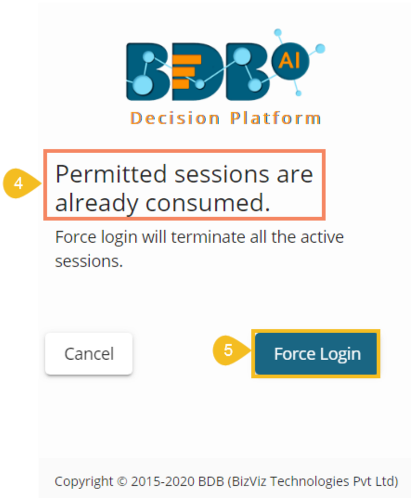
Forgot Password ?

Sign In

Copyright © 2015-2020 BDB (BizViz Technologies Pvt Ltd)

iv) The user gets the following message if the permitted active sessions (3 sessions at a time) are consumed.

v) Click the '**Force Login**' option.



Decision Platform

Permitted sessions are already consumed.

Force login will terminate all the active sessions.

Cancel

Force Login

Copyright © 2015-2020 BDB (BizViz Technologies Pvt Ltd)

vi) A warning message appears the currently active sessions get killed, and the user gets redirected to the BDB Platform Sign In page.

vii) The user needs to provide valid credentials once again and click the '**Continue**' option to access the platform.