

Forms & Workflows Nifi Flow Setup Documentation

Prerequisites:

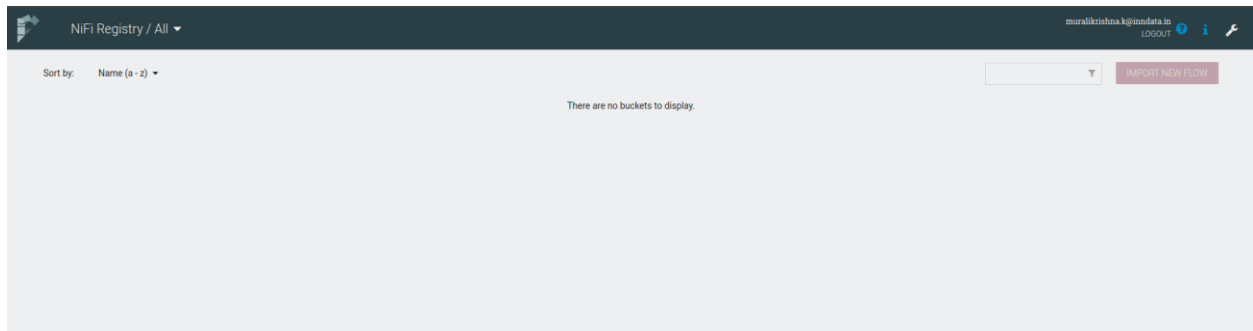
1. **Apache NiFi Installed:** Ensure Apache NiFi is installed and running.
2. **NiFi Registry Installed:** Ensure NiFi Registry is installed and running.
3. **Forms & Workflows JSON File:** A NiFi template for mapping forms and workflows from github.

Contents:

1. Upload Json template to Nifi Registry
2. Importing template from Nifi Registry to Nifi & Connection to Github
3. Management Controller Services setup
4. Parameter Contexts setup
5. Nifi flow configuration
6. Maintain Cached Date in a Separate table
7. Run the Complete Flow

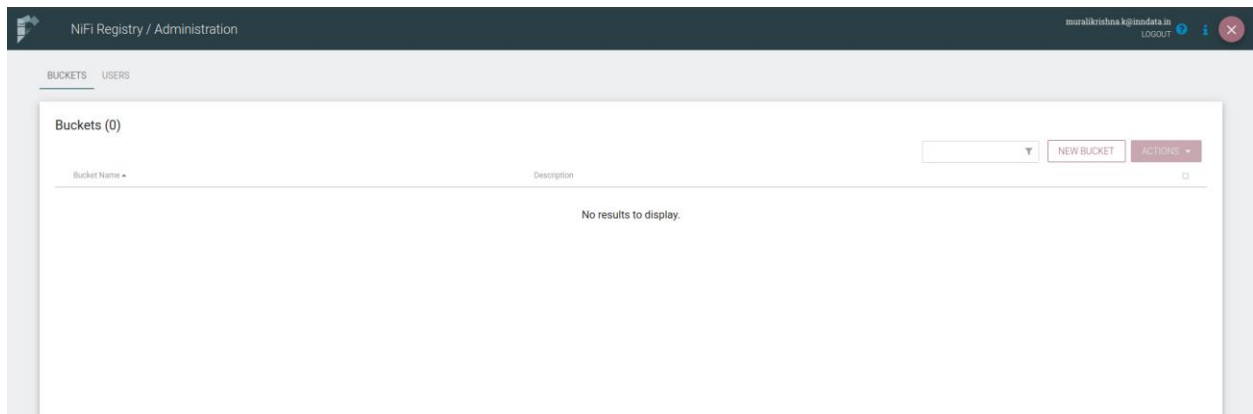
1. Steps to Upload JSON template to NiFi Registry:

Access the NiFi Registry using credentials and then you will have interface like below image.

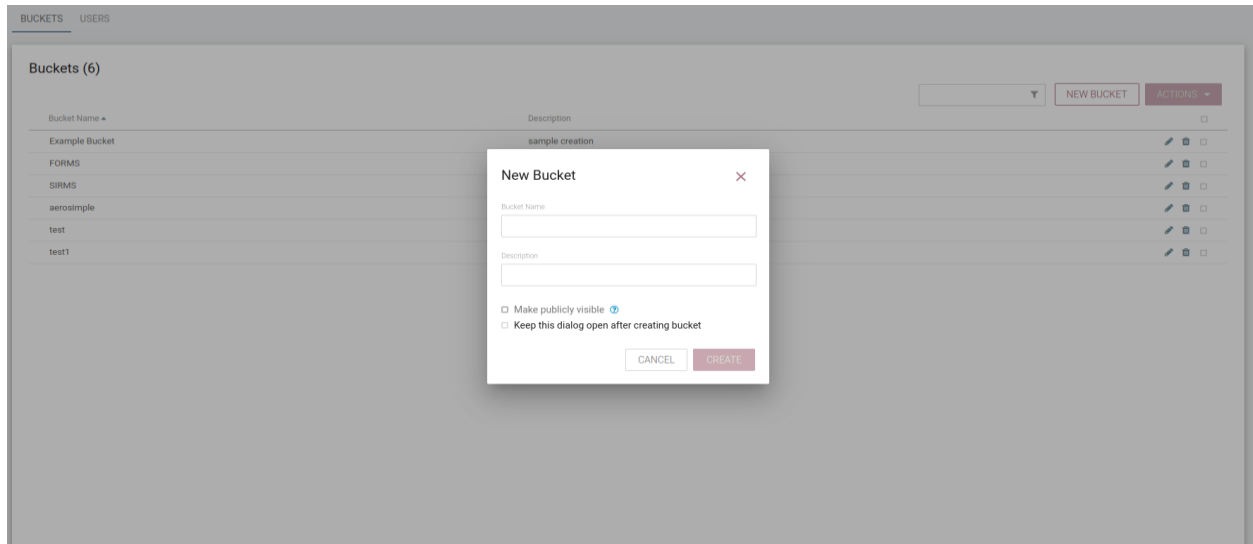


Create a new Bucket in Registry

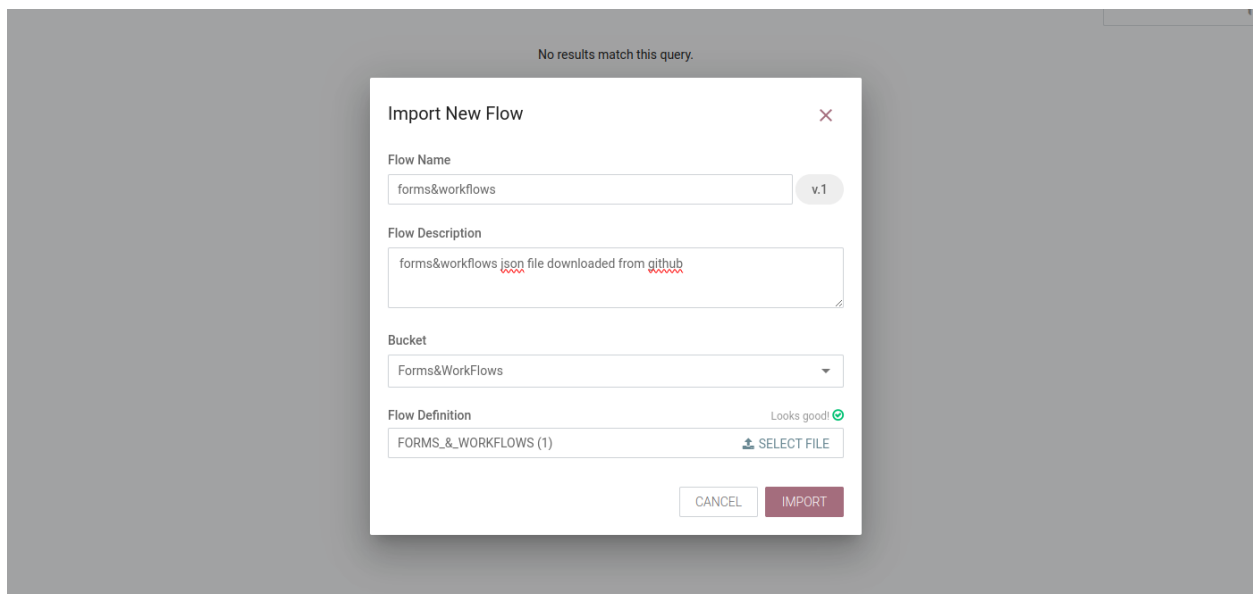
To create a new bucket, click on Settings (Wrench symbol), which is at top right corner then you will have interface like below:



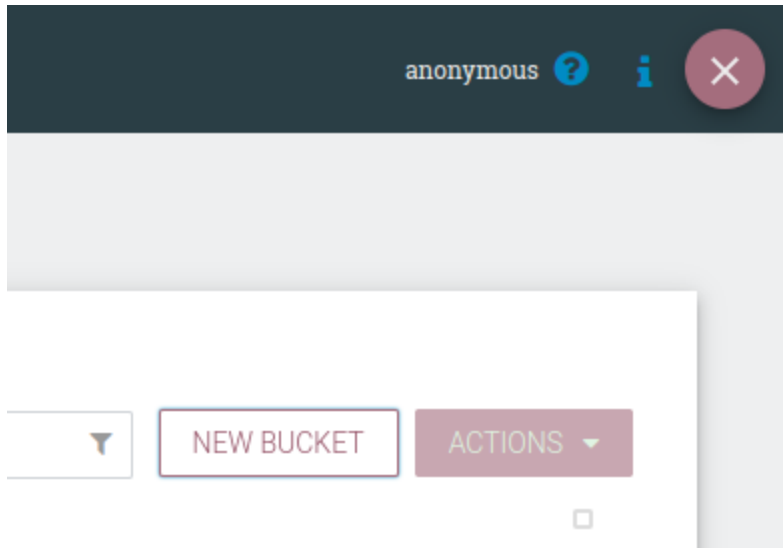
Now click on '**NEW BUCKET**' option, you will have interface like below



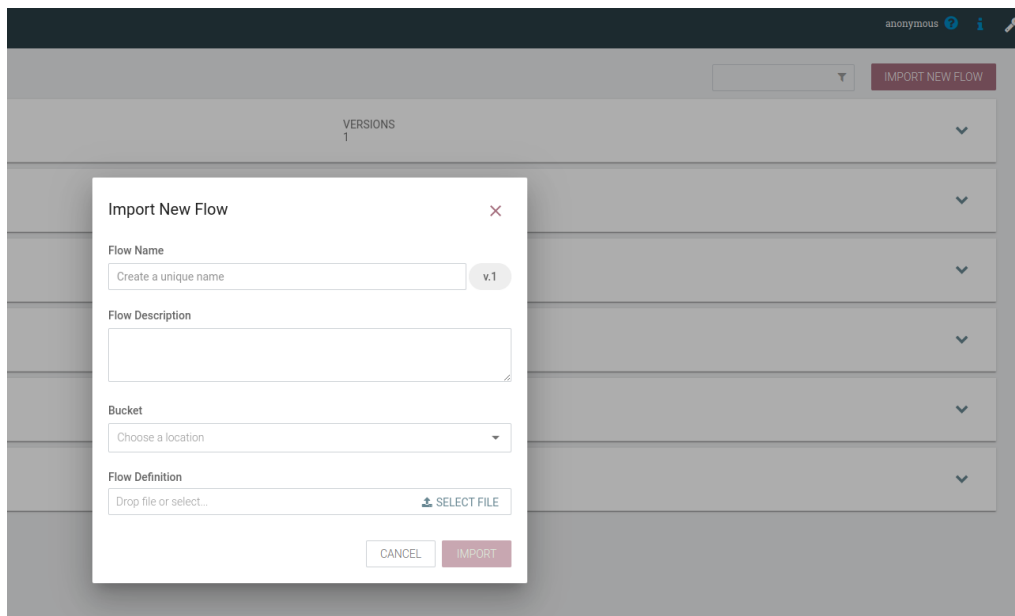
Give appropriate **bucket name** whatever you want & add description and then click on **‘CREATE’** button which is at bottom right corner.



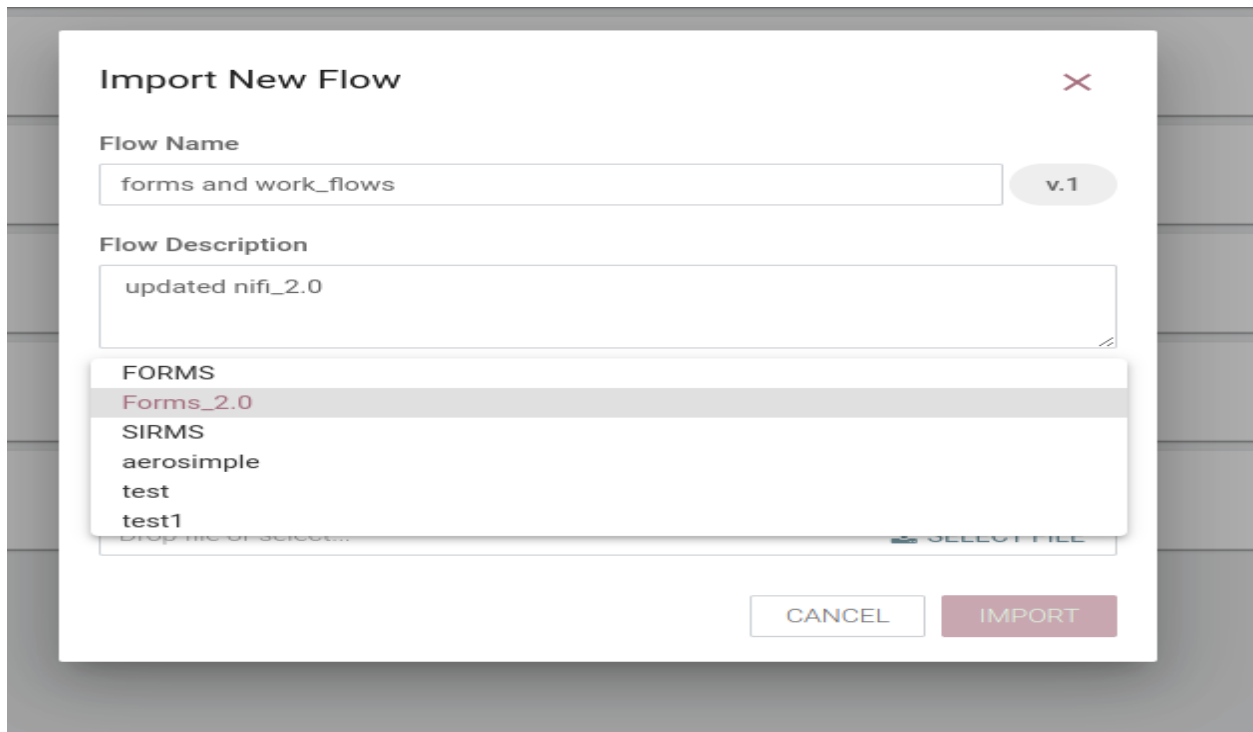
After creating bucket get back to home page by clicking on close button at the top right corner.



In home page we have **‘IMPORT NEW FLOW’** option at right side below the settings icon click on it; after clicking we will have prompt/interface.

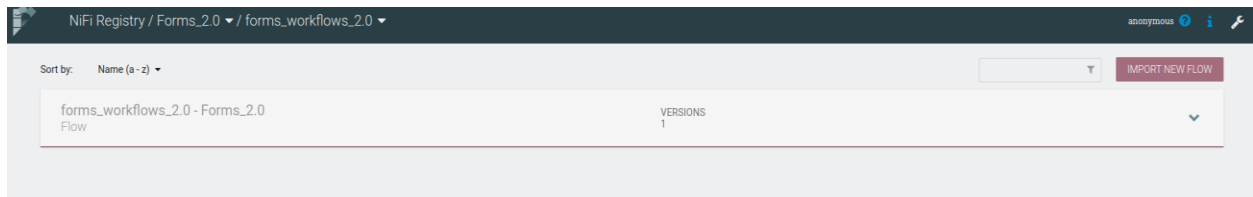


Enter all details and in **‘flow definition’** field select **Forms & Workflows JSON File** which you have previously downloaded and make sure you have given the correct bucket name which you have created previously.



The image shows a modal dialog box titled "Import New Flow" with a close button (X) in the top right corner. It contains two text input fields: "Flow Name" with the value "forms and work_flows" and a version selector "v.1" to its right; and "Flow Description" with the value "updated nifi_2.0". Below the description is a dropdown menu currently showing "FORMS", with a list of options: "Forms_2.0" (highlighted), "SIRMS", "aerosimple", "test", and "test1". At the bottom right are "CANCEL" and "IMPORT" buttons.

After selecting all details click on **‘IMPORT’** button in the bottom right corner. Then the file will be successfully imported to nifi registry in the bucket you have created.



The image shows the NiFi Registry web interface. The breadcrumb navigation at the top reads "NiFi Registry / Forms_2.0 / forms_workflows_2.0". The main content area displays a table with one row:

Sort by: Name (a - z)		IMPORT NEW FLOW
forms_workflows_2.0 - Forms_2.0 Flow	VERSIONS 1	

Click on that bucket to see the details

NiFi Registry / Forms&WorkFlows ▾ / forms&workflows ▾

Sort by: Name (a - z) ▾

forms&workflows - Forms&WorkFlows
Flow

VERSIONS
1

BUCKET IDENTIFIER
306b8a75-f116-4c90-ae7d-64dd9c479121

CHANGE LOG [↻](#)

Version 1 - a few seconds ago
by muralikrishna.k@innodata.in

FLOW IDENTIFIER
a19e5dfe-8870-4430-932f-82aa08a552cc

DESCRIPTION
forms&workflows json file downloaded from github

ACTIONS ▾

IMPORT NEW FLOW

Now get back to the home page and copy the Nifi registry URL from the web page

← → 🔍 Not secure 10.10.5.43:18080/nifi-registry/#/explorer/grid-list ☆

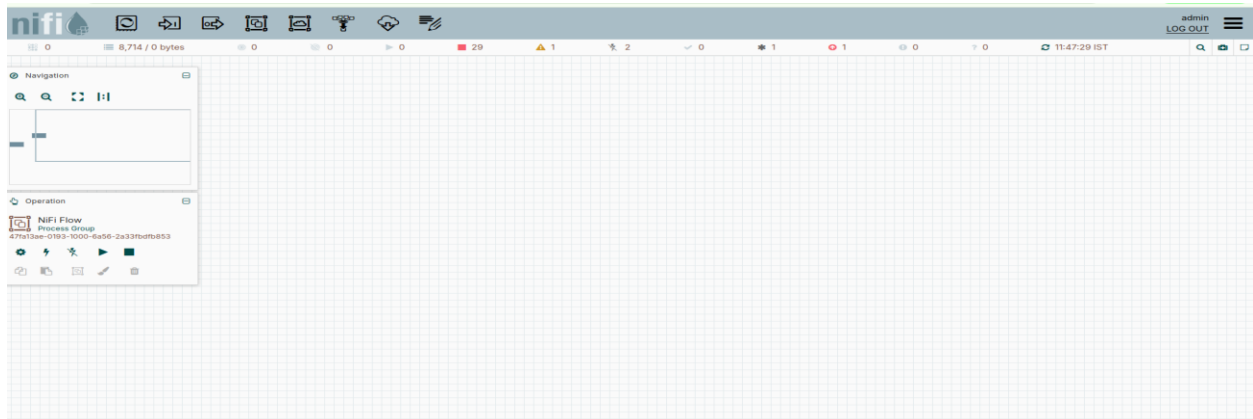
NiFi Registry / All ▾

Sort by: Name (a - z) ▾

Forms & Workflows - aerosimple Flow	VERSIONS 1
Forms & Workflows - FORMS Flow	VERSIONS 9
RetailDemo - test Flow	VERSIONS 2
ee - test1 Flow	VERSIONS 1
forms_workflows_2.0 - Forms_2.0 Flow	VERSIONS 1

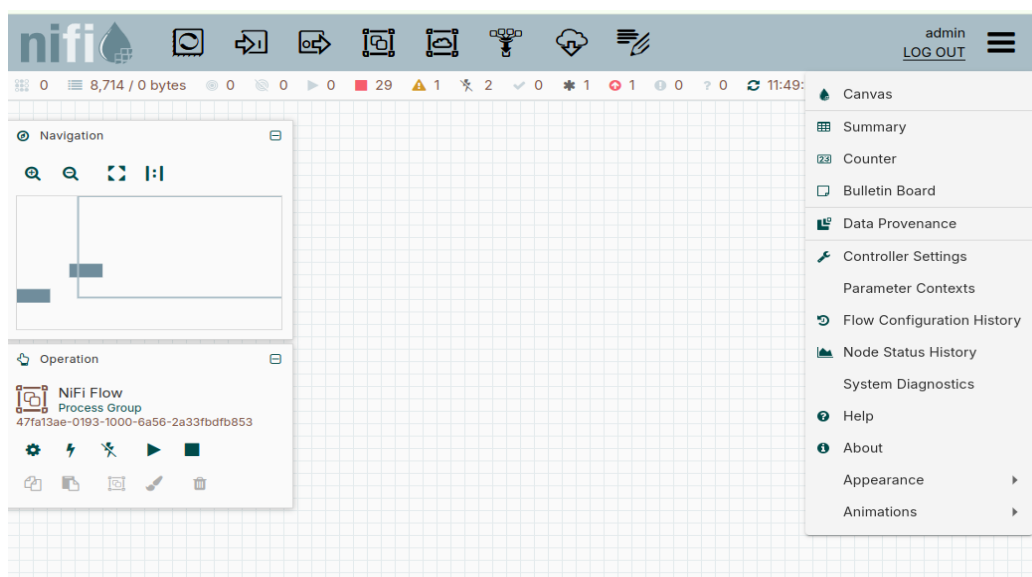
2.Importing template from Nifi Registry to Nifi:

After the above steps, open the **Nifi UI / Nifi 2.2 canvas** using credentials, you will see the interface as in below picture.



To integrate nifi registry with nifi canvas/UI follow below steps:

Please click on nifi **'GLOBAL MENU'** which is at top right corner (3 horizontal lines) it will give you options like in the below image and from these options click on **'CONTROLLER SETTINGS'** to set up the nifi registry.

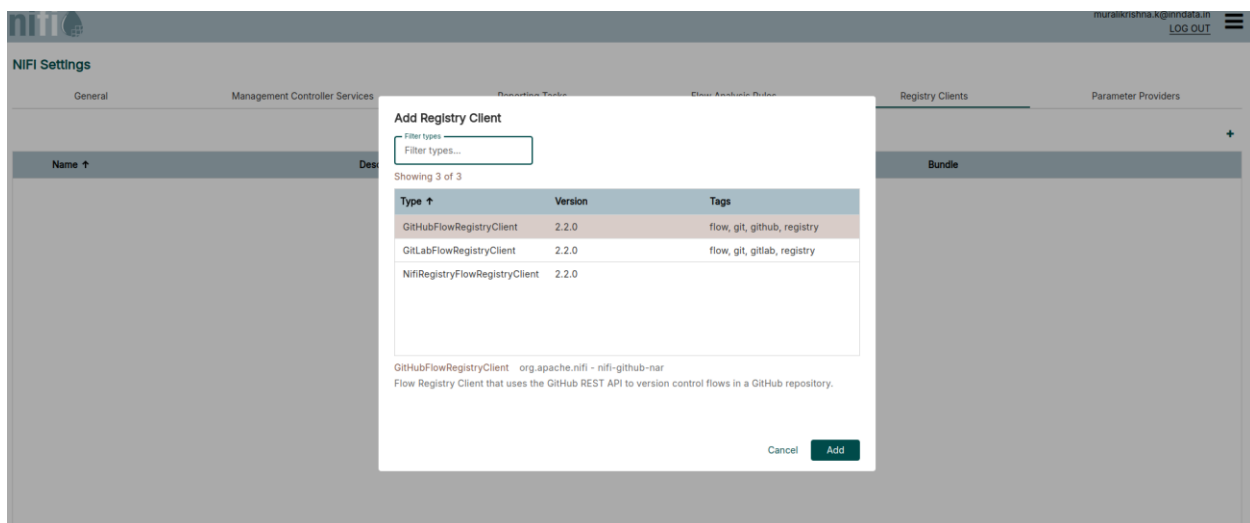


Note: To get back to nifi **home** page canvas from the middle of the any processor group/properties please select **'CANVAS'** option from global menu.

After selecting '**controller settings**' option we will have interface like below image

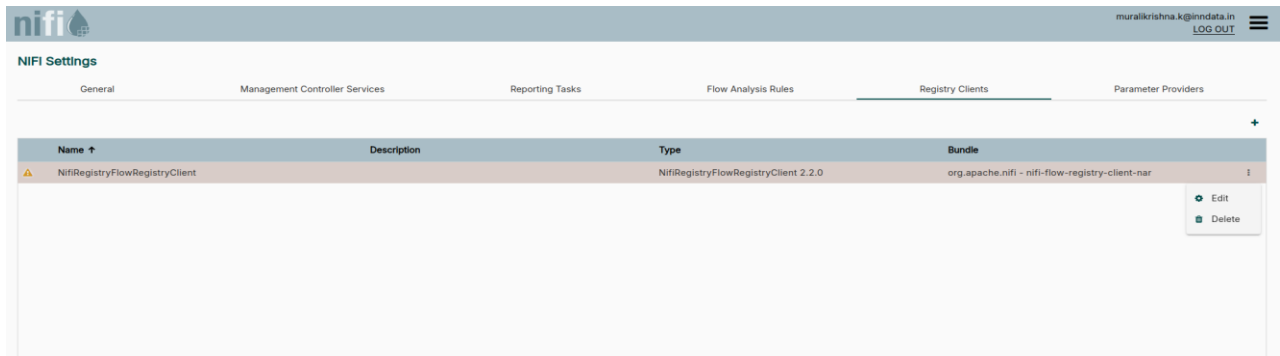


Select '**REGISTRY CLIENTS**' option like in the above image & click on '+' to add register client. And select **NifiRegistryFlowRegistryClient**

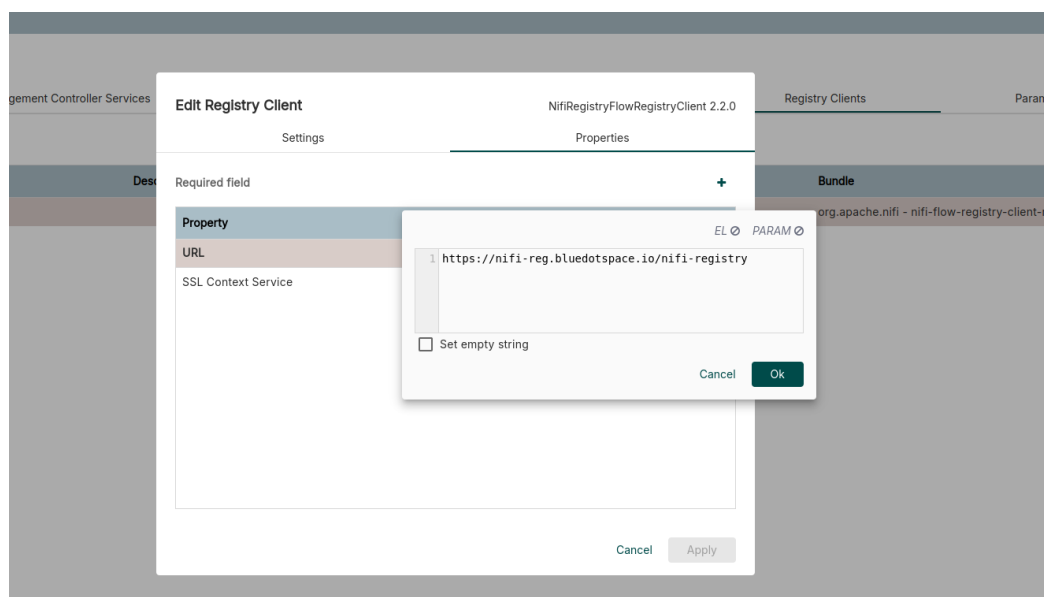


By doing the above steps you have registered the type of client you want to integrate and after this you must provide required client details to access the nifi registry.

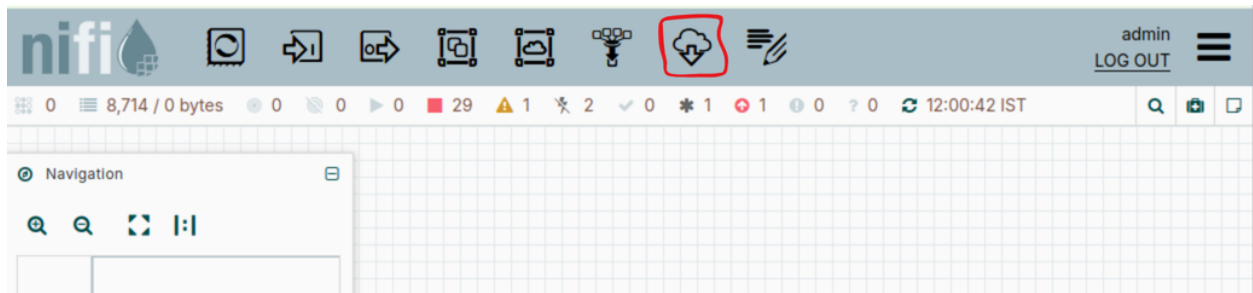
click on 3 dots at the right end of registered client you will get '**EDIT & DELETE**' options, now click on '**EDIT**' to give URL of the client/registry.



Go to properties & paste the URL we have copied from nifi registry webpage/your Nifi Registry URL, at the place of URL property value and click on '**APPLY**', we have successfully registered client.



Now go back to main Canvas, from the components tool bar drag and drop the '**IMPORT FROM REGISTRY**'



And fill the fields with appropriate data like below image

In this:

Registry means name we have given while registering client in nifi UI controller settings

Bucket means bucket name we have created in nifi registry

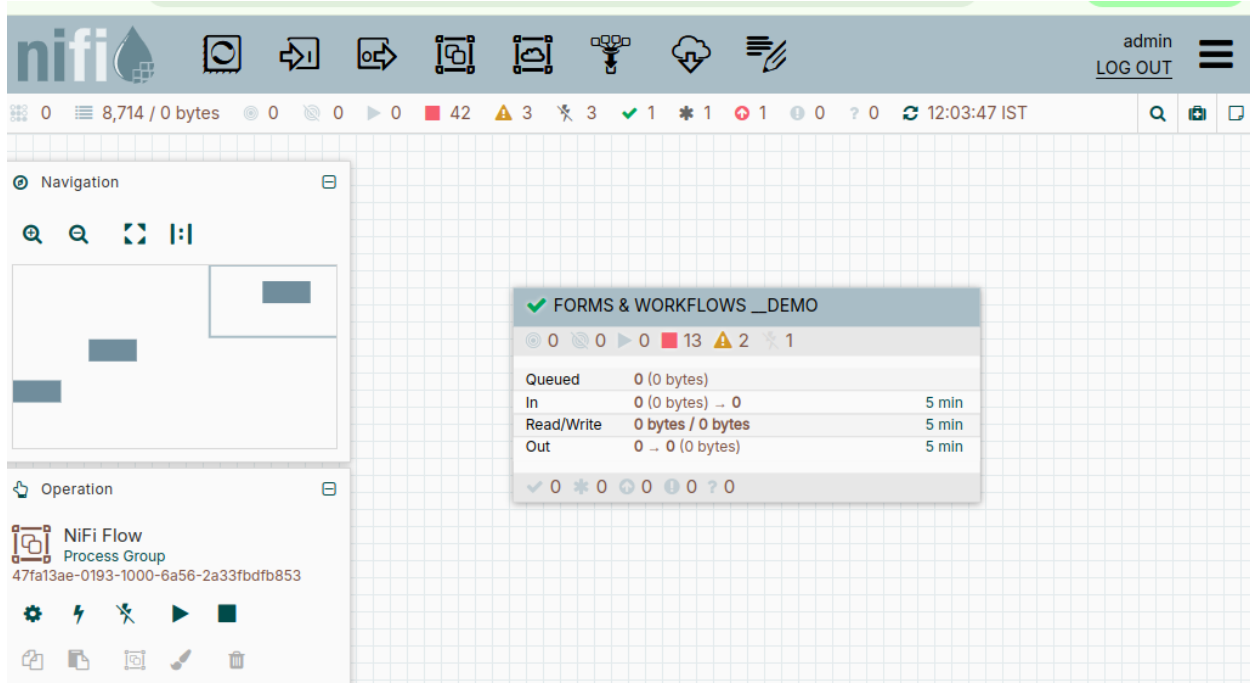
Flow means flow name we have given while importing the new flow in the nifi registry

After giving all the details select 'keep existing parameter context' and click on '**IMPORT**'.

A screenshot of the 'Import From Registry' dialog box. It contains three dropdown menus: 'Registry*' with 'forms_new', 'Bucket*' with 'Forms_2.0', and 'Flow*' with 'forms_workflows_2.0'. Below these is a checked checkbox labeled 'Keep existing Parameter Contexts'. A 'Flow Description' field contains the text 'nifi_2.0 updated'. At the bottom, there is a table with one row of data and two buttons: 'Cancel' and 'Import'.

Version	Created ↓	Comments
1	12/26/2024 11:38:23.8	

After clicking 'import' we have successfully imported the nifi template in the canvas like below



Create the GitHubFlowRegistryClient:

Please click on nifi '**GLOBAL MENU**' which is at top right corner (3 horizontal lines) it will give you options like in the below image and from these options click on '**CONTROLLER SETTINGS**' to set up the nifi registry. After selecting 'controller settings' option. Select '**REGISTRY CLIENTS**' option like in the above image & click on '+' to add register client. Now select **GitHubFlowRegistryClient**.



Now click on 3 dots at the right end of registered client, now click on **'EDIT'** to configure. Go to the properties section and fill in the values. For the **Authentication Type** property, select the **Personal Access Token** (which you need to create a personal access token in Github).

The screenshot shows the 'Edit Registry Client' dialog box with the 'Properties' tab selected. The dialog has a title bar with 'Edit Registry Client' and 'GitHubFlowRegistryClient 2.2.0'. Below the title bar are two tabs: 'Settings' and 'Properties'. The 'Properties' tab contains a table with the following data:

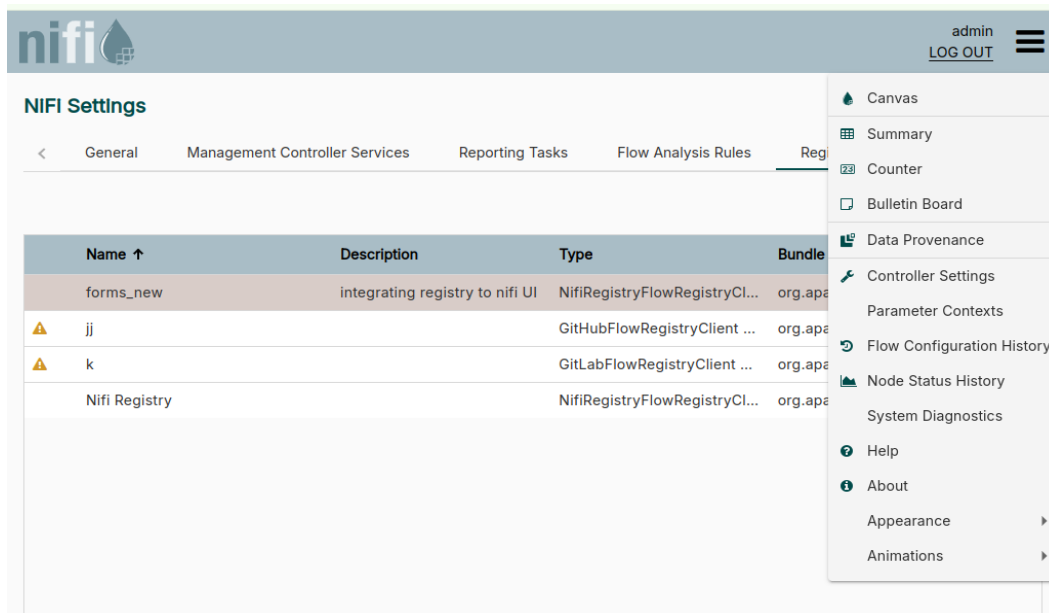
Property	Value
GitHub API URL	https://api.github.com/
Repository Owner	pavanganeshM
Repository Name	
Authentication Type	Personal Access Token
Default Branch	None
Repository Path	
Directory Filter Exclusion	Personal Access Token

The 'Authentication Type' row is highlighted, and a dropdown menu is open showing the following options: 'Personal Access Token' (selected), 'None', 'Personal Access Token' (with a checkmark), and 'App Installation'. At the bottom right of the dialog are 'Cancel' and 'Apply' buttons.

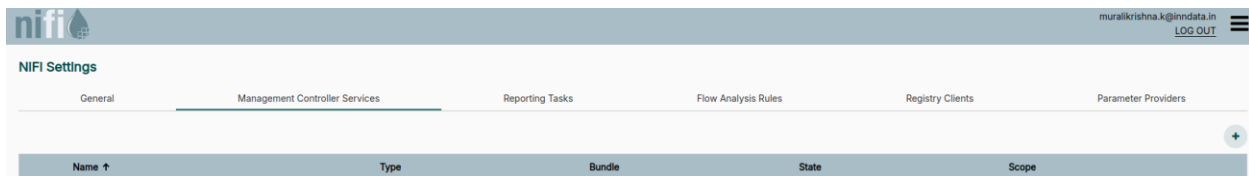
After completion click on apply. We have connected to the Github now.

3. Management Controller Services setup

Enter the Forms & Workflows_Demo processgroup and click on the ‘**GLOBAL MENU**’ which is at top right corner (3 horizontal lines) it will give you options like in the below image and from these options please click on ‘**CONTROLLER SETTINGS**’ to setup the nifi registry.



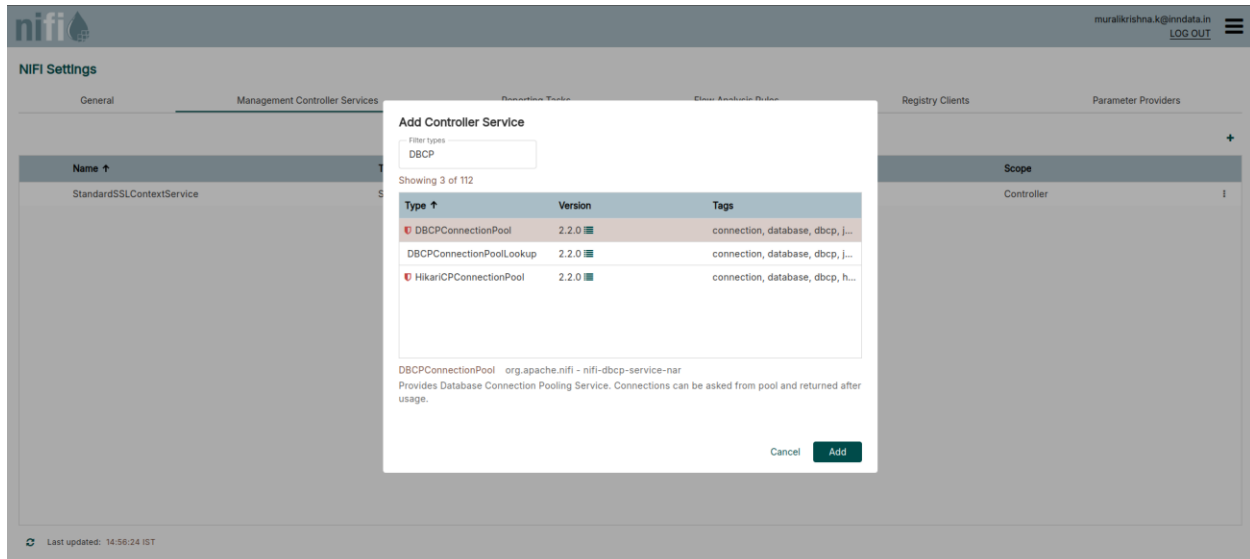
Now select the ‘**Management Controller Services**’ option & click on ‘+’ to add create a controller services.



Set Database Connection Pooling Service:

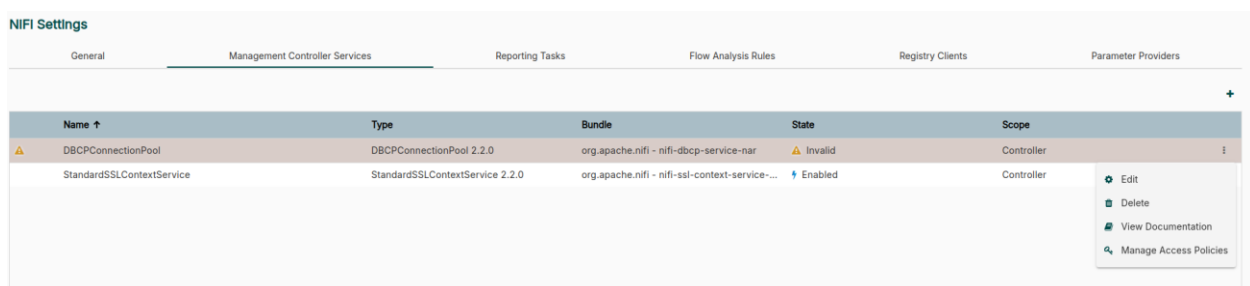
Select DBCPConnectionPool Service:

In the Compatible Controller Services section, choose "DBCPConnectionPool 2.2" and click on Add.



Open DBCPConnectionPool Configuration:

Next to the DBCPConnectionPool entry in the Database Connection Pooling Service property, click on the three dots on the right side and click on Edit.

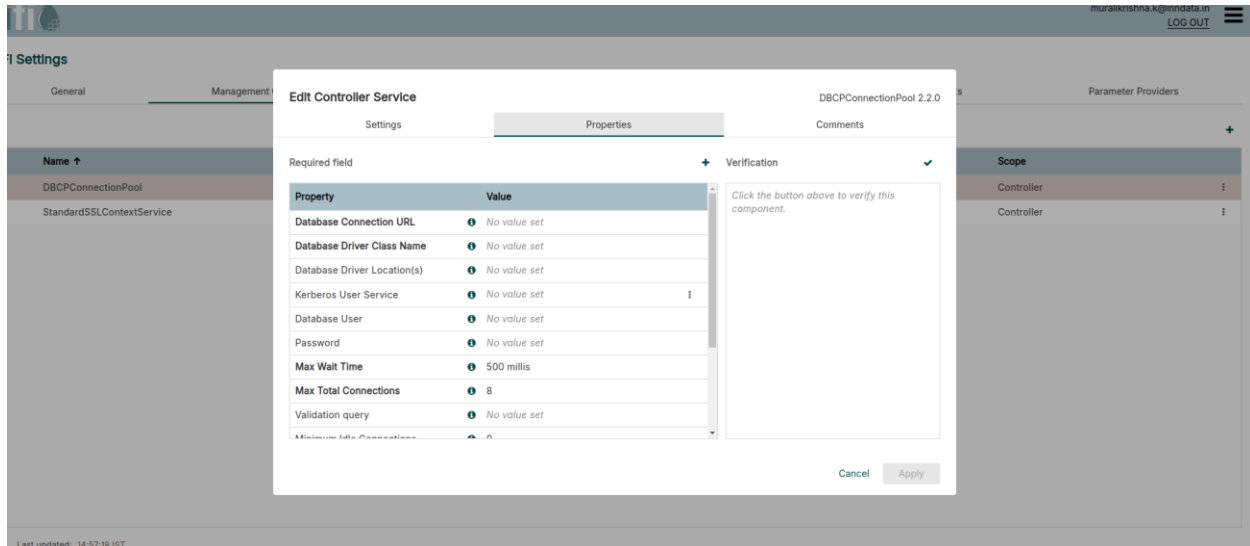


Configure Connection Pool Settings:

Go to the Properties tab and fill in the following required fields:

- **Database Connection URL:** Enter the JDBC URL to your database (e.g., jdbc:postgresql://123.0.0.1:5432/db_Name).
- **Database Driver Class Name:** Provide the fully qualified class name of the driver (e.g., org.postgresql.Driver).
- **Database Driver Location(s):** Specify a comma-separated list of paths or URLs for the driver JAR and its dependencies (e.g., /opt/nifi/lib/postgresJarName.jar).

- **Database User and Password:** Enter the credentials for connecting to the database.



Apply Configuration: After configuring the properties, click Apply.

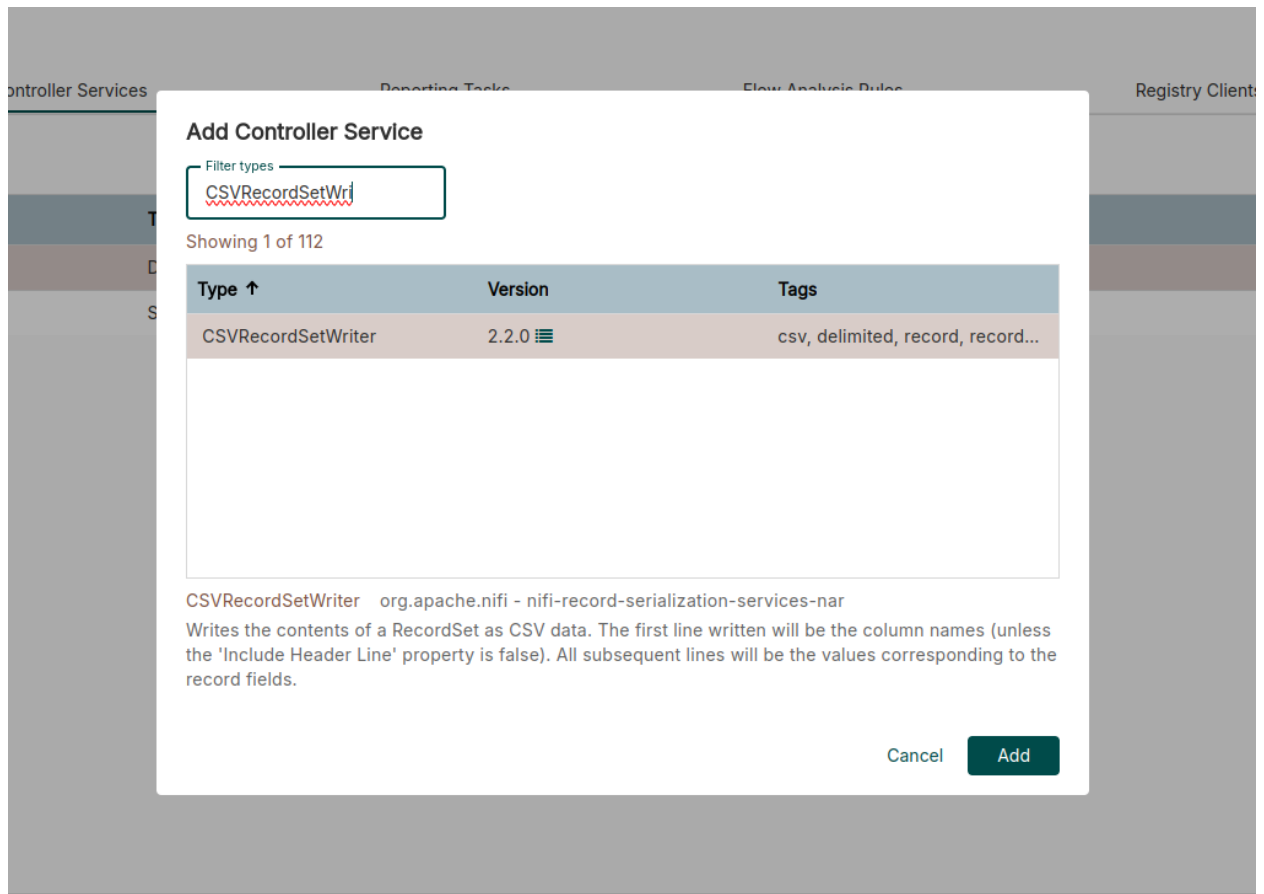
Enable the Service: Click on the Enable button (lightning bolt symbol) next to DBCPConnectionPool to activate it and select 'Service and Reference components' in Scope before enable.

Note: This DBCPConnectionPool is for fetching the data from the Source Database. For easy reference in future rename this service as 'Source database connection' or any other name for reference from Settings.

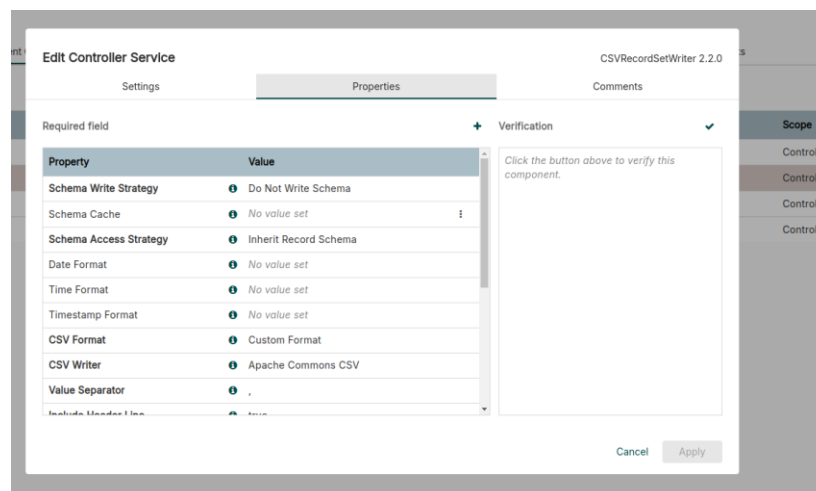
****** In the same way we need to create another DBCPConnectionPool Service for the database which is maintaining the cached date table. And rename the connection pool for reference ******

Create New Service for CSV Record Writer:

Click on ‘+’ to add create a CSVRecordWriter , and choose **CSVRecordSetWriter** from the list of available services.



Click on the three dots on the right side and click on **Edit** to configure.



Set the following properties as needed:

Schema Access Strategy:

Set this to Inherit Record Schema to use the schema from the incoming record.

Date Format, Time Format, Timestamp Format:

Date Format: Leave it, as if there's no custom format.

Time Format: Leave it, as if there's no custom format.

Timestamp Format:

Set to yyyy-MM-dd HH:mm:ss.SSS Z for the specified timestamp format.

CSV Format:

Choose Custom Format since you're specifying a custom delimiter and formatting.

Value Separator:

Set this to µ for a custom value separator.

Include Header Line:

Set this to true to include a header line in the output CSV.

Quote Character:

Set this to " (double quotes) to wrap fields in quotes if required.

Escape Character:

Set this to ^ to escape special characters.

Comment Marker:

Leave this empty if there's no specific character to mark comments.

Null String:

Leave this empty if there's no specific value needed for null fields.

Trim Fields:

Set this to true to remove any leading or trailing whitespace from fields.

Quote Mode:

Set to Do Not Quote Values to avoid quoting fields unless necessary.

Record Separator:

Set this to \n to use a newline as the record separator.

Include Trailing Delimiter:

Set this to false to avoid adding a trailing delimiter after the last record.

Character Set:

Set to UTF-8 for compatibility with most applications.

Apply Configuration: After configuring the properties, click Apply.

Enable the Service: Click on the Enable button (lightning bolt symbol) next to CSVRecordSetWriter to activate it and select 'Service and Reference components' in Scope before enable.

Create New Service for CSVReader:

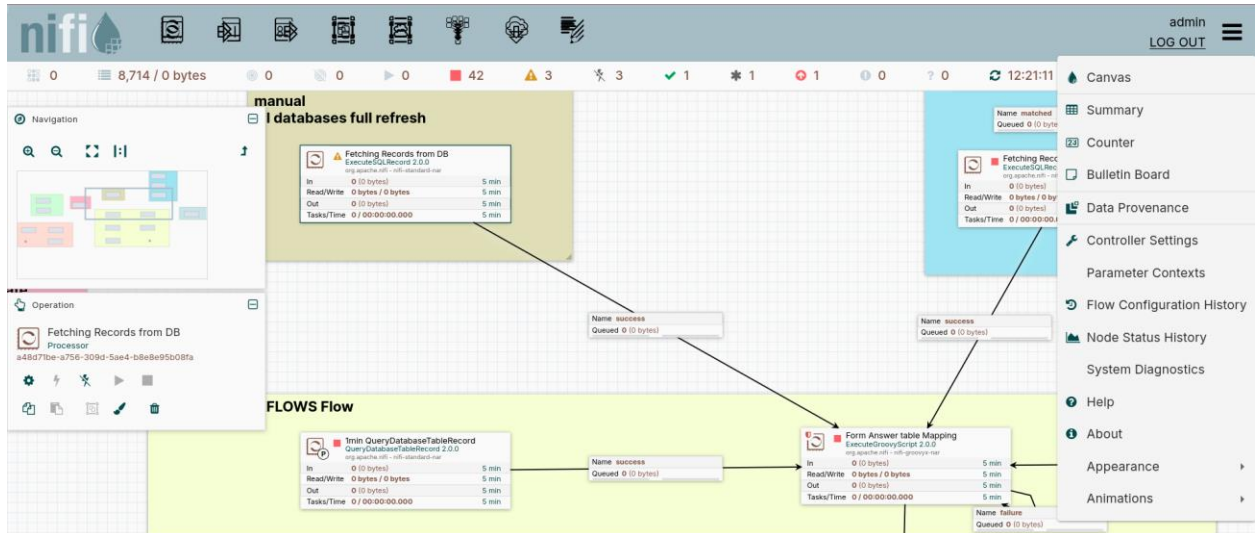
Click on '+' to add create a CSVReader, and choose CSVReader from the list of available services. For this service we do not need to configure anything leave the default values.

Enable the Service: Click on the Enable button (lightning bolt symbol) next to CSVReader to activate it and select 'Service and Reference components' in Scope before enable.

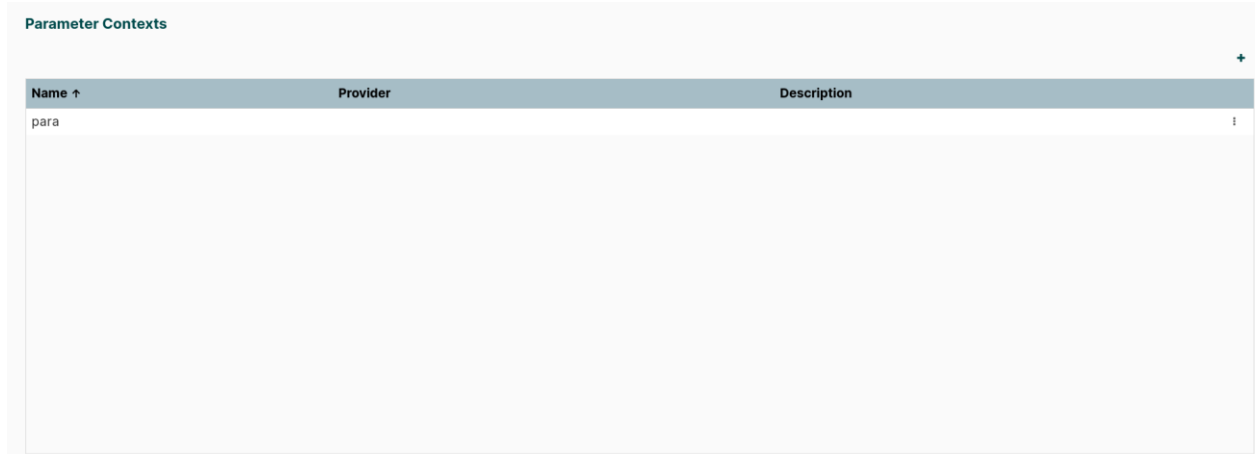
4.Parameter Contexts setup

Steps to Add Parameters in NiFi:

1. In the top-right corner, click on the **Global Menu**.



2. From the dropdown, select **Parameter Contexts**.
3. Click the '+' button to create a new Parameter Context.



4. In the **Settings** tab, provide a **Name** for the Parameter Context. Move to the **PARAMETERS** tab, then click the '+' button.

The screenshot shows a dialog box titled "Add Parameter Context" with three tabs: "Settings", "Parameters", and "Inheritance". The "Parameters" tab is active. It features a table with two columns: "Name ↑" and "Value". Above the table is a "+" button. To the right of the table is a section labeled "Parameter" with the value "None" and a "Referencing Components" link with an information icon. At the bottom right are "Cancel" and "Apply" buttons.

Name ↑	Value
--------	-------

5. Specify the **Name** and **Value** for each parameter as below to add:

- i. **delimiter** = 'μ'
- ii. **Source_DB_URL** = The JDBC URL to your Source database (for e.g., jdbc:mysql://hostname:port/database).
- iii. **Source_DB_User** = Enter the credentials for connecting to the Source database.
- iv. **Source_DB_Pwd** = Enter the credentials for connecting to the Source database.
- v. **Target_DB_URL** = The JDBC URL to your Target/Destination database (for e.g., jdbc:mysql://hostname:port/). Database is not required for this parameter as we create databases/tables using this connection.
- vi. **Target_DB_User** = Enter the credentials for connecting to the Target/Destination database.
- vii. **Target_DB_Pwd** = Enter the credentials for connecting to the Target/Destination database.
- viii. **DBname** = Enter the airport code you want to refresh for the particular database (e.g., *VOGA*, *KDSM*).
- ix. **Form_id** = Enter the required 'form_parent_id' here to refresh the specific form, If you do not want to specify a form_parent_id, enter **NULL** in this field to ensure the flow runs successfully and it will refresh entire database if you put null in this place (e.g., 12176,21).

- x. **Maintain_history_config_flag** = This flag is used to maintain the delete record history. You need to provide a Boolean value (e.g., **true**, **false**).

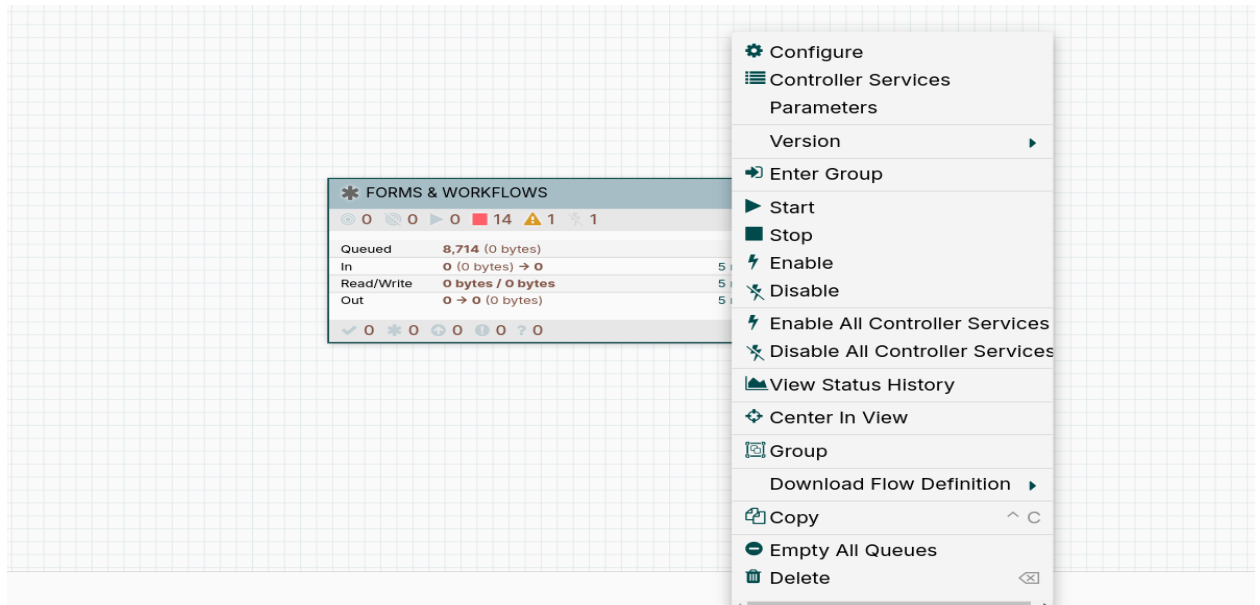
NOTE: please check that script must contain parameters with those names only.

The screenshot shows a software interface with a modal dialog box titled "Add Parameter". The dialog has a "Name*" field with a red border and a red error message "Property name is required." below it. Below the name field is a "Value" field with an information icon. There is a checkbox labeled "Set empty string" which is currently unchecked. Below that is a "Sensitive Value" section with two radio buttons: "Yes" and "No", with "No" being selected. At the bottom of the dialog is a "Description" field with a green circular icon and a slash. The dialog has "Cancel" and "Ok" buttons at the bottom right. In the background, there is a "Settings" panel with a "Name" header and an "Add Parameter Context" button. To the right, there is an "Inheritance" panel and a "Cancel" button.

6. After adding all parameters, click **Apply** to save.

Applying the Parameter Context to a Process Group:

1. Open the main **workflow canvas**.
2. Select the **Process Group** where you want to apply the Parameter Context.
3. Right-click on the Process Group and choose **Configure** (or click the settings icon).



4. In the **Parameter Context** dropdown, select the newly created Parameter Context, and click apply.

Edit Process Group

Settings

Comments

Name*
FORMS & WORKFLOWS

Parameter Context
para

☐ Apply Recursively ⓘ

Execution Engine*
Inherited

Process Group FlowFile Concurrency*
Unbounded

Process Group Outbound Policy*
Stream When Available

Default FlowFile Expiration ⓘ*
0 sec

Default Back Pressure Object Threshold ⓘ*
10000

Default Back Pressure Data Size Threshold ⓘ*
1 GB

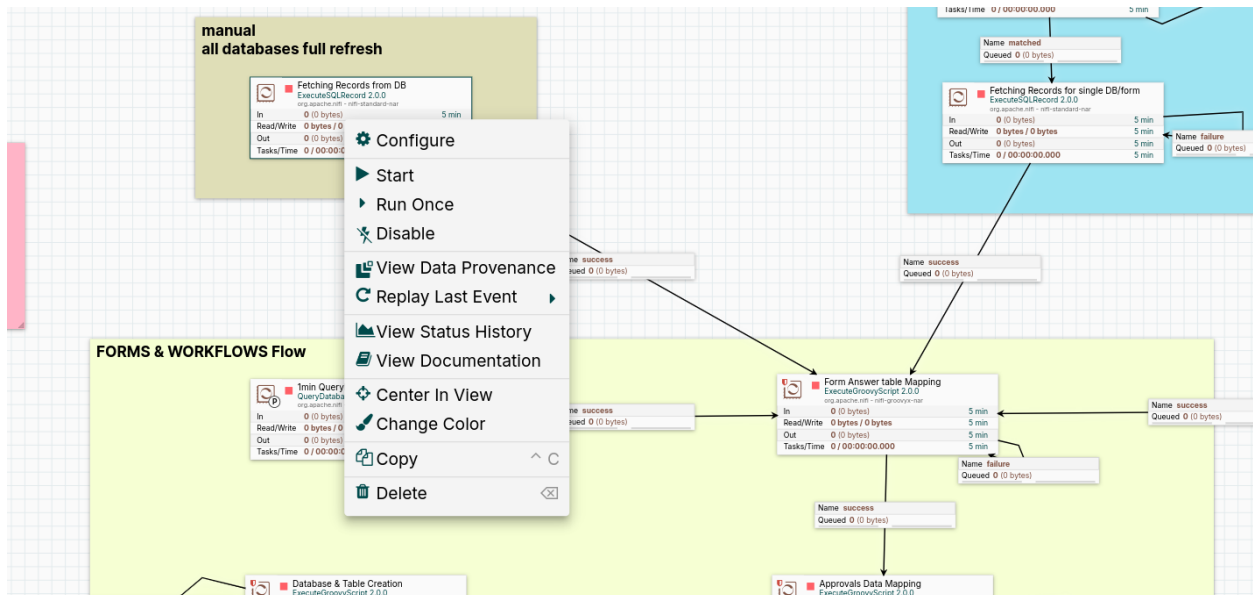
Log File Suffix ⓘ

Cancel

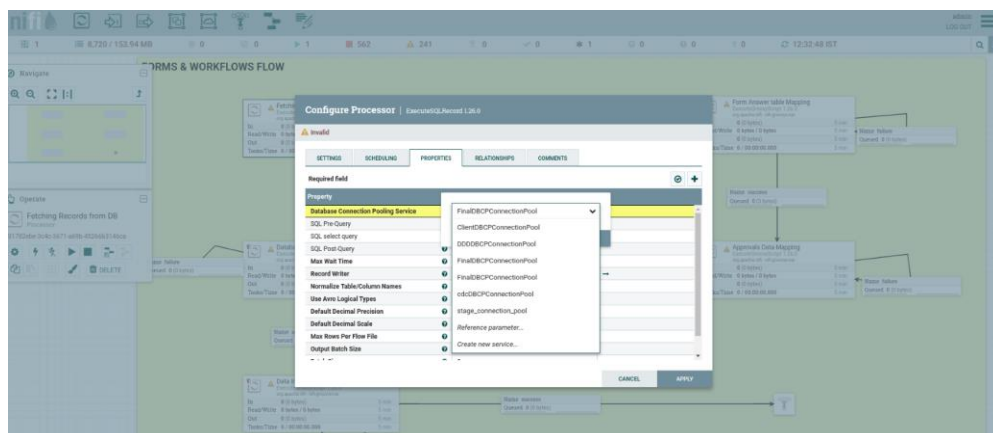
Apply

5.Nifi flow configuration

Once the template is uploaded, you'll see the NiFi flow. Right-click on the processor named "Fetching Records from DB" and select "Configure".

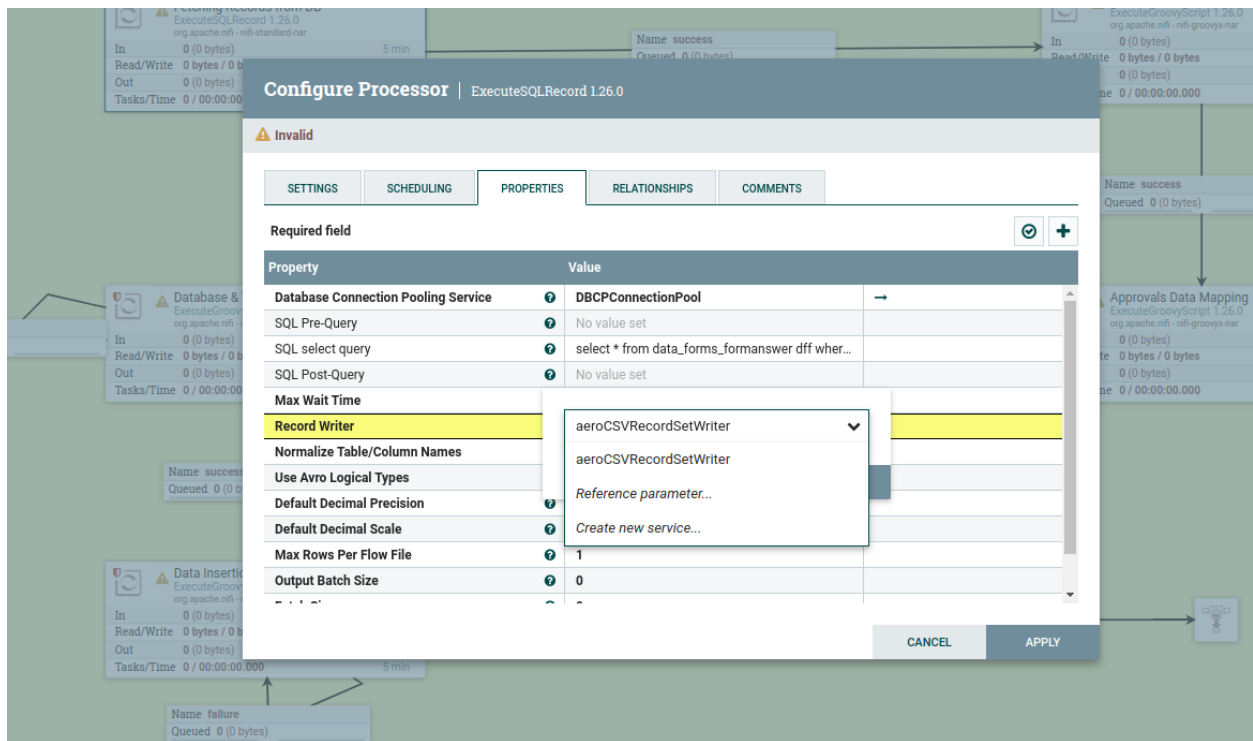


Go to the Properties tab. For the Database Connection Pooling Service property, select the value dropdown we will find the **DBCPCConnectionPool Service** for the Source database



Click on the **Value** field for **Record Writer**.

- From the dropdown, select **Create New Service**.
- Choose **CSVRecordSetWriter**, which we have created.



Note: In the ExecuteSqlRecord/QueryDatabaseRecord processor we must set property named 'Use Avro Logical Types' to 'true' to get desired time format as output using csv record writer

Click on apply.

6.Maintaining Cached Date in a Separate Table:

To maintain the cached date, you need to create a table that stores the **max(updated_at)** columns date from 'data_frms_formanswer' table. Below is the SQL script to create this table.

```
CREATE TABLE public.date_table (  
    id SERIAL PRIMARY KEY, -- Unique identifier for each record  
    max_date TIMESTAMP NOT NULL -- Stores the maximum updated_at date  
);
```

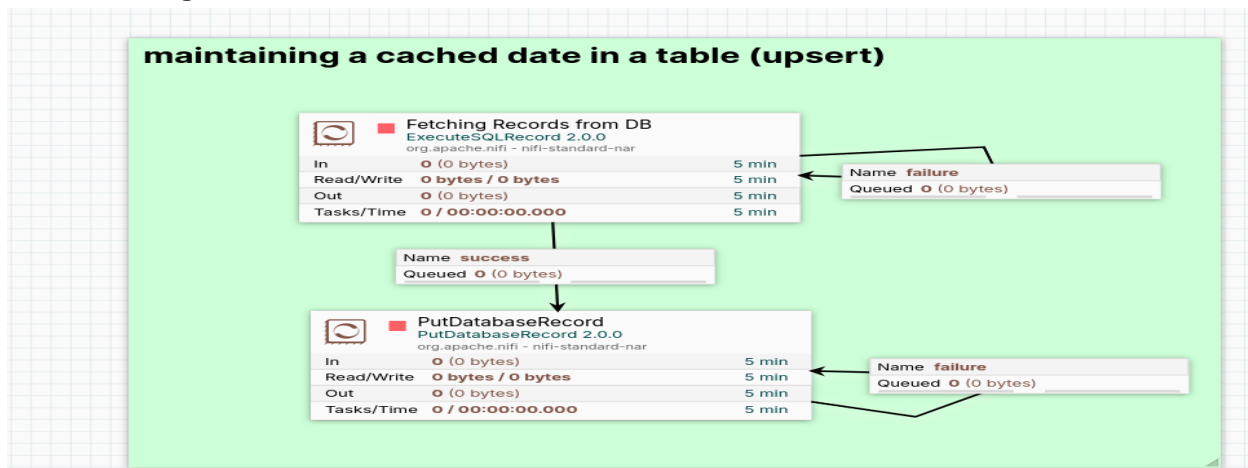
Note:

The table for maintaining the cache date can be created in a location of your choice, depending on your database setup, **we have used the source database for this table.**

Replace date_table with your desired table name if needed.

Ensure to update the corresponding processor properties with the new table name to maintain consistency.

1.below image will show the flow related to this :



2. below image shows the properties of first processor (ExecuteSQLRecord):

Edit Processor ExecuteSQLRecord 2.0.0

Settings Scheduling **Properties** Relationships Comments

Required field + Verification ✓

Property	Value
Database Connection Pool...	AERO_DBCPConnectionPool
SQL Pre-Query	No value set
SQL select query	select 1 as id, max(updated_a...
SQL Post-Query	No value set
Max Wait Time	0 seconds
Record Writer	AERO_CSVRecordSetWriter
Normalize Table/Column N...	false
Use Avro Logical Types	true

Click the button above to verify this component.

Cancel Apply

3. Below image shows the properties of second processor (PutDatabaseRecord):

Edit Processor PutDatabaseRecord 2.0.0

Settings Scheduling **Properties** Relationships Comments

Required field + Verification ✓

Property	Value
Record Reader	CSVReader
Database Type	PostgreSQL
Statement Type	UPSERT
Data Record Path	No value set
Database Connection Pooli...	AERO_DBCPConnectionPool
Catalog Name	aerosimple_test
Schema Name	public
Table Name	date_table

Click the button above to verify this component.

Cancel Apply

--> 'catalog name' means database name.

--> For **PutDatabaseRecord** we will give the Database connection pool service of the Database which is maintaing the cached data table.

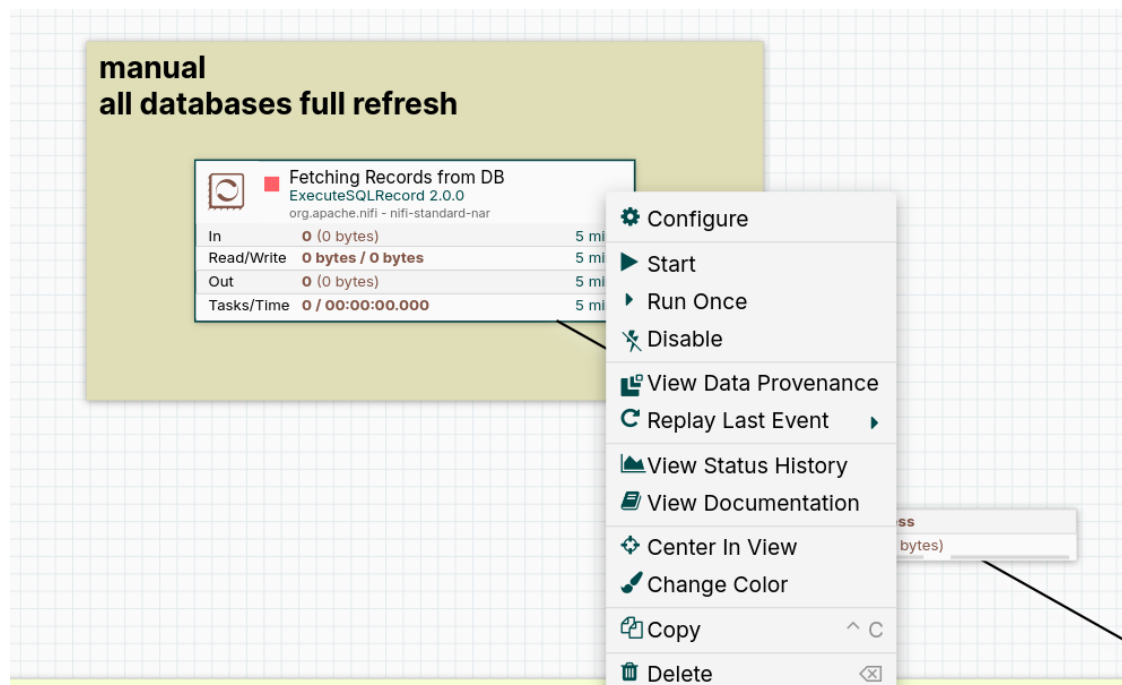
Note: This flow should be run simultaneously with the flow 'Automatic refresh of Databases based on Cached date'

7.Steps to Run the Complete Flow :

This below 4 out of 5 steps are based on our requirement, you can select which is suitable for your operation but the final 5th step is common for this all 4 steps (this final 5th step includes the processors 'formanswer_table_mapping', 'approvals_mapping', 'database & table creation', 'data insertion').

1. For manual full refresh of all databases:

Right-click on the processor named "**Fetching Records from DB**" and select "**Run Once**", Below image shows the details about it.



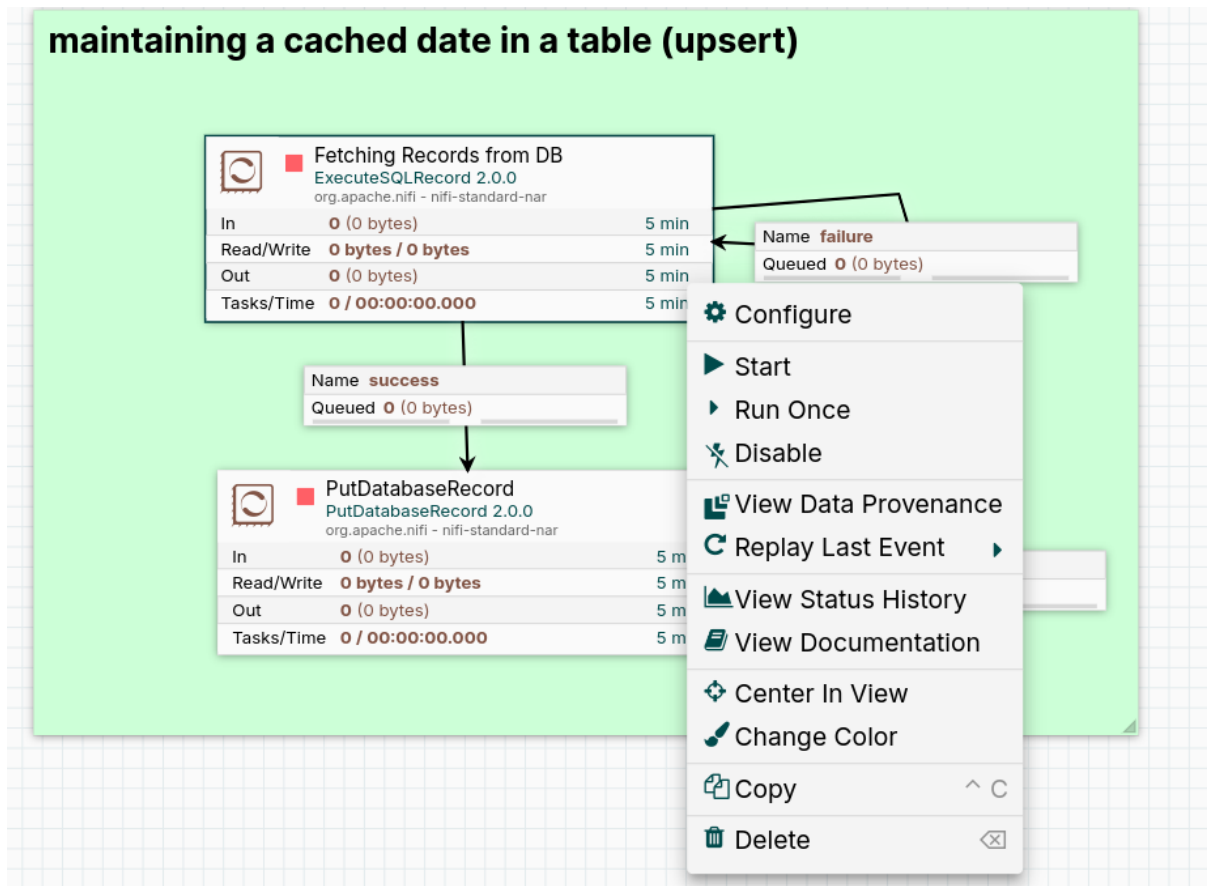
2. For Automatic refresh of Databases based on Cached date :

For automatic refresh, we are using the **QueryDatabaseRecord** processor in Apache NiFi, which is configured to run on a **cron schedule** with a 1-minute interval.

records we should also start the flow '**maintaining a cached date in a table (upsert)**' which is used to store the Cached date in the separate table named '**date_table**' we are created in the source db to store the max(updated_at) columns value .

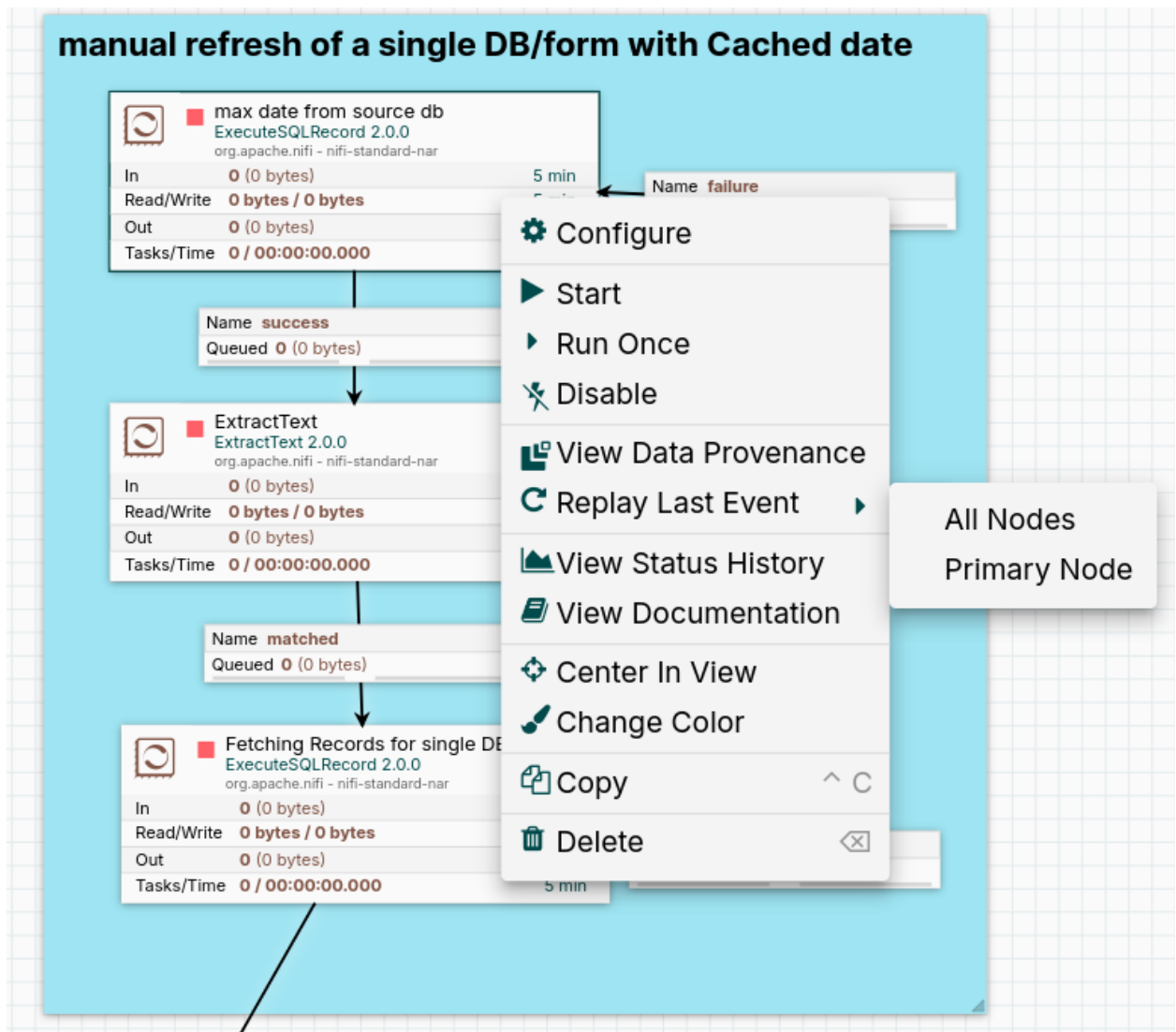
To start this flow ('**maintaining a cached date in a table (upsert)**') follow below steps :

- A) Right click on 1st processor '**Fetching Records from DB**' and click '**start**' this processor is also scheduled with 1 minute interval using nifi cron scheduling
- B) **Next**, Right click on 2nd processor '**PutDatabaseRecord**' and click '**start**'



3.Manual refresh of single DB/FORM with Cached date :

- **Right-click** on the first processor: **Max Date from Source DB**.
Click **'Run Once'**.
- **Right-click** on the second processor: **Extract Text**.
Click **'Run Once'**.
- **Right-click** on the third processor: **Fetching Records for single DB/form**.
Click **'Run Once'**.



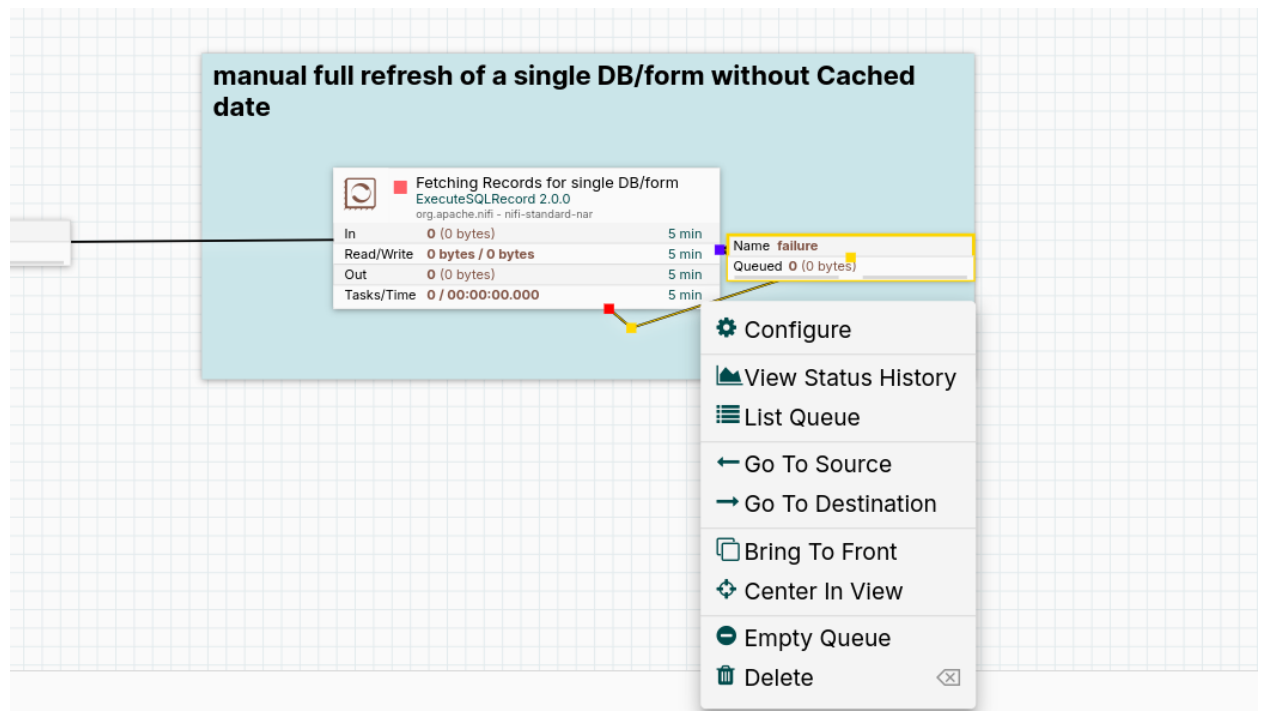
Note:

1. This process will fetch **only the records inserted after the cached date**.

2. we must specify the values to the parameters **DBname & Form_id** before clicking 'run once' on 3rd processor named '**Fetching Records for single DB/form**'.

4. Manual full refresh of a single DB/form without Cached date:

- **Right-click** on the processor named '**Fetching Records for single DB/form**'
Click '**Run Once**'.



Note:

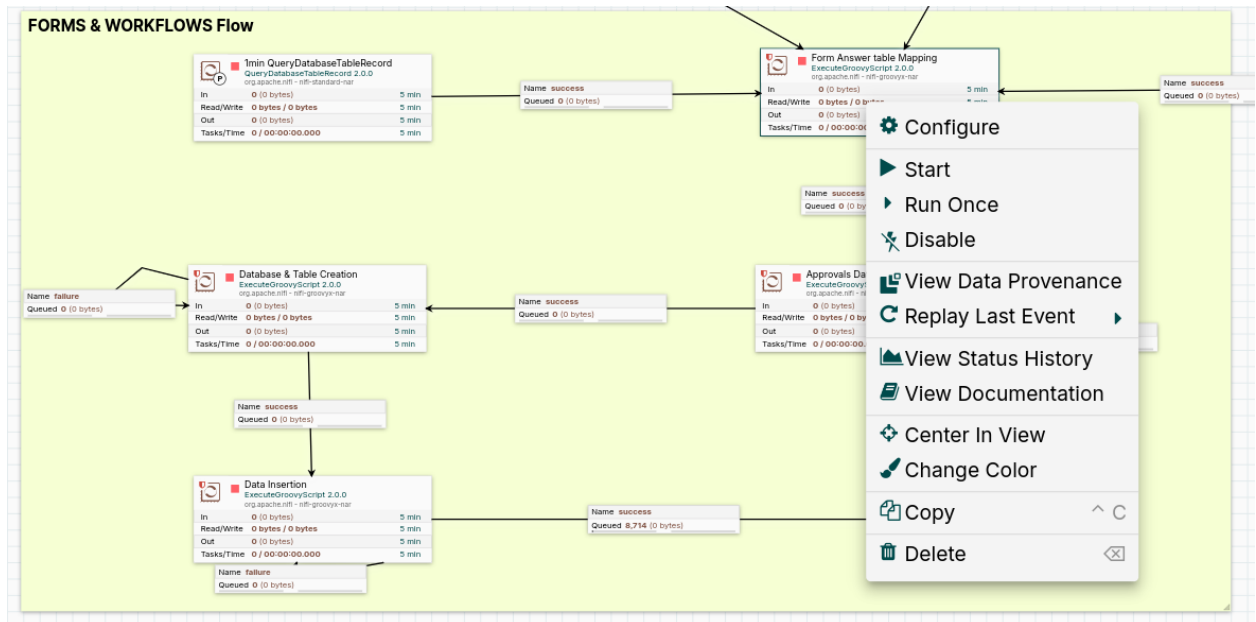
1. In this 4th step full database/form will be refreshed from the beginning because of not using cached date
2. we must specify the values to the parameters **DBname & Form_id** before clicking 'run once' on '**Fetching Records for single DB/form**'.

After selecting one of the methods from the above 4 steps please perform this below final 5th step.

5. right-click on each of the following processors in sequence and select "**Start**" for each:

a. Form Answer Table Mapping

- b. Approvals Data Mapping
- c. Database & Table Creation
- d. Data Insertion



This Flow will Map & create tables for all records from the data_forms_formanswer table where parent_id is not null.

For the Manual delete flag maintenance flow :

- Follow this below steps :

First Run:

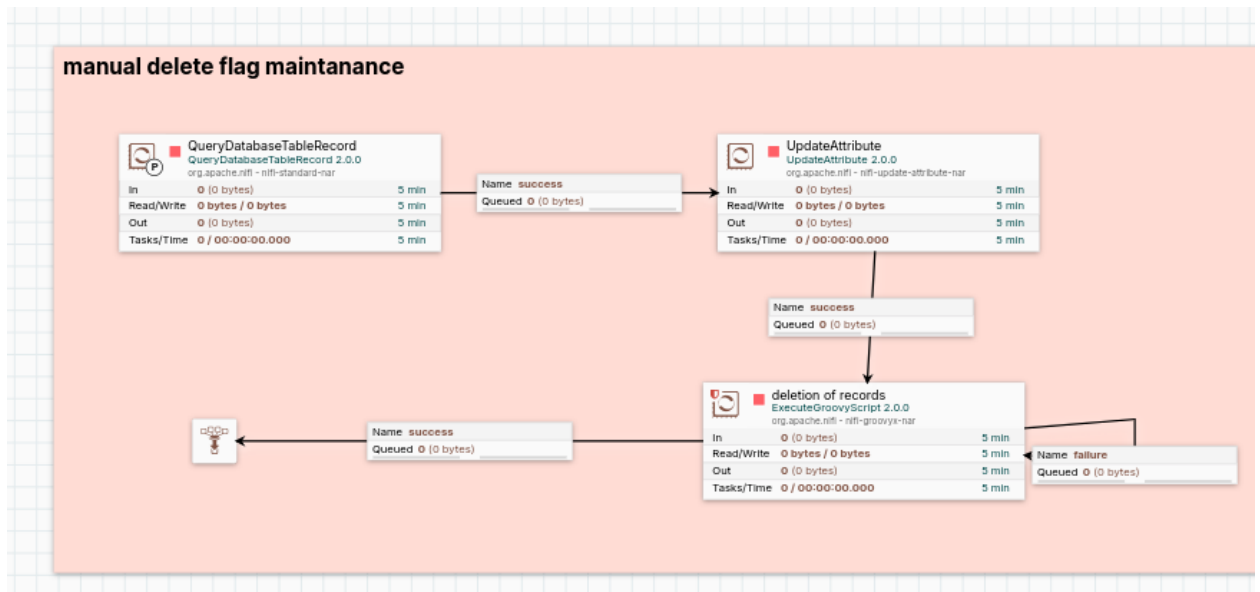
Right-click on the processor named '**QueryDatabaseRecord**' and select '**Run Once**'.

This will fetch all the records from the data '**data_forms_dataformsrecyclebin**' table on the first run.

Subsequent Runs:

After the first run, the processor will store the **cached date** of the records. From the next run onward, it will fetch only the records with a timestamp **greater than this cached date**.

This below image is the flow for delete



After the run once of first processor you have to give boolean value true/false in the global parameter **'Maintain_history_config_flag'** based on your requirement

If true is given:

- The **is_deleted** column in the target table will be updated to **true**.

If false is given:

- The **is_deleted** column in the target table will be **deleted**.

So, make sure you have given proper true/false value to the **'Maintain_history_config_flag'**

Edit Processor UpdateAttribute 2.0.0

Settings Scheduling **Properties** Relationships Comments

Required field + Verification ☒

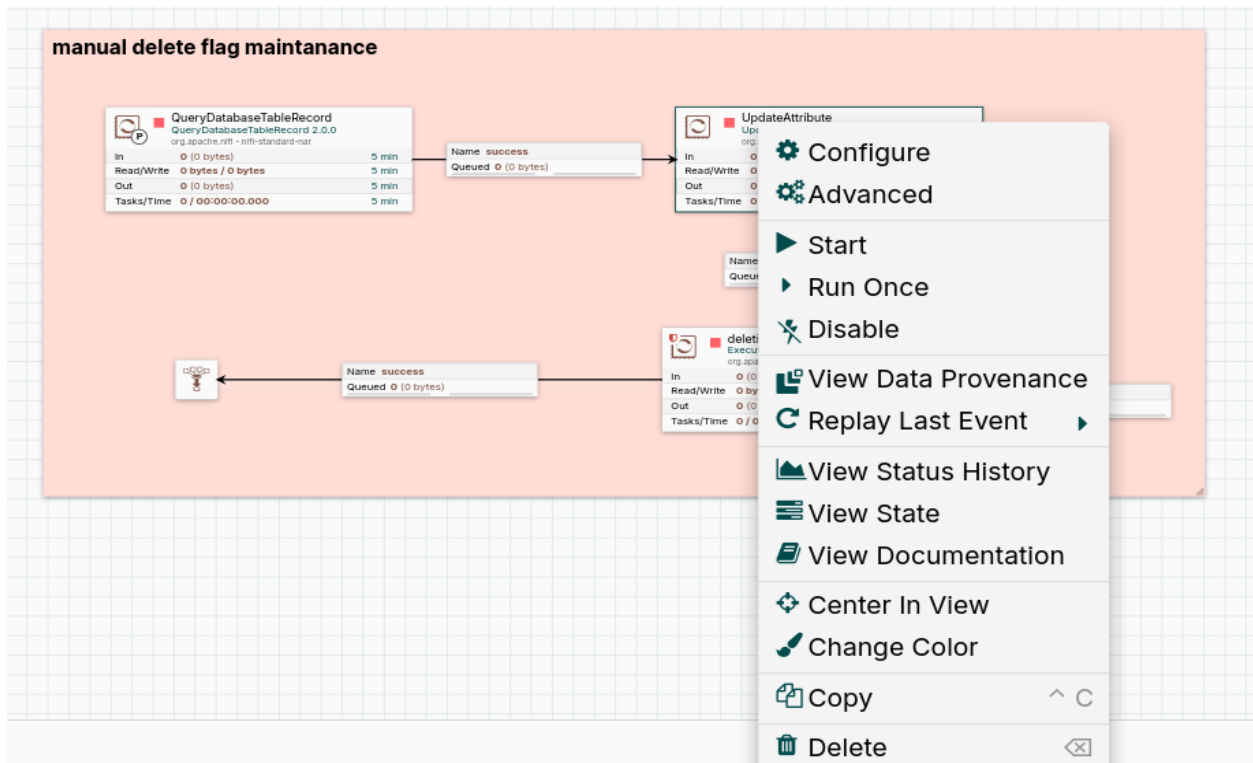
Property	Value
Delete Attributes Expression	No value set
Store State	Do not store state
Stateful Variables Initial Value	No value set
Cache Value Lookup Cache ...	100
Maintain_history_config_flag	#{Maintain_history_config_flag}

Click the button above to verify this component.

Cancel Apply

this flag is maintained in the 'updateattribute' processor

Now right click on processor named 'updateAttribute' and click '**start**'



And then Right click on 3rd processor named '**deletion of records**' and click '**start**'