CLOUDERA

# Transparency and Visibility
# NiFi Monitoring Implementation Steps

---

## Implementation

Implementing NiFi monitoring involves these tasks:

1. Create external Hive table
2. Build and deploy NiFi Flow
3. Connect the table to a visualization tool
4. Build a dashboard
5. Implement end-user alerting mechanism

- ### Create External Hive Table

  How it works: Create an external Hive table where the processor failure data is stored on the HDFS storage layer.

  Steps to Perform:

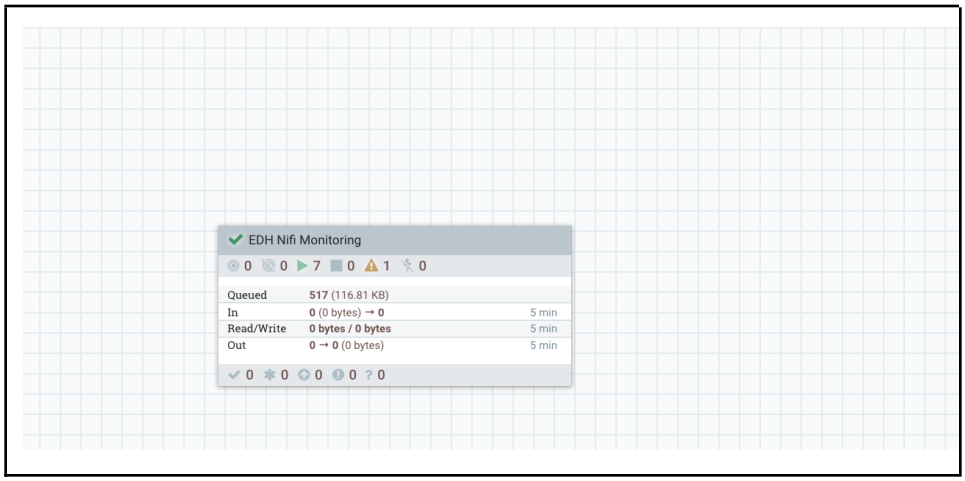
```
CREATE EXTERNAL TABLE edhoperations.edh_nifi_monitor (   datetime1 STRING,
loglevel STRING,   processor STRING,   processor_id STRING,   error STRING )
PARTITIONED BY (   snapshottime BIGINT ) STORED AS PARQUET LOCATION
'hdfs://nameservice1/user/hive/warehouse/edhoperations.db/edh_nifi_monitor'
```

- ### Build and Deploy  NiFi Flow

  How it works: The Nifi flow uses a python script to retrieve the failed processor details from the Nifi app logs every 3 hours and imports the data in the HDFS location in CSV format which is accessed by the Hive table for querying.

Steps to Perform:

1. Drag the **Processor Group** in the NiFi canvas to create a new NIFI flow in the UI with the name - NIFI Processor Monitoring.

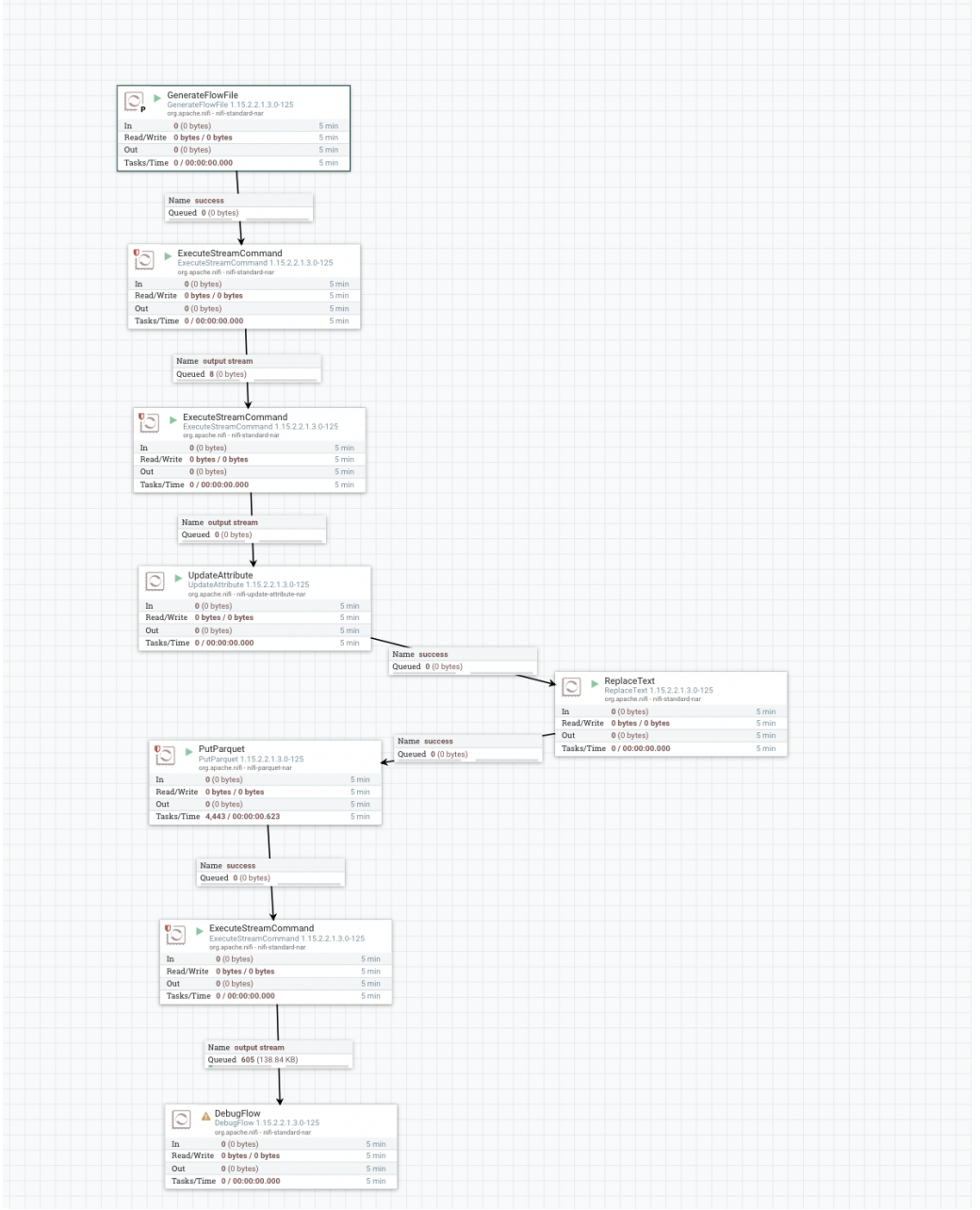



2. Creating the NiFi Flow:

Go inside the process group to create the NiFi flow. For deploying this flow, we are using the below processors:

a. GenerateFlowFile
b. UpdateAttribute
c. ExecuteStreamCommand
d. ReplaceText
e. PutParquate

The flow will look like the below:


GenerateFlowFile
GenerateFlowFile 1.15.2.2.1.3.0-125
org.apache.nifi - nifi-standard-nar
In 0 (0 bytes) 5 min
Read/Write 0 bytes / 0 bytes 5 min
Out 0 (0 bytes) 5 min
Tasks/Time 0 / 00:00:00.000 5 min
Name success
Queued 0 (0 bytes)
ExecuteStreamCommand
ExecuteStreamCommand 1.15.2.2.1.3.0-125
org.apache.nifi - nifi-standard-nar
In 0 (0 bytes) 5 min
Read/Write 0 bytes / 0 bytes 5 min
Out 0 (0 bytes) 5 min
Tasks/Time 0 / 00:00:00.000 5 min
Name output stream
Queued 8 (0 bytes)
ExecuteStreamCommand
ExecuteStreamCommand 1.15.2.2.1.3.0-125
org.apache.nifi - nifi-standard-nar
In 0 (0 bytes) 5 min
Read/Write 0 bytes / 0 bytes 5 min
Out 0 (0 bytes) 5 min
Tasks/Time 0 / 00:00:00.000 5 min
Name output stream
Queued 0 (0 bytes)
UpdateAttribute
UpdateAttribute 1.15.2.2.1.3.0-125
org.apache.nifi - nifi-update-attribute-nar
In 0 (0 bytes) 5 min
Read/Write 0 bytes / 0 bytes 5 min
Out 0 (0 bytes) 5 min
Tasks/Time 0 / 00:00:00.000 5 min
Name success
Queued 0 (0 bytes)
ReplaceText
ReplaceText 1.15.2.2.1.3.0-125
org.apache.nifi - nifi-standard-nar
In 0 (0 bytes) 5 min
Read/Write 0 bytes / 0 bytes 5 min
Out 0 (0 bytes) 5 min
Tasks/Time 0 / 00:00:00.000 5 min
Name success
Queued 0 (0 bytes)
PutParquet
PutParquet 1.15.2.2.1.3.0-125
org.apache.nifi - nifi-parquet-nar
In 0 (0 bytes) 5 min
Read/Write 0 bytes / 0 bytes 5 min
Out 0 (0 bytes) 5 min
Tasks/Time 4,443 / 00:00:00.623 5 min
Name success
Queued 0 (0 bytes)
ExecuteStreamCommand
ExecuteStreamCommand 1.15.2.2.1.3.0-125
org.apache.nifi - nifi-standard-nar
In 0 (0 bytes) 5 min
Read/Write 0 bytes / 0 bytes 5 min
Out 0 (0 bytes) 5 min
Tasks/Time 0 / 00:00:00.000 5 min
Name output stream
Queued 605 (138.84 KB)
DebugFlow
DebugFlow 1.15.2.2.1.3.0-125
org.apache.nifi - nifi-standard-nar
In 0 (0 bytes) 5 min
Read/Write 0 bytes / 0 bytes 5 min
Out 0 (0 bytes) 5 min
Tasks/Time 0 / 00:00:00.000 5 min

a. **Generate Flow File Processor** - Start by placing the Generate Flow File Processor in the NiFi canvas. The processor properties are kept to **Default**.

It is scheduled to run every 4000 secs. For example:

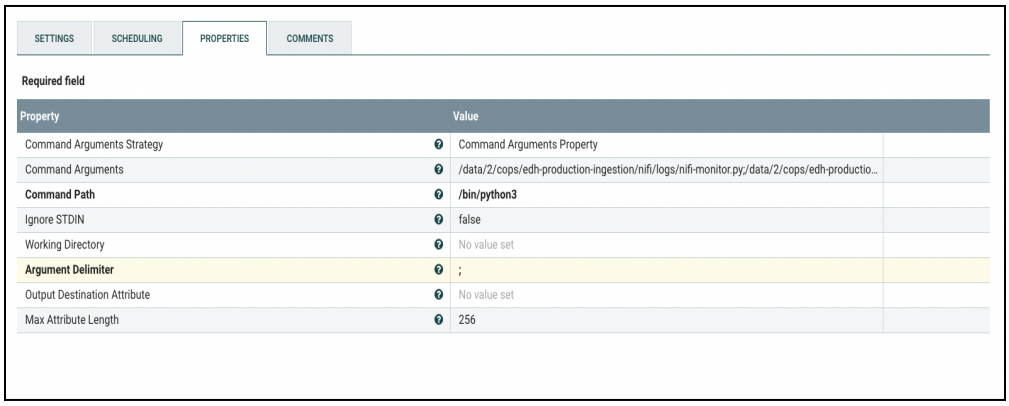

b. **ExecuteStreamCommand** - Place the ExecuteStreamCommand Processor next in the series, used to provide execute permissions to the NiFi input log file.

The processor properties section looks like below:

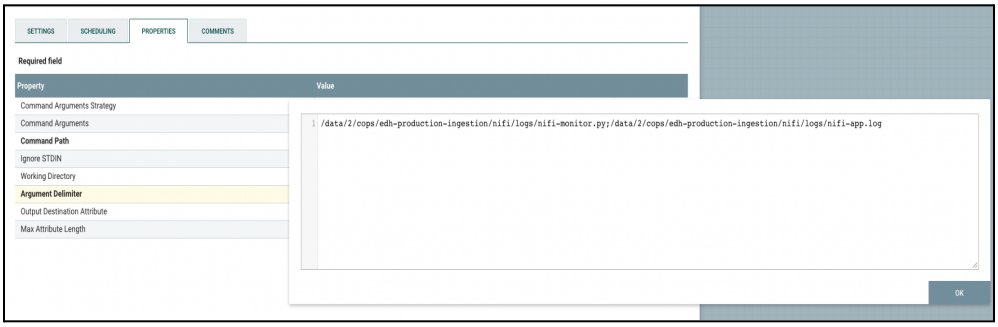

c. **ExecuteStreamCommand** - The second ExecuteStreamCommand Processor executes the python script and input the NiFi log file as an argument.

Link to python script : NiFi Monitoring Pythin Script

The processor properties looks like below:

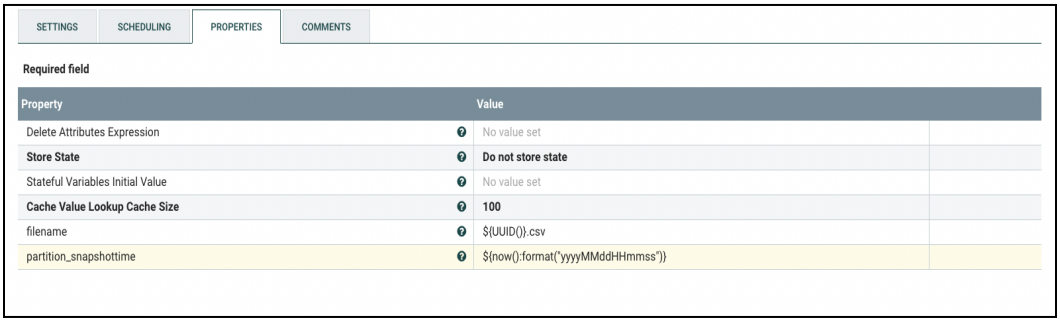



Command Arguments value - Pass the nifi-app.log file as an argument after python script.



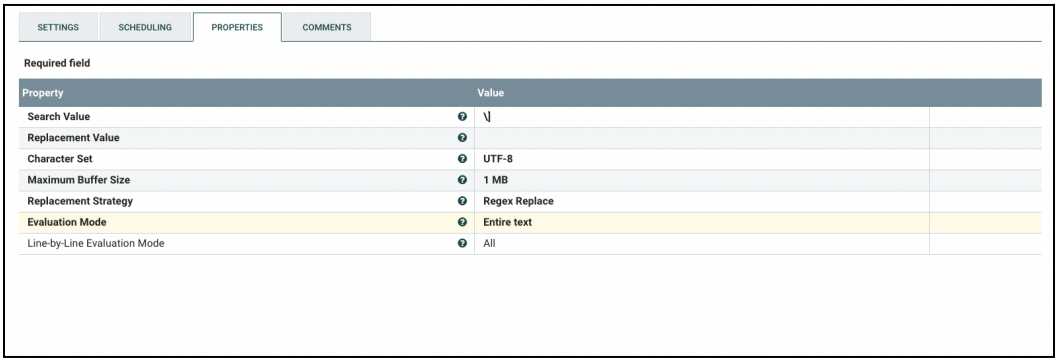

The Run Schedule of this processor is set to 4000 sec. For example:

d. **UpdateAttribute** - Defines flowfile name and partition snapshot time variables. The processor properties looks like below:



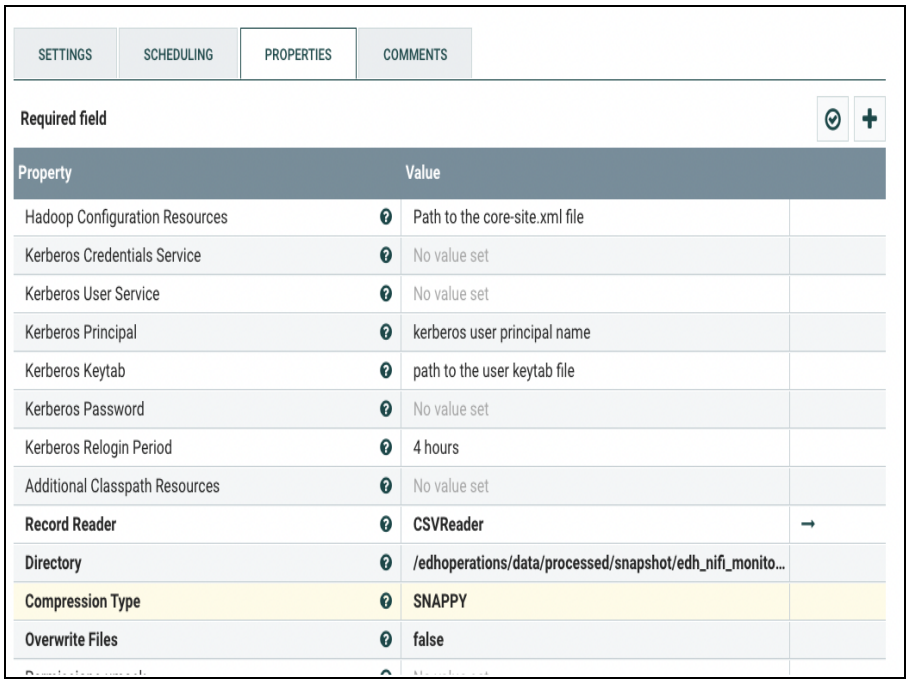

e. **Replace Text Processor** - This processor searches for "/]" and removes the unwanted keyword from the updated file.

The processor properties looks like below:




f. **PutParquet** - The PutParquet Processor loads the failed processor data into HDFS in CSV format. The CSV file data is imported into a different directory for every new partition_snapshot.

The processor properties looks like below:



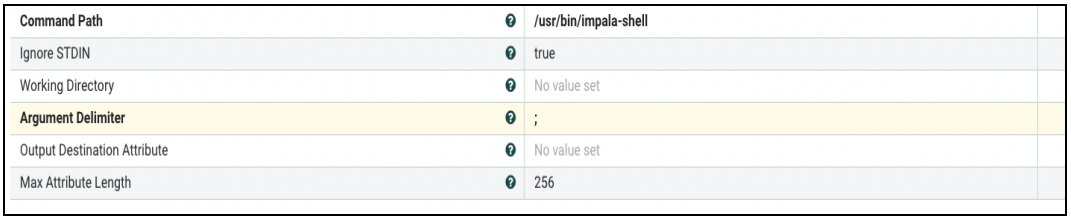

Directory Value:


g. **ExecuteStreamCommand** - This ExecuteStreamCommand alters the hive external table to add partition "partition_snapshottime" and loads the data from HDFS directory location passed in the previous step.

The processor properties looks like below:

Command Argument Value :

```
-k;--ssl;-i;(impalagatewayhostname);-q;"alter table edhoperations.edh_nifi_monitor
add partition (snapshottime=${partition_snapshottime}) location
'/edhoperations/data/processed/snapshot/edh_nifi_monitor/snapshot=${partition_s
napshottime}/'"
```



| | |
|---|---|
| **Command Path** | **/usr/bin/impala-shell** |
| Ignore STDIN | true |
| Working Directory | No value set |
| **Argument Delimiter** | ; |
| Output Destination Attribute | No value set |
| Max Attribute Length | 256 |

- Connect the table to a visualization tool

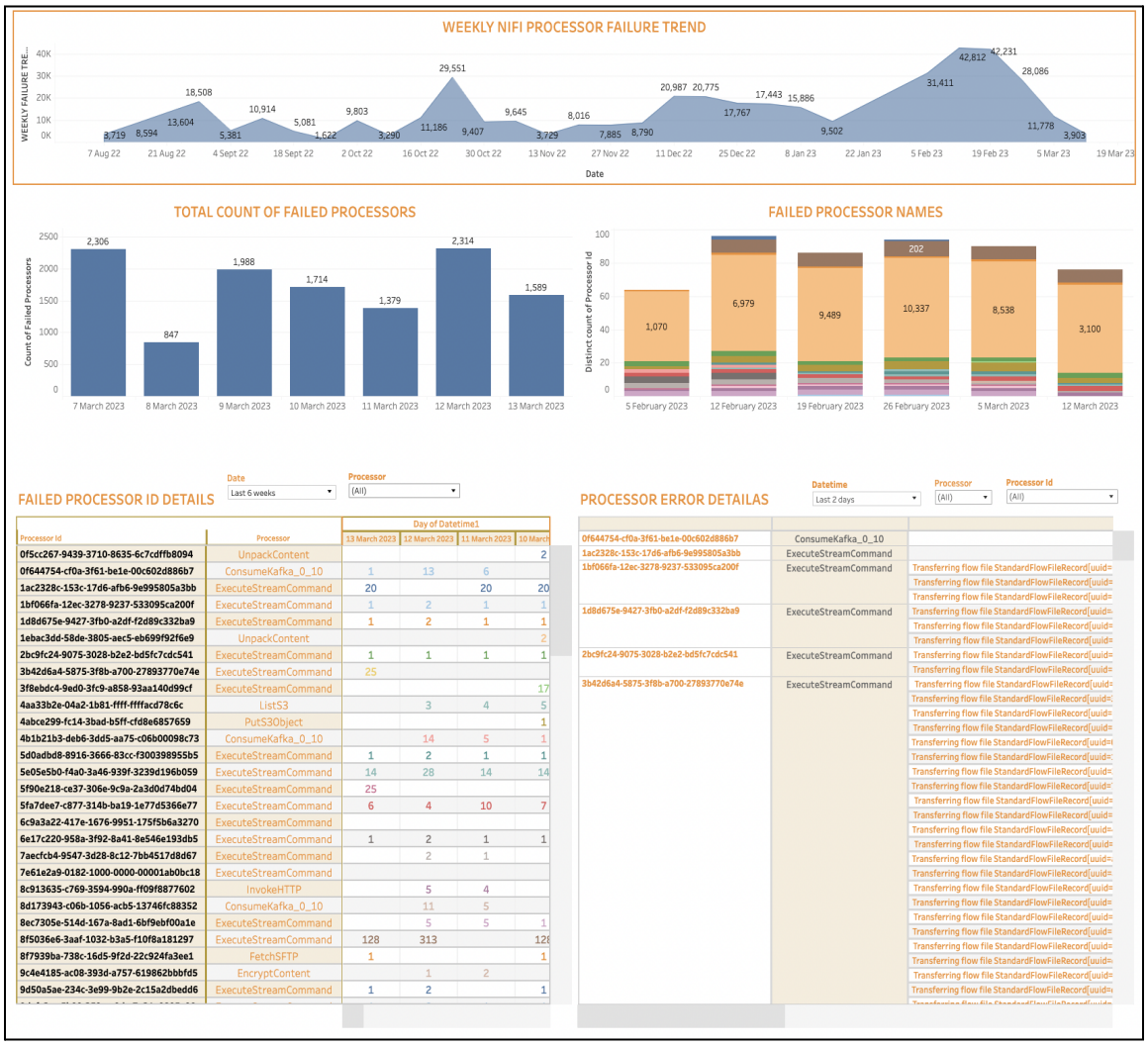
  How it works: To explore and represent the data, you can make use of any visualization tool that has connectivity to Hive or Impala through a JDBC/ODBC connection. We'll demonstrate using Tableau.

  Steps to Perform:

  1. Follow the article link Cloudera Hadoop Tableau Connection to connect Tableau to a Cloudera Data Platform Hive Database.

- Build the dashboard

  How it works: The basic structure of your visualization should look like below. As seen from the example, different views are brought together to build the dashboard.



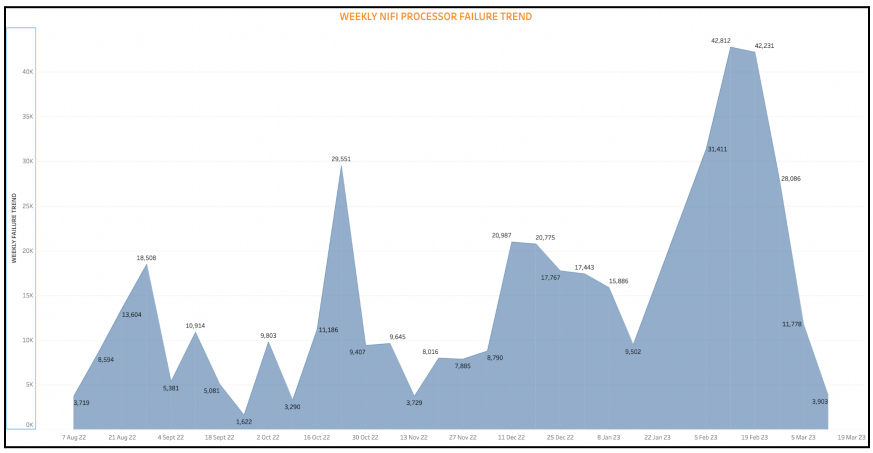

Before you begin: Make sure that you have connected to your table data source.

Steps to Perform:

1. Building the Worksheets
    a. **Weekly NiFi Processor Failure Trend :** This view displays a weekly trend showing processor failure rate over an area chart.
        1. Navigate to a New Worksheet and name it "Weekly NiFi Processor Failure Trend".
        2. In the **Columns** shelf, drop **Datetime1** and select **Week** in the format Week 5, 2015.
        3. In the **Rows** shelf, drop **Processor Id**, right-click and select **Measure -> Count.**

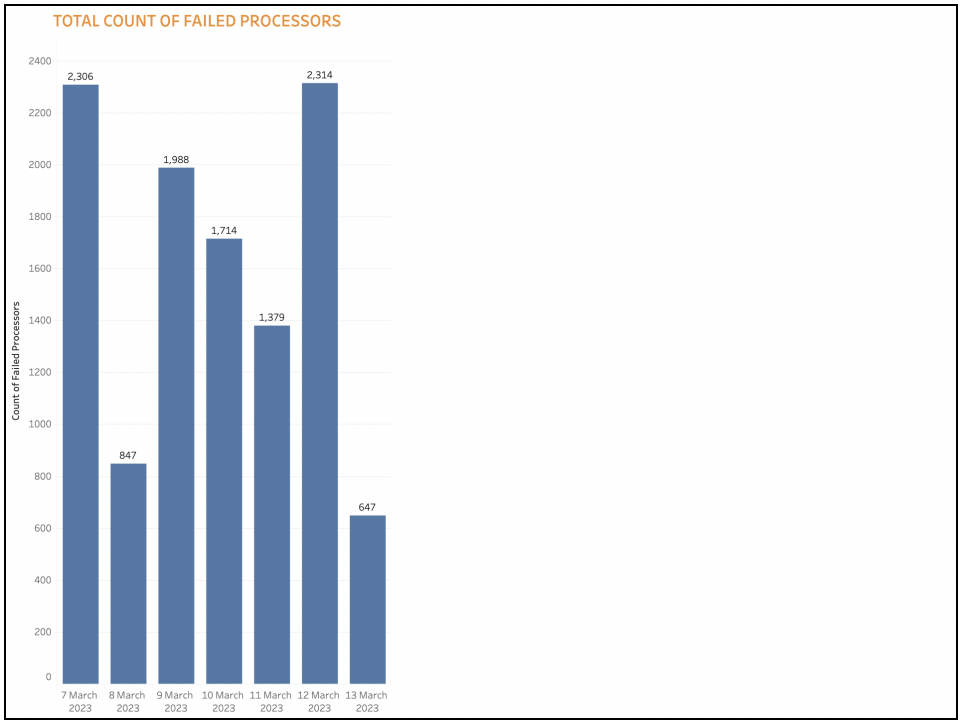
4. Select **Area** representation from the **Marks** section
5. Drop **Processor Id** in the **Label** icon of the **Marks** section, right-click and select **Measure -> Count.**

   The visualization updates to the following:



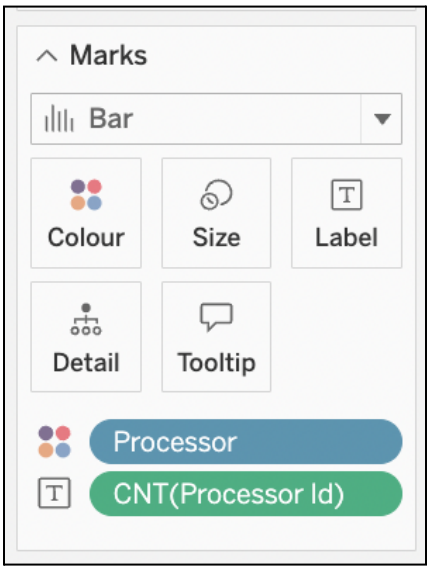

b. **Total Count Of Failed Processors:** This view shows the NiFi Cluster's total daily failure rate for NiFi Processors.
   1. Navigate to a New Worksheet and name it "Total Count Of Failed Processors".
   2. In the **Columns** shelf, drop **Datetime1** and select **Day** in the format 8th May, 2015.
   3. In the **Rows** shelf, drop **Processor Id**, right-click and select **Measure -> Count.**
   4. Select **Bar Graph** representation from the **Marks** section
   5. Drop **Processor Id** in the **Label** icon of the **Marks** section, right-click and select **Measure -> Count.**
   6. In the **Filters** section, drop the Datetime1 field, Select **Edit Filter** -> Select **Relative Dates** -> select **Days** from the drop down and enter **7** in the Last days tab.

      The visualization updates to the following:



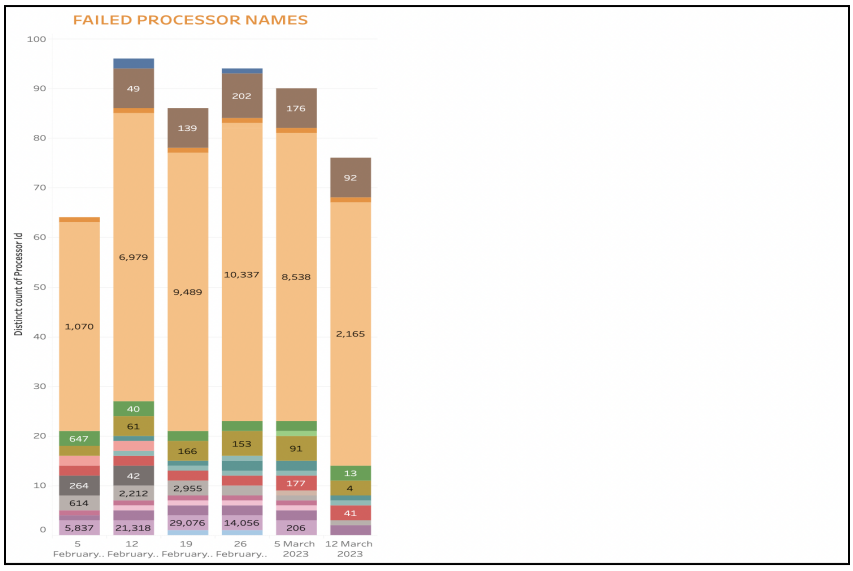

c. **Failed Processor Names :** This view shows a week-long time graph with the names of the failed processors and the number of times each one failed.

1. Navigate to a New Worksheet and name it "Failed Processor Names"
2. In the **Columns** shelf, drop **Datetime1** and select **Day** in the format March 5th, 2015
3. In the **Rows** shelf, drop **Processor Id**, right-click and select **Measure -> Count (Distinct).**
4. Select **Bar Graph** representation from the **Marks** section
5. In the **Marks** section, drop **Processor Id** and **Processor** as shown in the below example:



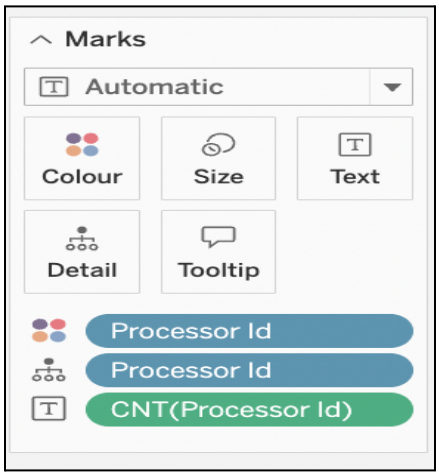

6. In the **Filters** section, drop **Processor** and select **Show Filter .** Additionally, drop the Datetime1 field, Select **Edit Filter** -> Select **Relative Dates** -> select **Days** from the drop down and enter **7** in the Last days tab.

   The visualization updates to the following:

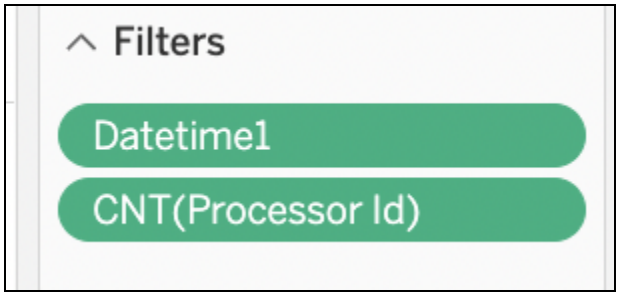



d. **Failed Processor ID Details (Daily) :** This view displays the processor id's of the failed processors and the failure rate for each.
   1. Navigate to a New Worksheet and name it "Failed Processor ID Details (Daily)"

2. In the **Columns** shelf, drop **Datetime1** and select **Day** in the format March 5th, 2015
3. In the **Rows** shelf, drop Processor ID and Processor in series.
4. In the **Marks** section, drop Processor Id as shown in the below example:

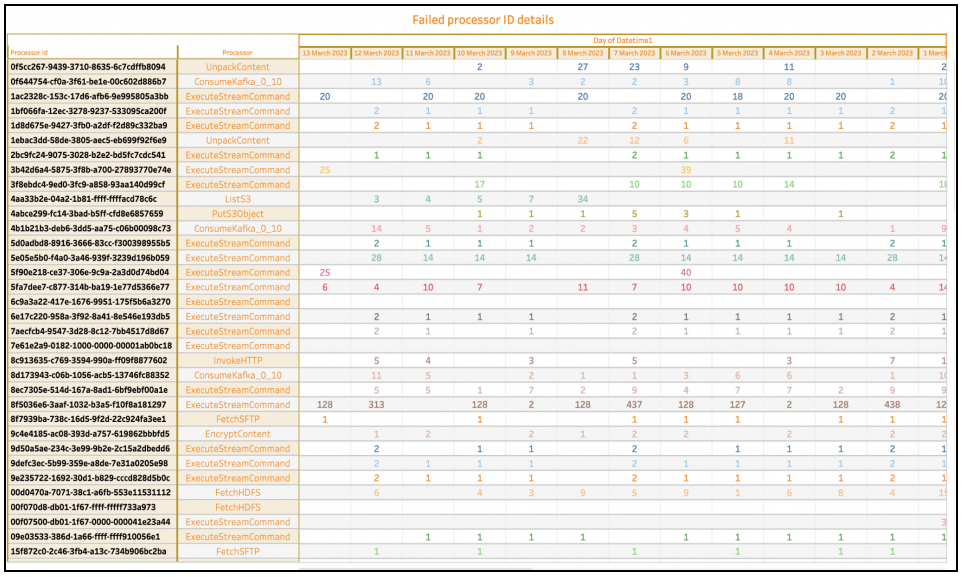



5. In the **Filters** section, drop the **Datetime1** and **CNT(Processor Id)** and **Processor** fields as shown in the below example:




For **Datetime1**, Select **Edit Filter** -> Select **Relative Dates** -> select **Days** from the drop down and enter **7** in the Last days tab. For **Processor** , Select **Show Filter**

The visualization updates to the following:



Failed processor ID details

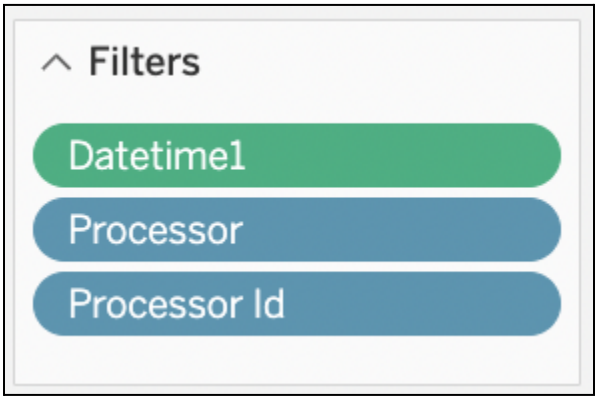
| Processor Id | Processor | Day of Datetime1 | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 13 March 2023 | 12 March 2023 | 11 March 2023 | 10 March 2023 | 9 March 2023 | 8 March 2023 | 7 March 2023 | 6 March 2023 | 5 March 2023 | 4 March 2023 | 3 March 2023 | 2 March 2023 | 1 March |
| 0f5cc267-9439-3710-8635-6c7cdffb8094 | UnpackContent | | | | 2 | | 27 | 23 | 9 | | 11 | | | 2 |
| 0f644754-cf0a-3f61-be1e-00c602d886b7 | ConsumeKafka_0_10 | | 13 | 6 | | 3 | 2 | 2 | 3 | 8 | 8 | | 1 | 1 |
| 1ac2328c-153c-17d6-afb6-9e995805a3bb | ExecuteStreamCommand | 20 | | 20 | 20 | | 20 | | 20 | 18 | 20 | 20 | | 2 |
| 1bf066fa-12ec-3278-9237-533095ca200f | ExecuteStreamCommand | | 2 | 1 | 1 | 1 | | 2 | 1 | 1 | 1 | 1 | 2 | 1 |
| 1d8d675e-9427-3fb0-a2df-f2d89c332ba9 | ExecuteStreamCommand | | 2 | 1 | 1 | 1 | | 2 | 1 | 1 | 1 | 1 | 2 | 1 |
| 1ebac3dd-58de-3805-aec5-eb699f92f6e9 | UnpackContent | | | | 2 | | 22 | 12 | 6 | | 11 | | | |
| 2bc9fc24-9075-3028-b2e2-bd5fc7cdc541 | ExecuteStreamCommand | | 1 | 1 | 1 | | | 2 | 1 | 1 | 1 | 1 | 2 | 1 |
| 3b42d6a4-5875-3f8b-a700-27893770e74e | ExecuteStreamCommand | 25 | | | | | | | 39 | | | | | |
| 3f8ebdc4-9ed0-3fc9-a858-93aa140d99cf | ExecuteStreamCommand | | | | 17 | | | 10 | 10 | 10 | 14 | | | 1 |
| 4aa33b2e-04a2-1b81-ffff-ffffacd78c6c | ListS3 | | 3 | 4 | 5 | 7 | 34 | | | | | | | |
| 4abce299-fc14-3bad-b5ff-cfd8e6857659 | PutS3Object | | | | 1 | 1 | 1 | 5 | 3 | 1 | | 1 | | |
| 4b1b21b3-deb6-3dd5-aa75-c06b00098c73 | ConsumeKafka_0_10 | | 14 | 5 | 1 | 2 | 2 | 3 | 4 | 5 | 4 | | 1 | 9 |
| 5d0adbd8-8916-3666-83cc-f30039895 5b5 | ExecuteStreamCommand | | 2 | 1 | 1 | 1 | | 2 | 1 | 1 | 1 | | 2 | 1 |
| 5e05e5b0-f4a0-3a46-939f-3239d196b059 | ExecuteStreamCommand | | 28 | 14 | 14 | 14 | | 28 | 14 | 14 | 14 | 14 | 28 | 1 |
| 5f90e218-ce37-306e-9c9a-2a3d0d74bd04 | ExecuteStreamCommand | 25 | | | | | | | 40 | | | | | |
| 5fa7dee7-c877-314b-ba19-1e77d5366e77 | ExecuteStreamCommand | 6 | 4 | 10 | 7 | | 11 | 7 | 10 | 10 | 10 | 10 | 4 | 1 |
| 6c9a3a22-417e-1676-9951-175f5b6a3270 | ExecuteStreamCommand | | | | | | | | | | | | | |
| 6e17c220-958a-3f92-8a41-8e546e193db5 | ExecuteStreamCommand | | 2 | 1 | 1 | 1 | | 2 | 1 | 1 | 1 | 1 | 2 | 1 |
| 7aecfcb4-9547-3d28-8c12-7bb4517d8d67 | ExecuteStreamCommand | | 2 | 1 | | 1 | | 2 | 1 | 1 | 1 | 1 | 2 | 1 |
| 7e61e2a9-0182-1000-0000-00001ab0bc18 | ExecuteStreamCommand | | | | | | | | | | | | | |
| 8c913635-c769-3594-990a-ff09f8877602 | InvokeHTTP | | 5 | 4 | | 3 | | 5 | | | 3 | | 7 | 1 |
| 8d173943-c06b-1056-acb5-13746fc88352 | ConsumeKafka_0_10 | | 11 | 5 | | 2 | 1 | 1 | 3 | 6 | 6 | | 1 | 1 |
| 8ec7305e-514d-167a-8ad1-6bf9ebf00a1e | ExecuteStreamCommand | | 5 | 5 | 1 | 7 | 2 | 9 | 4 | 7 | 7 | 2 | 9 | 9 |
| 8f5036e6-3aaf-1032-b3a5-f10f8a181297 | ExecuteStreamCommand | 128 | 313 | | 128 | 2 | 128 | 437 | 128 | 127 | 2 | 128 | 438 | 12 |
| 8f7939ba-738c-16d5-9f2d-22c924fa3ee1 | FetchSFTP | 1 | | | 1 | | | 1 | 1 | 1 | | 1 | 1 | 1 |
| 9c4e4185-ac08-393d-a757-619862bbbfd5 | EncryptContent | | 1 | 2 | | 2 | 1 | 2 | 2 | | 2 | | 2 | 2 |
| 9d50a5ae-234c-3e99-9b2e-2c15a2dbedd6 | ExecuteStreamCommand | | 2 | | 1 | 1 | | 2 | | 1 | 1 | 1 | 2 | 1 |
| 9defc3ec-5b99-359e-a8de-7e31a0205e98 | ExecuteStreamCommand | | 2 | 1 | 1 | 1 | | 2 | 1 | 1 | 1 | 1 | 2 | 1 |
| 9e235722-1692-30d1-b829-cccd828d5b0c | ExecuteStreamCommand | | 2 | 1 | 1 | 1 | | 2 | 1 | 1 | 1 | 1 | 2 | 1 |
| 00d0470a-7071-38c1-a6fb-553e11531112 | FetchHDFS | | 6 | | 4 | 3 | 9 | 5 | 9 | 1 | 6 | 8 | 4 | 1 |
| 00f070d8-db01-1f67-ffff-fffff733a973 | FetchHDFS | | | | | | | | | | | | | |
| 00f07500-db01-1f67-0000-000041e23a44 | ExecuteStreamCommand | | | | | | | | | | | | | 3 |
| 09e03533-386d-1a66-ffff-ffff910056e1 | ExecuteStreamCommand | | | 1 | 1 | 1 | 1 | | 1 | 1 | 1 | 1 | 1 | 1 |
| 15f872c0-2c46-3fb4-a13c-734b906bc2ba | FetchSFTP | | 1 | | 1 | | | 1 | | 1 | | 1 | 1 | |

e. **Processor ERROR Details** : This view displays the exception details of the failed processor.

1. Navigate to a New Worksheet and name it "Processor ERROR Details"
2. In the **Rows** shelf, drop Processor ID, Processor and ERROR in series.

3. In the **Filters** section, drop the **Datetime1**, **Processor** and **Processor Id** fields as shown in the below example.

   Right click and Select **Show Filter** on each of the fields



The visualization updates to the following:

Failed processor ID details

| Processor Id | Processor | Day of Datetime1 | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 13 March 2023 | 12 March 2023 | 11 March 2023 | 10 March 2023 | 9 March 2023 | 8 March 2023 | 7 March 2023 | 6 March 2023 | 5 March 2023 | 4 March 2023 | 3 March 2023 | 2 March 2023 | 1 Marc |
| 0f5cc267-9439-3710-8635-6c7cdffb8094 | UnpackContent | | | | 2 | | 27 | 23 | 9 | | 11 | | | 2 |
| 0f644754-cf0a-3f61-be1e-00c602d886b7 | ConsumeKafka_0_10 | | 13 | 6 | | 3 | 2 | 2 | 3 | 8 | 8 | | 1 | 1 |
| 1ac2328c-153c-17d6-afb6-9e995805a3bb | ExecuteStreamCommand | 20 | | 20 | 20 | | 20 | | 20 | 18 | 20 | 20 | | 2 |
| 1bf066fa-12ec-3278-9237-533095ca200f | ExecuteStreamCommand | | 2 | 1 | 1 | 1 | | 2 | 1 | 1 | 1 | 1 | 2 | 1 |
| 1d8d675e-9427-3fb0-a2df-f2d89c332ba9 | ExecuteStreamCommand | | 2 | 1 | 1 | 1 | | 2 | 1 | 1 | 1 | 1 | 2 | 1 |
| 1ebac3dd-58de-3805-aec5-eb699f92f6e9 | UnpackContent | | | | 2 | | 22 | 12 | 6 | | 11 | | | |
| 2bc9fc24-9075-3028-b2e2-bd5fc7cdc541 | ExecuteStreamCommand | | 1 | 1 | 1 | | | 2 | 1 | 1 | 1 | 1 | 2 | 1 |
| 3b42d6a4-5875-3f8b-a700-27893770e74e | ExecuteStreamCommand | 25 | | | | | | | 39 | | | | | |
| 3f8ebdc4-9ed0-3fc9-a858-93aa140d99cf | ExecuteStreamCommand | | | | 17 | | | 10 | 10 | 10 | 14 | | | 1 |
| 4aa33b2e-04a2-1b81-ffff-ffffacd78c6c | ListS3 | | 3 | 4 | 5 | 7 | 34 | | | | | | | |
| 4abce299-fc14-3bad-b5ff-cfd8e6857659 | PutS3Object | | | | 1 | 1 | 1 | 5 | 3 | 1 | | 1 | | |
| 4b1b21b3-deb6-3dd5-aa75-c06b00098c73 | ConsumeKafka_0_10 | | 14 | 5 | 1 | 2 | 2 | 3 | 4 | 5 | 4 | | 1 | 9 |
| 5d0adbd8-8916-3666-83cc-f300398955b5 | ExecuteStreamCommand | | 2 | 1 | 1 | 1 | | 2 | 1 | 1 | 1 | | 2 | 1 |
| 5e05e5b0-f4a0-3a46-939f-3239d196b059 | ExecuteStreamCommand | | 28 | 14 | 14 | 14 | | 28 | 14 | 14 | 14 | 14 | 28 | 14 |
| 5f90e218-ce37-306e-9c9a-2a3d0d74bd04 | ExecuteStreamCommand | 25 | | | | | | | 40 | | | | | |
| 5fa7dee7-c877-314b-ba19-1e77d5366e77 | ExecuteStreamCommand | 6 | 4 | 10 | 7 | | 11 | 7 | 10 | 10 | 10 | 10 | 4 | 1 |
| 6c9a3a22-417e-1676-9951-175f5b6a3270 | ExecuteStreamCommand | | | | | | | | | | | | | |
| 6e17c220-958a-3f92-8a41-8e546e193db5 | ExecuteStreamCommand | | 2 | 1 | 1 | 1 | | 2 | 1 | 1 | 1 | 1 | 2 | 1 |
| 7aecfcb4-9547-3d28-8c12-7bb4517d8d67 | ExecuteStreamCommand | | 2 | 1 | | 1 | | 2 | 1 | 1 | 1 | 1 | 2 | 1 |
| 7e61e2a9-0182-1000-0000-00001ab0bc18 | ExecuteStreamCommand | | | | | | | | | | | | | |
| 8c913635-c769-3594-990a-ff09f8877602 | InvokeHTTP | | 5 | 4 | | 3 | | 5 | | | 3 | | 7 | 1 |
| 8d173943-c06b-1056-acb5-13746fc88352 | ConsumeKafka_0_10 | | 11 | 5 | | 2 | 1 | 1 | 3 | 6 | 6 | | 1 | 1 |
| 8ec7305e-514d-167a-8ad1-6bf9ebf00a1e | ExecuteStreamCommand | | 5 | 5 | 1 | 7 | 2 | 9 | 4 | 7 | 7 | 2 | 9 | 9 |
| 8f5036e6-3aaf-1032-b3a5-f10f8a181297 | ExecuteStreamCommand | 128 | 313 | | 128 | 2 | 128 | 437 | 128 | 127 | 2 | 128 | 438 | 12 |
| 8f7939ba-738c-16d5-9f2d-22c924fa3ee1 | FetchSFTP | 1 | | | 1 | | | 1 | 1 | 1 | | 1 | 1 | 1 |
| 9c4e4185-ac08-393d-a757-619862bbbfd5 | EncryptContent | | 1 | 2 | | 2 | 1 | 2 | 2 | | 2 | | 2 | 2 |
| 9d50a5ae-234c-3e99-9b2e-2c15a2dbedd6 | ExecuteStreamCommand | | 2 | | 1 | 1 | | 2 | | 1 | 1 | 1 | 2 | 1 |
| 9defc3ec-5b99-359e-a8de-7e31a0205e98 | ExecuteStreamCommand | | 2 | 1 | 1 | 1 | | 2 | 1 | 1 | 1 | 1 | 2 | 1 |
| 9e235722-1692-30d1-b829-cccd828d5b0c | ExecuteStreamCommand | | 2 | 1 | 1 | 1 | | 2 | 1 | 1 | 1 | 1 | 2 | 1 |
| 00d0470a-7071-38c1-a6fb-553e11531112 | FetchHDFS | | 6 | | 4 | 3 | 9 | 5 | 9 | 1 | 6 | 8 | 4 | 1 |
| 00f070d8-db01-1f67-ffff-ffff1733a973 | FetchHDFS | | | | | | | | | | | | | |
| 00f07500-db01-1f67-0000-000041e23a44 | ExecuteStreamCommand | | | | | | | | | | | | | 3 |
| 09e03533-386d-1a66-ffff-ffff910056e1 | ExecuteStreamCommand | | | 1 | 1 | 1 | 1 | | 1 | 1 | 1 | 1 | 1 | 1 |
| 15f872c0-2c46-3fb4-a13c-734b906bc2ba | FetchSFTP | | 1 | | 1 | | | 1 | | 1 | | 1 | 1 | |

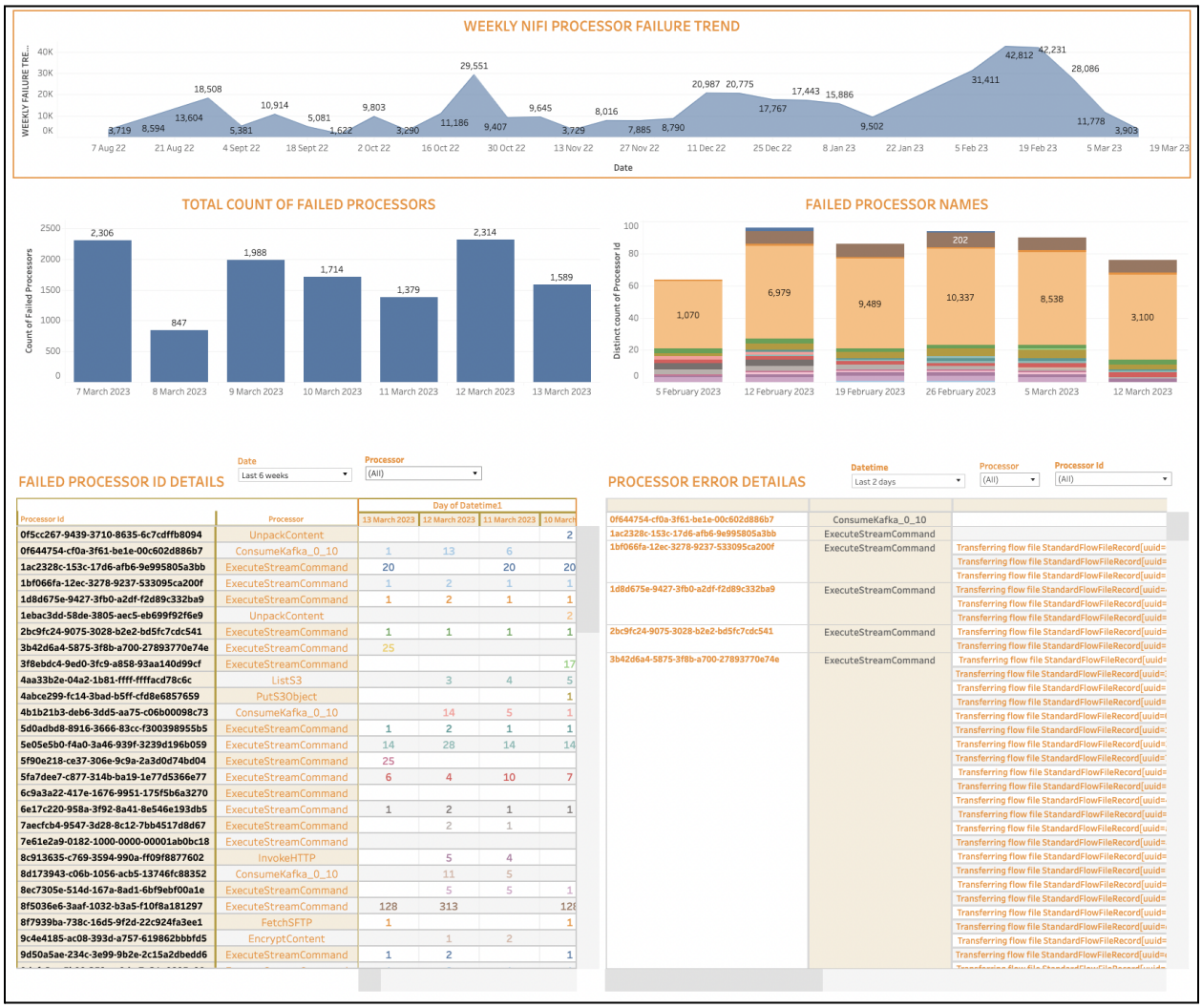
2. Gathering the views to build the dashboard
    a. At the bottom of the workbook, click the New Dashboard icon:


    b. You can use horizontal and vertical objects to provide a visual appeal to the dashboard and group different worksheet views together.
    c. Make use of filters on your views where necessary.

Note : Design your dashboard as per your visualization preferences.

For more details and best practices on building a dashboard in Tableau refer link : Create a Dashboard

Your dashboard visualization updates as below if the views are sequenced properly:




- Implement end-user alerting mechanism

  How it works: The deployed airflow dag scheduled to run every 23 hours scans across the hive table to collect the day-1 failed processor data, and then sends an email alert with the failure details to the data-engineering team.

  Steps to perform: On the Airflow master host, deploy the below dag code in the dags directory and schedule it to run every 23 hours

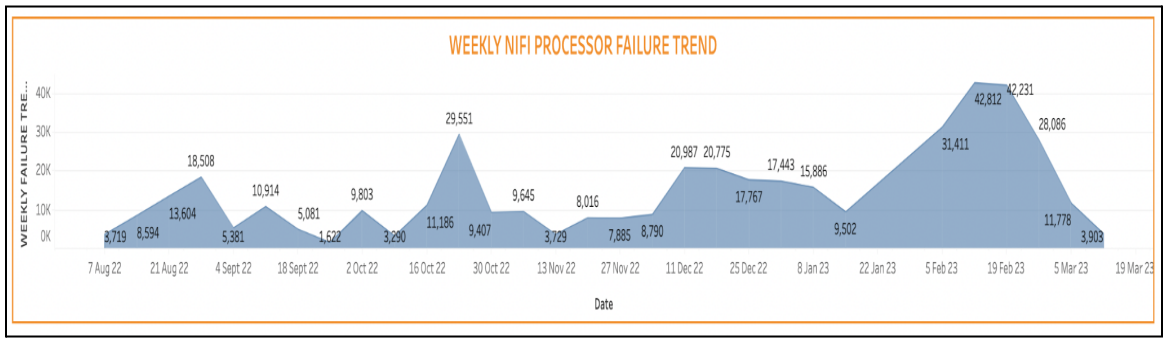
Airflow Dag Code: [Provided at the end of the article]

Note: The code provided is merely for reference purposes. Customize the airflow dag per your environment requirements.

## Actionable Insights

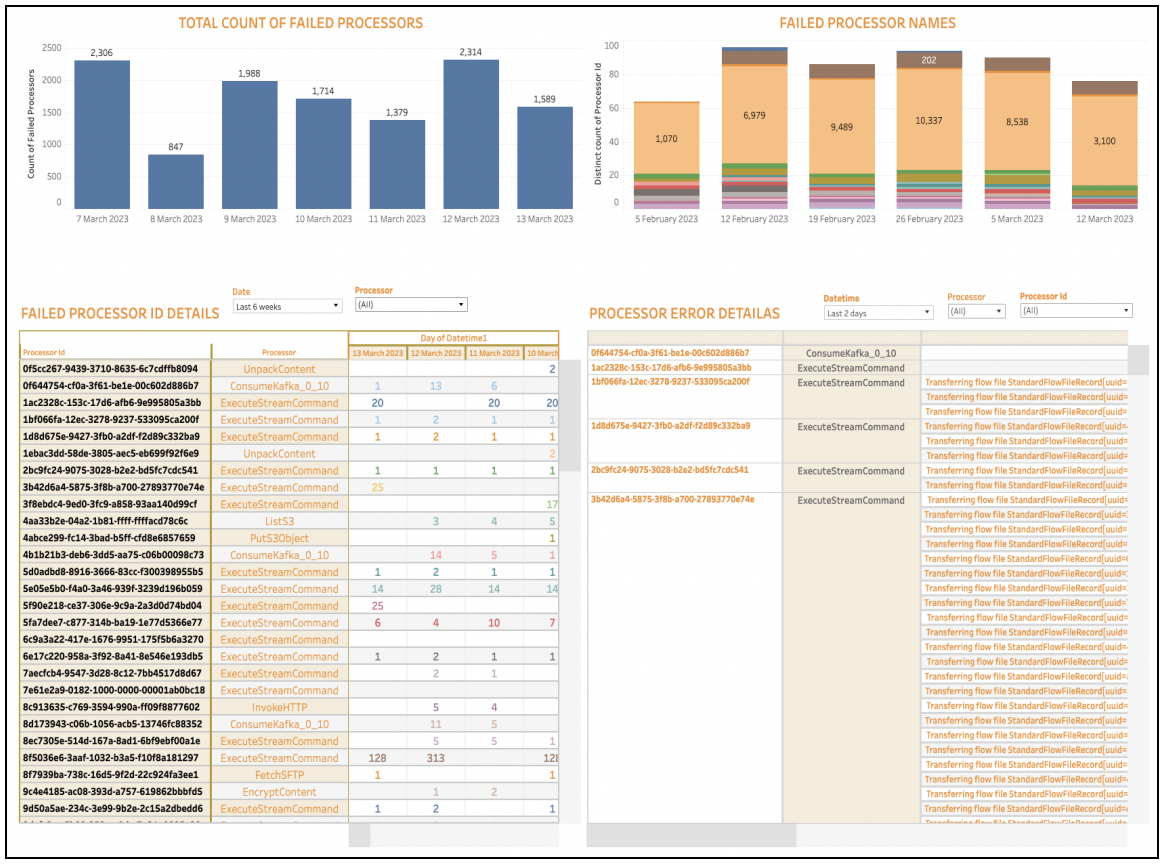
A good rule of thumb is to keep a view of data showing no less than 6 months to give you a comprehensive view.

1. Tracking progress overtime :




- Pay appropriate attention to any increase in the number of failures using the area chart above, and identify which processors spiked those problems. Use the views in the next section to get processor details for further troubleshooting.
- Observe and compare the present and past numbers to visualize and understand if there's an overall increase or reduction in the trend of failure rate. If your visual observes a decrease in the failure trend it's a good indication that your Nifi failures are decreasing; however, if they are increasing, it may be an indication that your operations team should identify and action root causes (see next section).

2. Your daily view for identifying problems, spotting issue trends, and taking action for long term preventive measures.




- "Total Count Of Failed Processors" and "Failed Processor Names"- Keep a close eye on the total number of daily processor failures and identify the names of failed processors using the "Failed Processor Names" view. Assess the processors with the highest failure count.
- "Failed Processor ID Details" and "Processor Error Details" : Use both these views to get the processor id and error exception details of the failed processor to further troubleshoot and accelerate the root cause using the NiFi cluster UI.

## Recommended Operational Processes

- Scheduled daily monitoring calls to analyze NiFi processor failures and troubleshoot with the help of the dashboard views.
- KPIs to closely monitor daily :
    1. Increase or sudden spikes in the total processor failure rate.
    2. Processors and Processor Id's resulting in the highest number of failures.
    3. Should any of the failures prompt either NiFi pipeline tuning or service tuning?

- Periodically assess the progress of the failure rate utilizing the historic data trend view to reduce failure rate.

## Airflow Dag Code

### #Importing the Modules

```
from datetime import datetime, timedelta
import json
from airflow import DAG
from airflow.operators.python import PythonOperator
from airflow.providers.jdbc.operators.jdbc import JdbcOperator
from airflow.models.variable import Variable
from airflow.operators.email_operator import EmailOperator
from airflow.providers.jdbc.hooks.jdbc import JdbcHook
import pandas as pd
from email.mime.text import MIMEText
from email.mime.application import MIMEApplication
from email.mime.multipart import MIMEMultipart
import email
from email import encoders
from email.mime.base import MIMEBase
from smtplib import SMTP
import smtplib
import sys
```

### #Defining Default Arguments for the Dag

```
default_args = {
    'owner': 'airflow',
    'depends_on_past': False,
    'email': ['job owneremail address'],
    'email_on_failure': True,
    'email_on_retry': False,
    'retries': 1,
    'retry_delay': timedelta(minutes=2)
}

doc = '''
## NIFI monitoring dag to send an email to the Data engineering team informing about the
NIFI processor failures that occurred in the last 24 hours.
```

```
'''
#Initiating the Dag
JOB_ID = 'EDH_nifi_processor_monitoring'

dag = DAG(
    dag_id=JOB_ID,
    doc_md=doc,
    default_args=default_args,
    description='EDH_nifi_jobs_monitoring',
    schedule_interval="0 */23 * * *",
    start_date=datetime(2021, 1, 1),
    tags=['EDH', 'Monitoring', 'failure jobs'],
    max_active_runs=1,
    catchup=False
)

#Defining a callable function


def func(jdbc_conn_id, sql, **kwargs):
    """Print df from JDBC """
    print(kwargs)
    hook = JdbcHook(jdbc_conn_id=jdbc_conn_id)
    df = hook.get_pandas_df(sql=sql,parameters=None)
    print("printing the jobs details")
    print(df.to_string())
    recipients = ['recipient email address']
    msg = MIMEMultipart()
    msg['Subject'] = "NiFi Processor Failures For DAY-1"
    msg['From'] = 'sender email address'


    html = """\
    <html>
      <head></head>
      <body>
    <pre>
    Hello Team,

          Below is the list of NIFI Processors that failed in the last 24 hours. Please check on
         priority level.
```

```
    NIFI UI - Pass the link to the NiFi UI for further troubleshooting
 For more details on ERROR/Exception, please check the dashboard -  <Link to the
visualization>
    </pre>

       {0}

      <pre>

      Please reach out to ops team in case of any queries. Thanks
      Regards
      OPS TEAM

      </pre>

      </body>
    </html>
    """.format(df.to_html())

    part1 = MIMEText(html, 'html')
    msg.attach(part1)

    server = smtplib.SMTP('SMTP Server hostname', 25)
    server.sendmail(msg['From'], 'recipient email address' , msg.as_string())
```

## #Creating a Task

```
run_this = PythonOperator(
    task_id='Job_owner_details',
    python_callable=func,
    op_kwargs={'jdbc_conn_id': 'dcoe_impala', 'sql': 'select
datetime1,loglevel,processor,processor_id, count(*) as count from
edhoperations.edh_nifi_monitor where snapshottime IN (select distinct snapshottime from
edhoperations.edh_nifi_monitor order by snapshottime desc limit 22) group by
processor,processor_id , datetime1, loglevel HAVING COUNT(*) > 0;' },
    dag=dag,
)
```