CLOUDERA

Transparency and Visibility
Airflow Monitoring Implementation Steps

Implementation

Implementing Airflow monitoring includes these tasks:

1. Create External Hive Table
2. Deploy a Python Script
3. Connect the table to a visualization tool
4. Build a Dashboard
5. Implementing end user alerting mechanism

- Create Hive Table

How it works: Create an external Hive table, where the job details data is stored on the HDFS storage layer.

Steps to perform: Hive table schema:

```
CREATE EXTERNAL TABLE edhoperations.airflow_jobs_details ( dag_id STRING, status STRING, start_date TIMESTAMP, end_date TIMESTAMP, run_type STRING ) ROW FORMAT DELIMITED FIELDS TERMINATED BY '|' WITH SERDEPROPERTIES ('field.delim'='|', 'serialization.format'='|') STORED AS TEXTFILE LOCATION 'hdfs://nameservice1/edhoperations/airflow' TBLPROPERTIES ('bucketing_version'='2', 'last_modified_by'='pbhadange', 'last_modified_time'='1664803224', 'numFiles'='95', 'numFilesErasureCoded'='0', 'totalSize'='1921094')
```

- Deploying a Python Script

How it works: The python script makes the connection to the backend airflow database, retrieves the job details and loads the data in the HDFS location in CSV format which is accessed by Hive table for querying.

Python Script is provided at the end of this article

Steps to Perform:

1. Place the python script on the host that is used to deploy Airflow Master Server,
2. Schedule a crontab on the same server where the python script is deployed and set it to run every 3 hours.

```
0 */3  * * * python3 /home/airflow/fetch_airflow_data.py
```

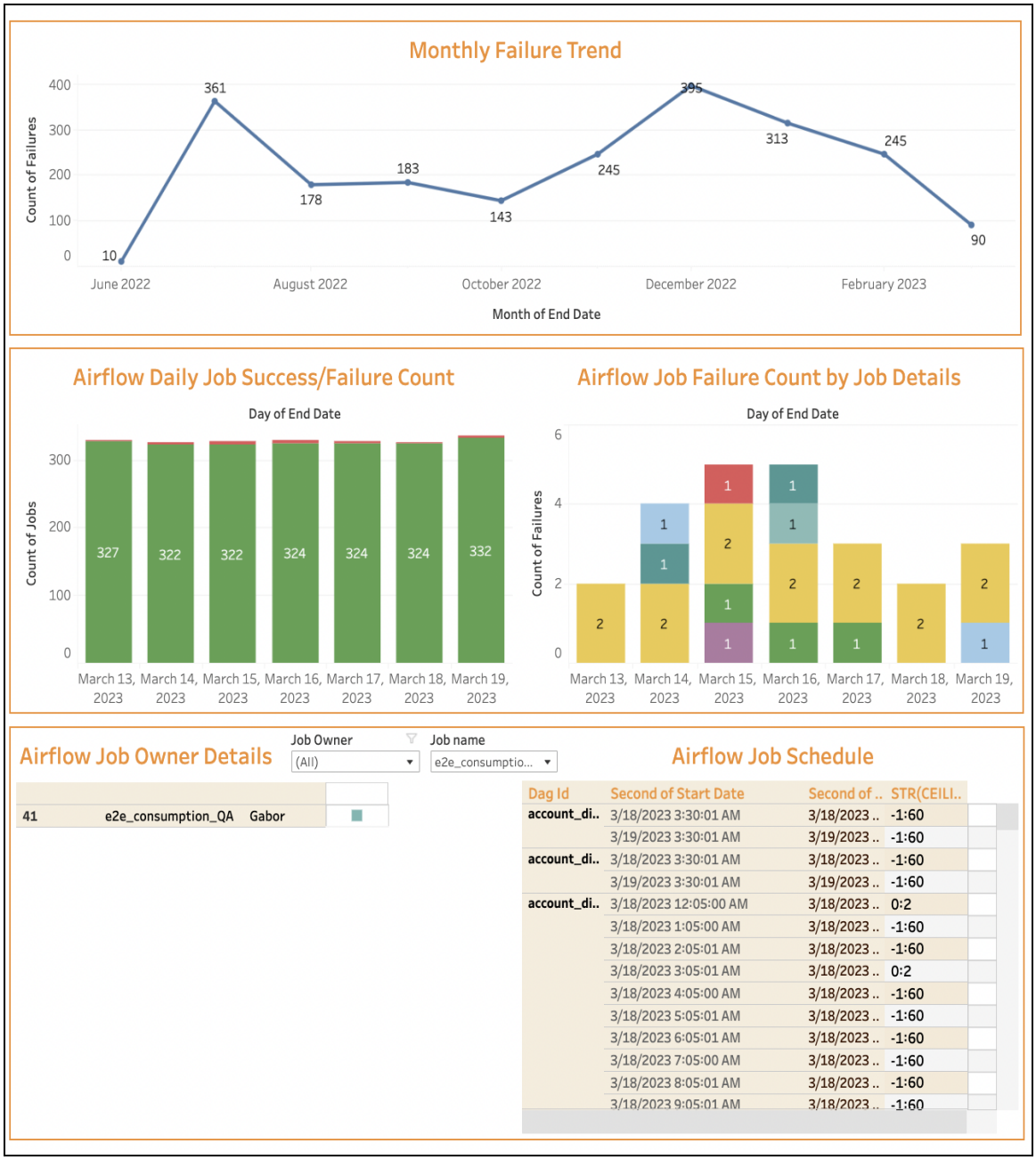
- Connect the table to a Visualization tool

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:
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.


Monthly Failure Trend
Count of Failures
400
300
200
100
0
361
178
183
143
245
395
313
245
90
10
June 2022
August 2022
October 2022
December 2022
February 2023
Month of End Date
Airflow Daily Job Success/Failure Count
Day of End Date
Count of Jobs
300
200
100
0
327
322
322
324
324
324
332
March 13, 2023
March 14, 2023
March 15, 2023
March 16, 2023
March 17, 2023
March 18, 2023
March 19, 2023
Airflow Job Failure Count by Job Details
Day of End Date
Count of Failures
6
4
2
0
Airflow Job Owner Details
Job Owner
(All)
Job name
e2e_consumptio...
41 e2e_consumption_QA Gabor
Airflow Job Schedule
Dag Id
Second of Start Date
Second of ..
STR(CEILI..
account_di..
3/18/2023 3:30:01 AM
3/18/2023 ..
-1:60
3/19/2023 3:30:01 AM
3/19/2023 ..
-1:60
account_di..
3/18/2023 3:30:01 AM
3/18/2023 ..
-1:60
3/19/2023 3:30:01 AM
3/19/2023 ..
-1:60
account_di..
3/18/2023 12:05:00 AM
3/18/2023 ..
0:2
3/18/2023 1:05:00 AM
3/18/2023 ..
-1:60
3/18/2023 2:05:01 AM
3/18/2023 ..
-1:60
3/18/2023 3:05:01 AM
3/18/2023 ..
0:2
3/18/2023 4:05:00 AM
3/18/2023 ..
-1:60
3/18/2023 5:05:01 AM
3/18/2023 ..
-1:60
3/18/2023 6:05:01 AM
3/18/2023 ..
-1:60
3/18/2023 7:05:00 AM
3/18/2023 ..
-1:60
3/18/2023 8:05:01 AM
3/18/2023 ..
-1:60
3/18/2023 9:05:01 AM
3/18/2023 ..
-1:60

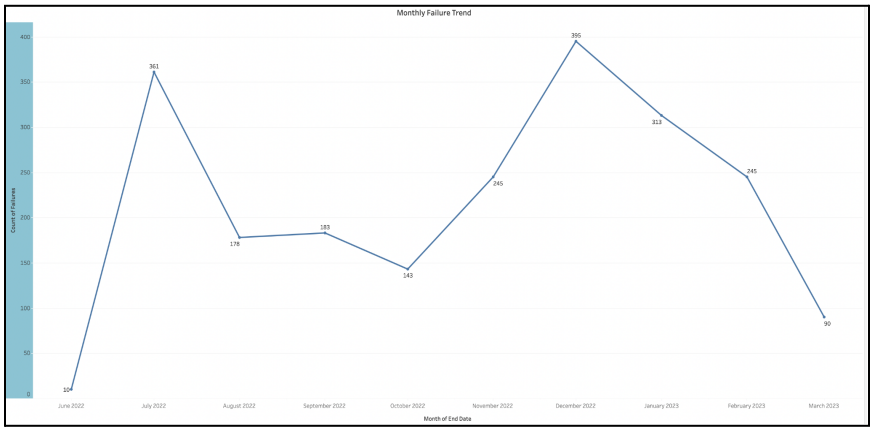
Before you begin: Make sure that you have connected to your table data source.

Steps to Perform:

1. Building the Worksheets

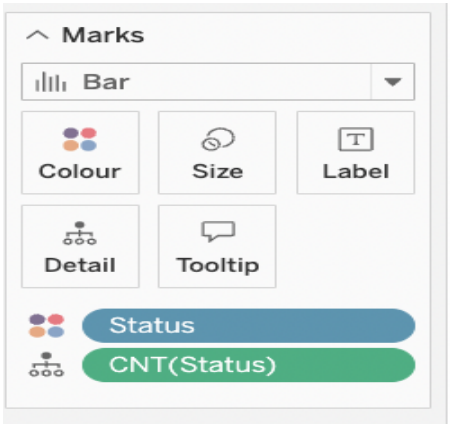
    a. Monthly Failure Trend : The total count of Airflow job failures is shown in this view with its historical trend.
        1. In your new workbook, Navigate to a New Worksheet and name it "Monthly Failure Trend"
        2. From the Data pane, drop **End Date** in the **Column** shelf and **Status** in **Row** Shelf.
        3. On the **Columns** shelf, right-click **End date** and select **Month** in the format May 2015.
        4. In the **Rows** shelf, right-click **Status** and select **Measure -> Count.**
        5. Drop **Status** field in the **Filter** section. Right Click -> **Edit Filter** and select **Failed** from the list of Status.
        6. In the **Marks** section, drop **Status** in the **Detail** Icon. Convert it into **Count** Measure. For example :

        The visualization should look like this:



    b. Airflow Daily Job Success/Failure Count: This view shows the overall count of jobs executed within Airflow for CURRENTDAY-1, broken down by failed and successful job color representation.

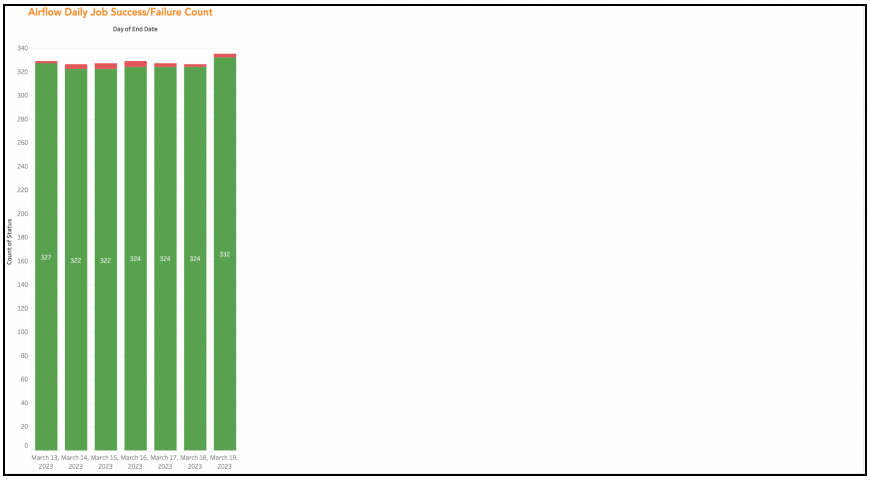
1. In your new workbook, Navigate to a New Worksheet and name it "Airflow Daily Job Success/Failure Count"
2. From the Data pane, drop **End Date** in the **Columns** shelf and select **Day** in the format **8th May, 2015.**
3. Drop **Status** in the **Rows** shelf and select **Measure -> Count**.
4. Select **Bar Chart** representation from the **Marks** section
5. In the **Marks** Section, drop **Status** in the **Color** and **Label** icon (Convert the **Status** field into **Count**) . For example:



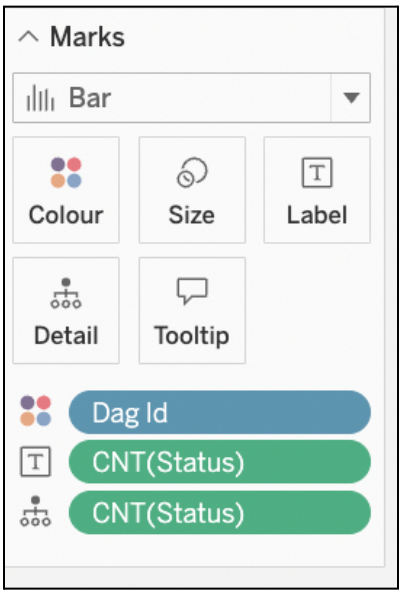
6. Drop **End Date** and **Status** in the **Filter** section.

   In this view, we are filtering to just show the last 7 days of job data. Right click on End Date, **Edit Filter** -> **Relative dates,** select **Days** from the drop down and enter **7**.
   Do the same for **Status** field, Edit **Filter** -> Select **Use All** from the **List**

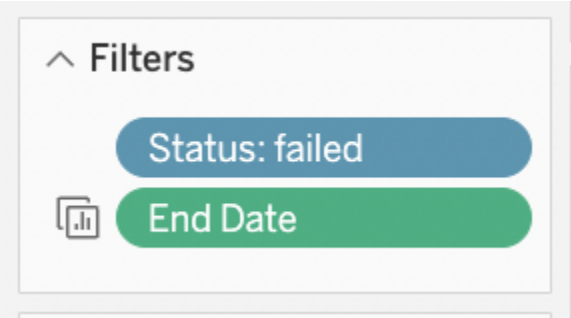
   The visualization should look like this:



c. Airflow Job Failure Count by Job Details: This view displays the count of job failures broken down by dag details.
    1. Navigate to a New Worksheet and name it "Airflow Job Failure Count by Job Name"
    2. From the Data pane, drop **End Date** in the **Columns** shelf and select **Day** in the format **8th May, 2015.**
    3. Drop **Status** in the **Rows** shelf and select **Measure -> Count**.
    4. Select **Bar Chart** representation from the **Marks** section
    5. In the **Marks** section, drop the Dag **Id** field in the **Color** icon and the Status field in the Label and Detail icon. (Convert the **Status** field into **Count**)
    For example:

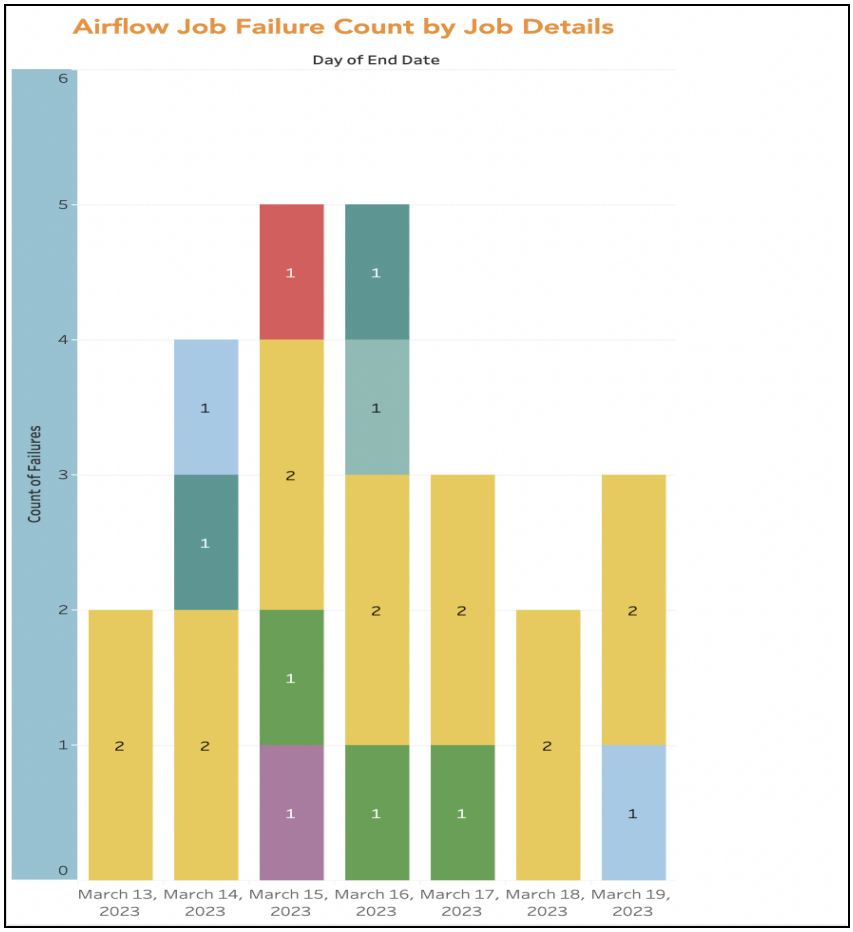


6. In the **Filters** section, drop the fields as shown in the below example:



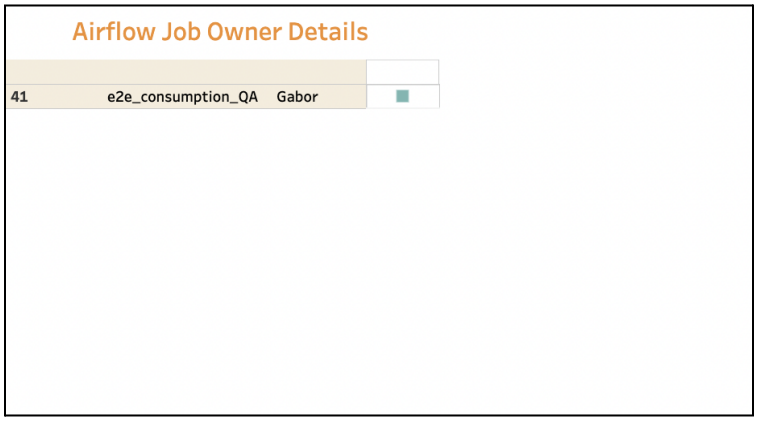
- Status is filtered to only show Failed jobs. Edit **Filter** -> Select **use Failed** from the **List**
- In this view, we are filtering to just show the last 7 days of job data. Right click on Day, **Edit Filter** -> **Relative dates,** select **Days** from the drop down and enter **7**.

The visualization should look like this:

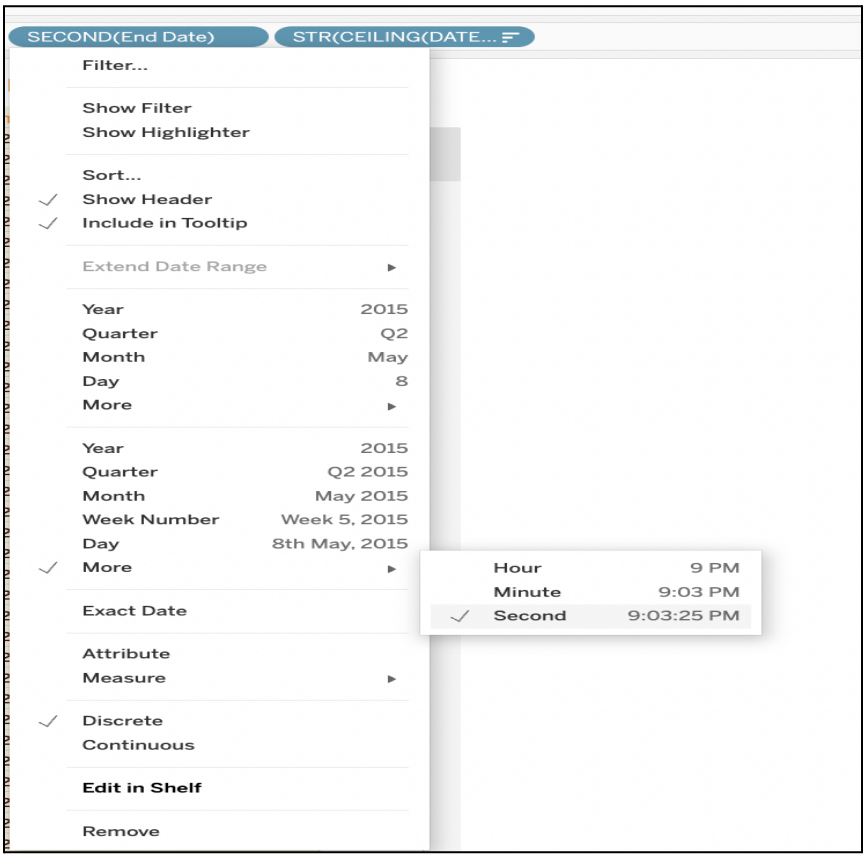


d. Airflow Job Owner Details : This view displays the job owners name of each Dag deployed within airflow.
    1. In your new workbook, Navigate to a New Worksheet and name it "Job Owner Details"
    2. Drop **ID, Job Name, Job Owner** in series in the **Rows** shelf.
    3. In the **Marks** Section, drop **Job Owner** in the **Color** and **Detail** icon.
    4. Drop **Job Name and Job Owner** in the **Filter** section. Right click and select **Show Filter** for both the fields.

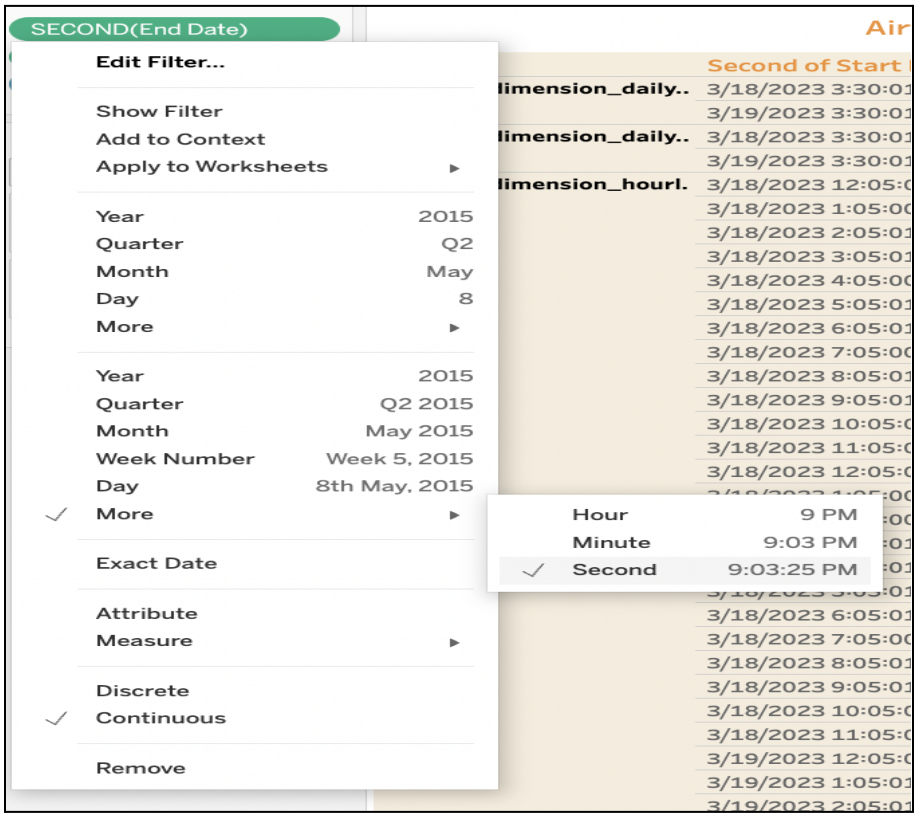
       The visualization should look like this:



e. Airflow Job Schedule: This view displays the run schedule of each Dag with the execution time of each schedule.
    1. In your new workbook, Navigate to a New Worksheet and name it "Airflow Job Schedule"
    2. Drop **ID, Start Date,End Date** in series in the **Rows** shelf. For **Start Date** and **End Date,** select **Measure -> More -> Second** Date Format, as shown in the below example:



3. Drop **Start Date**, **End Date** and **Dag Id** in the **Filter** section.
   - Convert the **Start Date and End Date** in the Seconds Time Format, as shown in the below example:
   - Right click and select **Show Filter** for all the 3 fields.

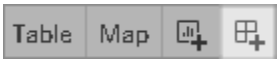


The visualization should look like this:

Airflow job schedule

Dag Id	Second of Start Date	Second of End Date
	3/19/2023 3:30:01 AM	3/19/2023 3:30:42 AM
account_dimension_hourl.	3/18/2023 12:05:00 AM	3/18/2023 12:07:37 AM
.	3/18/2023 1:05:00 AM	3/18/2023 1:05:22 AM
	3/18/2023 2:05:01 AM	3/18/2023 2:05:27 AM
	3/18/2023 3:05:01 AM	3/18/2023 3:07:43 AM
	3/18/2023 4:05:00 AM	3/18/2023 4:05:25 AM
	3/18/2023 5:05:01 AM	3/18/2023 5:05:30 AM
	3/18/2023 6:05:01 AM	3/18/2023 6:05:24 AM
	3/18/2023 7:05:00 AM	3/18/2023 7:05:35 AM
	3/18/2023 8:05:01 AM	3/18/2023 8:05:30 AM
	3/18/2023 9:05:01 AM	3/18/2023 9:05:30 AM
	3/18/2023 10:05:00 AM	3/18/2023 10:05:25 AM
	3/18/2023 11:05:01 AM	3/18/2023 11:05:21 AM
	3/18/2023 12:05:00 PM	3/18/2023 12:05:19 PM
	3/18/2023 1:05:00 PM	3/18/2023 1:05:22 PM
	3/18/2023 2:05:00 PM	3/18/2023 2:05:26 PM
	3/18/2023 3:05:01 PM	3/18/2023 3:05:21 PM
	3/18/2023 4:05:01 PM	3/18/2023 4:05:22 PM
	3/18/2023 5:05:01 PM	3/18/2023 5:05:20 PM
	3/18/2023 6:05:01 PM	3/18/2023 6:05:27 PM
	3/18/2023 7:05:00 PM	3/18/2023 7:05:26 PM
	3/18/2023 8:05:01 PM	3/18/2023 8:05:35 PM
	3/18/2023 9:05:01 PM	3/18/2023 9:05:26 PM
	3/18/2023 10:05:00 PM	3/18/2023 10:05:29 PM
	3/18/2023 11:05:01 PM	3/18/2023 11:05:29 PM
	3/19/2023 12:05:01 AM	3/19/2023 12:05:33 AM
	3/19/2023 1:05:01 AM	3/19/2023 1:05:35 AM
	3/19/2023 2:05:01 AM	3/19/2023 2:05:27 AM
	3/19/2023 3:05:01 AM	3/19/2023 3:05:25 AM
	3/19/2023 4:05:01 AM	3/19/2023 4:05:36 AM
	3/19/2023 5:05:01 AM	3/19/2023 5:05:21 AM
	3/19/2023 6:05:01 AM	3/19/2023 6:05:31 AM
	3/19/2023 7:05:01 AM	3/19/2023 7:05:33 AM
	3/19/2023 8:05:00 AM	3/19/2023 8:05:25 AM
	3/19/2023 9:05:01 AM	3/19/2023 9:05:23 AM
	3/19/2023 10:05:01 AM	3/19/2023 10:05:33 AM

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



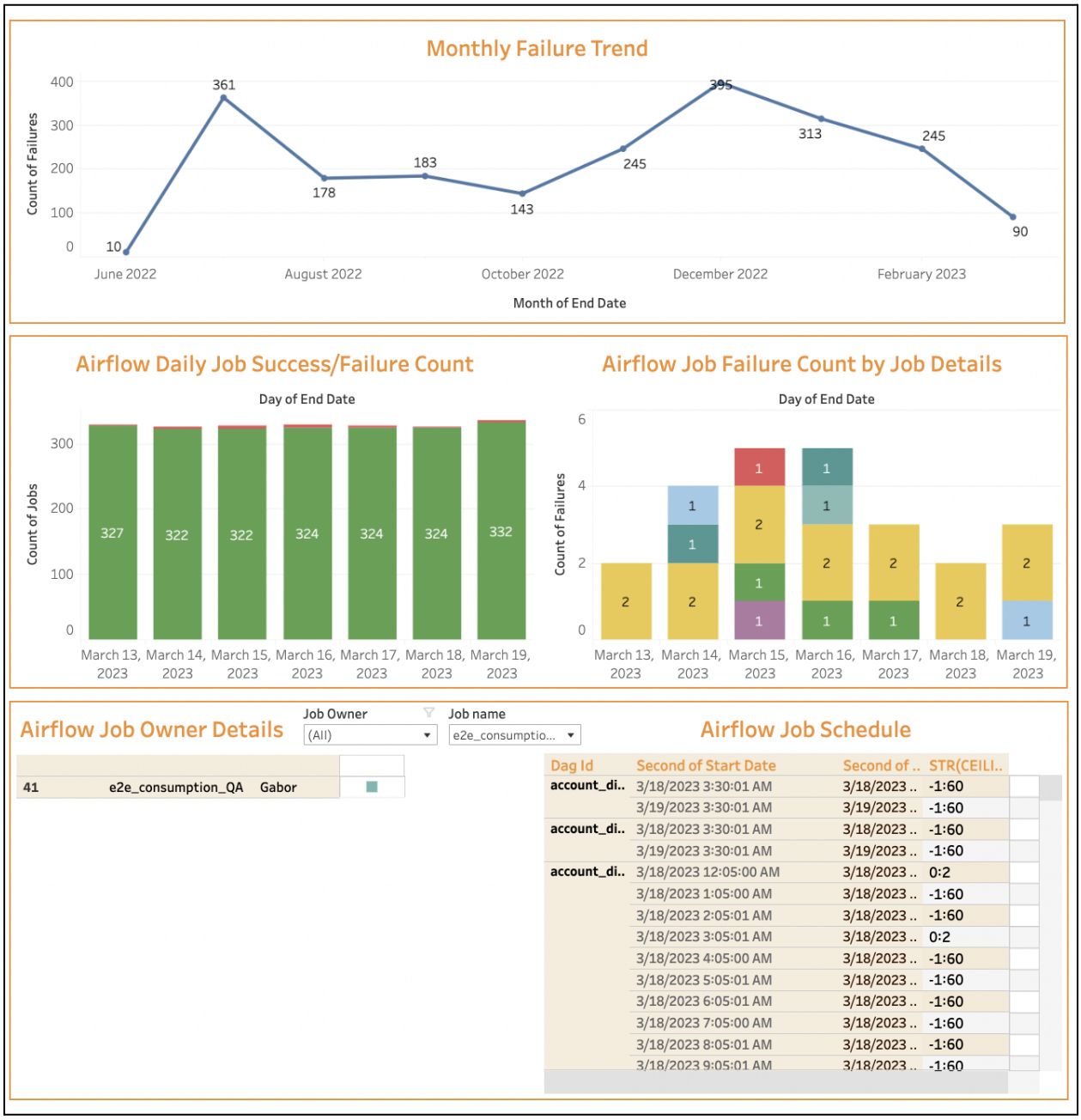
   b. You can use horizontal and vertical objects to provide a visual appeal to the dashboard and group different worksheet views together.

   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:



- Implementing 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 Airflow job failure data, and then sends an email alert with the failure details to the recipient.

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

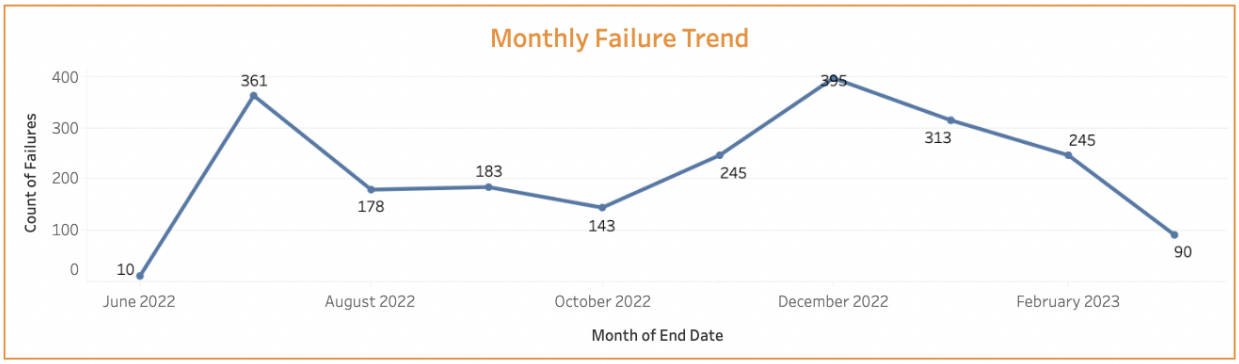
Airflow Dag Code: Provided at the end of this article

Note: The code provided is merely for reference purposes. Customize the airflow dag as 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 :

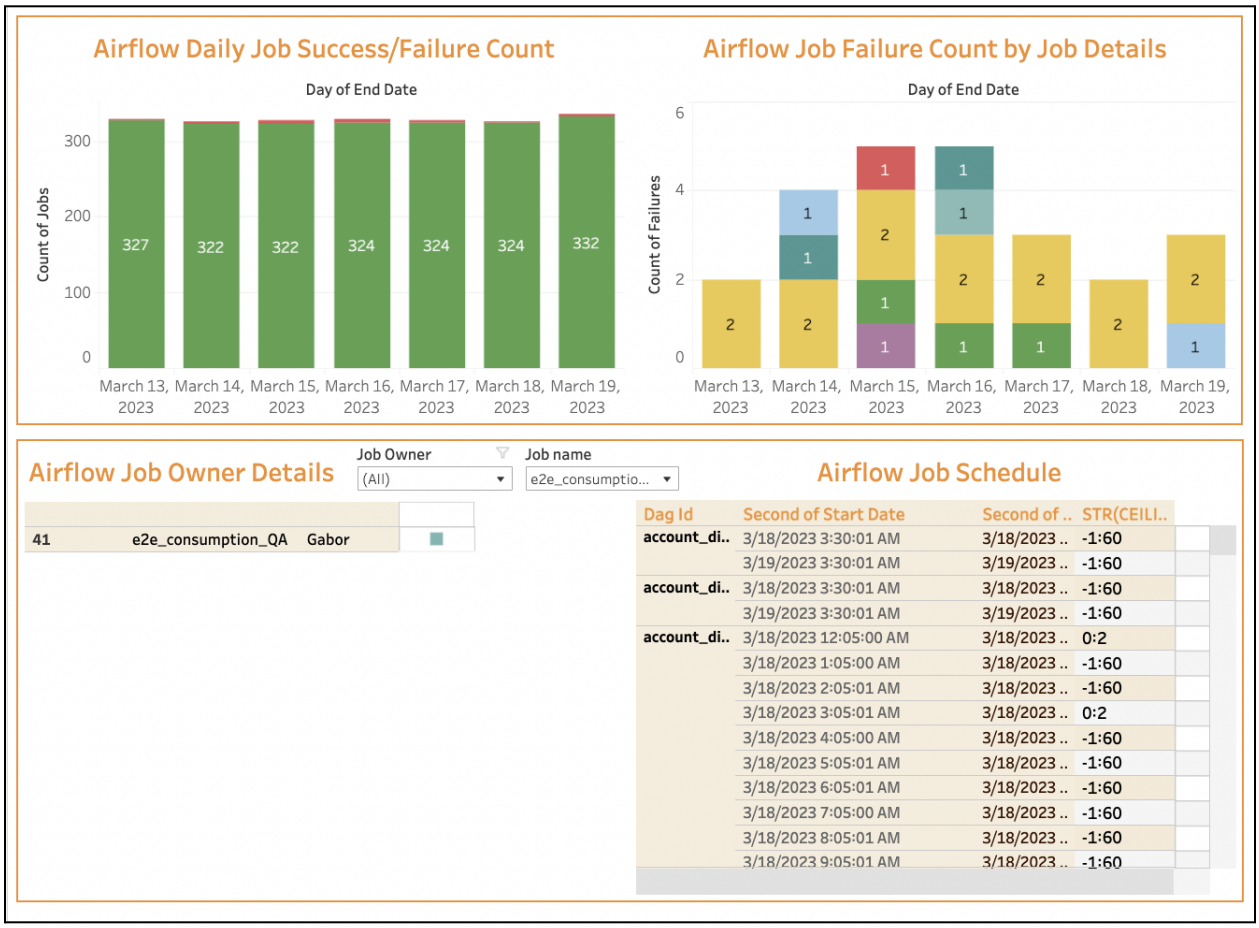


- Pay appropriate attention to any surge in the number of failures using the area chart above, and identify the source of the problem to understand which jobs and users

caused the spikes. Use the views in the next section to get job details for 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 job failures. If you observe a decrease in the failure trend it's a good indication that your Airflow job 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. Identifying the problem, spotting issue trends and taking action.



- "Airflow Daily Job Success/Failure Count" and "Airflow Job Failure Count By Job Details" : Keep a close eye on the total number of daily job failures and identify the dag names in the Details view to further troubleshoot.
    - Reference the total number of jobs getting executed within Airflow to get a basic idea of Load. Keep an eye out for surges in total and failed jobs count.

    - Assess the incline and decline patterns by comparing the current day count with the previous six days.
- Use the "Airflow Job Owner Details" view to identify the job owner name to report the issue and ask for action.
- Use the "Airflow Job Schedule" to assess the average job execution time for each dag schedule and compare it with previous schedule execution time. Pay attention to jobs with the highest execution time or odd increase/decrease in the completion time of a dag and action root causes.

Recommended Operational Processes

- Schedule daily monitoring calls to analyze Airflow job load and troubleshoot significant failures with the help of the dashboard views.
- KPIs to closely monitor daily :
    1. Increase or sudden spikes in the daily total number of airflow jobs executed and job failures.
    2. Users and Dag Names resulting in the highest number of job failures.
    3. An increase/odd behavior in the execution time of a dag.
    4. Should any of these failures prompt either job tuning or service tuning?
- Periodically assess job health progress utilizing the historic data trend view to reduce job failure rate.

Python

Python Script

```
import sys, psycopg2
import subprocess
from datetime import datetime
from subprocess import Popen, PIPE
import os, time, sys
import datetime
import glob
from datetime import datetime
date = datetime.now().strftime("%Y_%m_%d")

# Create the ticket for airflow user

kinit_args = [ '/bin/kinit', '-kt', '/home/home/airflow.keytab',
'airflow@domainname' ]
subp = Popen(kinit_args, stdin=PIPE, stdout=PIPE, stderr=PIPE)
subp.wait()

# delete the existing files from path

conn = psycopg2.connect("dbname='airflow_db' user='airflow_user' host='airflow
db hostname' password='airflow_user'")
cur = conn.cursor()
print('Connecting to Database')
query = """
       select dag_id,state,start_date, end_date, run_type from dag_run where
DATE(dag_run.end_date) = current_date - 1
"""
outputquery = "COPY ({0}) TO STDOUT WITH CSV DELIMITER '|'".format(query)
with open(f"/data/2/airflow_tmp/airflow_monitoring/airflow_jobs_{date}.csv",
"w") as file:
  cur.copy_expert(outputquery, file)
cur.close()
print('CSV File has been created')

# Remove the header

with open(f"/data/2/airflow_tmp/airflow_monitoring/airflow_jobs_{date}.csv",
"r") as fin:
       data = fin.read().splitlines(True)
```

```
with open(f"/data/2/airflow_tmp/airflow_monitoring/airflow_jobs_{date}.csv",
"w") as fout:
      fout.writelines(data[1:])

# put the file in HDFS

def run_cmd(args_list):
      """
      run linux commands
      """
      # import subprocess
      print('Running system command: {0}'.format(' '.join(args_list)))
      proc = subprocess.Popen(args_list, stdout=subprocess.PIPE,
stderr=subprocess.PIPE)
      s_output, s_err = proc.communicate()
      s_return =  proc.returncode
      return s_return, s_output, s_err


#Run Hadoop ls command in Python


(ret, out, err)= run_cmd(['hdfs', 'dfs', '-copyFromLocal',
f"/data/2/airflow_tmp/airflow_monitoring/airflow_jobs_{date}.csv",
'/edhoperations/airflow/'])
#lines = out.split('\n')
```

Python

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
from smtplib import SMTP
import smtplib
import sys

#Defining Default Arguments for the Dag
default_args = {
    'owner': 'airflow',
    'depends_on_past': False,
    'email': ['job owner email address'],
    'email_on_failure': True,
    'email_on_retry': False,
    'retries': 1,
    'retry_delay': timedelta(minutes=2)
}

doc = '''
## Airflow job monitoring dag to send an email to the end-user/team informing
about the  job failures that occurred in the last 24 hours.
'''

#Initiating the Dag

JOB_ID = 'Airflow_failure_job_Alert'

dag = DAG(
    dag_id=JOB_ID,
    doc_md=doc,
    default_args=default_args,
    description='Airflow_failure_job_Alert',
    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())
    print(df)
    recipients = ['recipient email address', 'recipient email address']
    emaillist = [elem.strip().split(',') for elem in recipients]
    print(emaillist)
    msg = MIMEMultipart()
    msg['Subject'] = "Alert : Airflow failed job status for Day -1 "
    msg['From'] = 'Sender Mail Address'


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


  Please fix the below Airflow jobs which failed yesterday ( Day -1 ). For more
detail visit to below dashboard -
             <Mention the link to the dashboard visualization>

    </pre>

        {0}

      <pre>

      Please reach out to the 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 HOSTNAME', 25)
    server.sendmail(msg['From'], emaillist , msg.as_string())



#Creating a Task

run_this = PythonOperator(
    task_id='Airflow_Alert_Email',
    python_callable=func,
    op_kwargs={'jdbc_conn_id': 'dcoe_impala', 'sql': 'select
dag_id,status,trunc(cast(end_date as timestamp), "dd") as finished_date from
edhoperations.airflow_jobs_details where status like "%failed%" and
trunc(cast(end_date  as timestamp), "dd") = date_sub(CURRENT_DATE(), interval 1
days);'},
    dag=dag,
)
```