# Transparency and Visibility
# Cron Job Alerting Implementation Steps

## Implementation

Implementing Cron Job Alerting includes these tasks:

1. Create External Hive Table
2. Deploy a Python Script
3. Implementing End User Alerting Mechanism.

- ### Create External Hive Table

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

Steps to perform: Hive table schema:

```
CREATE EXTERNAL TABLE edhoperations.edh_cron_monitor (   hostname STRING,
script_run_date STRING,   file_date STRING,   error STRING,   error_detail STRING )
PARTITIONED BY (   snapshottime BIGINT ) STORED AS PARQUET LOCATION
'hdfs://nameservice1/user/hive/warehouse/edhoperations.db/edh_cron_monitor'
```

- ### Deploying a Python Script

How it works: The python script reads across the cron job logs, retrieves the failed job details and loads the data in the HDFS location in CSV format which is accessed by the Hive table for querying.

Link to the shell script : [Python Script provided at the end of this article.]

Steps to Perform:

1. Place the python script on the host on which the cron jobs need to be monitored.
2. Schedule the python script via crontab on the same server and set it to run every day.

```
30 23 * * * python3 /opt/cops/cron_monitor/cron_test.py >
/opt/cops/cron_monitor_logs/cron_monitor_log-`date +\%F-\%H-\%M`.out
2>&1
```

- 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 cron 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: Dag code provided end of this article

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

Python

**Cron Job Python script**

```python
from  datetime import datetime, timedelta
import time
from distutils.log import ERROR
import re
import sys
from subprocess import PIPE, Popen
import os
import glob
import socket
import subprocess

hostname = socket.gethostname()
timestr = time.strftime("%Y-%m-%d")
file_out = hostname+"_"+timestr+".csv"
#print(file_out)
file_in = "cron"+"_"+timestr+".txt"
#print(file_in)
kinit_args = [ '/bin/kinit', '-kt', '/home/user/user.keytab',
'user@<DOMAINNAME>' ]
subp = Popen(kinit_args, stdin=PIPE, stdout=PIPE, stderr=PIPE)
subp.wait()
file_ = open(file_in, 'w+')
subprocess.run('crontab -l -u <username>', shell=True, stdout=file_)
file_.close()

content = file_in
now = datetime.now()
t = now.strftime("%b %-d")
#print(t)

with open(content) as f:
       lines = f.readlines()


dir_path = []
myfile = open('output.txt', 'w')
for l in lines:
       if '>' in l and not any(word in l for word in dir_path):
       command = re.findall(r"(?<=>).*", l)
```

```
        commandlist = ' '.join([str(elem) for elem in command])
        #print(command)
        location = os.path.dirname(commandlist)
        dir_path.append(location)
        #print(location)
        location_ext = "/*"
        final_loc = location + location_ext
        #print(location)
        files = glob.glob(location[1:]+"/*")
        #print(files)

        #modified_files = list()
        current_time = time.time()

        for csv_file in files:
                time_delta = current_time - os.path.getmtime(csv_file)
                time_delta_days = time_delta / (60 * 60 * 24)
                if time_delta_days < 2:
                #modified_files.append(csv_file)
                print(csv_file)
                try:
                textfile = open(csv_file, 'r')
                filetext = textfile.read()
                textfile.close()
                r = re.compile(r'\b.*caused.*\b' , flags=re.I | re.X)
                matches = r.findall(filetext)
                except Exception:
                        pass
                #matches = re.findall(r'.*Caused.*',filetext)
                if matches:
                        file_time = os.path.getmtime(csv_file)
                        matches_dummy = "Exception is big please check in log"
                        file_time_format =
(datetime.fromtimestamp(file_time).strftime('%Y-%m-%d %H:%M:%S'))
                        data = (hostname, '|', timestr ,'|',file_time_format, '|',
csv_file, '|', matches_dummy)
                        print(data)
                        myfile2 = open(file_out, 'a')
                        myfile2.write(str(data) + "\n")
                        myfile2.close()

myfile.close()
hdfs_path = os.path.join(os.sep, 'user', '<username>', file_out)
```

```
put = Popen(["hadoop", "fs", "-put", file_out, "/edhoperations/crontab/"],
stdin=PIPE, bufsize=-1)
put.communicate()
print("Remving the out file")
#os.remove(file_out)
print("Removed the out file and script run successfully")
```

Python

**Cron Job Airflow Dag**

```python
#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 owner email address'],
    'email_on_failure': True,
    'email_on_retry': False,
    'retries': 1,
    'retry_delay': timedelta(minutes=2)
}

doc = '''
'''
#Initiating the Dag

JOB_ID = 'Cron_monitoring'

dag = DAG(
    dag_id=JOB_ID,
    doc_md=doc,
    default_args=default_args,
    description='Cron_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'] = "Alert : <Hostname> Cron Jobs Failed  [Action Required]"
    msg['From'] = 'Sender Email Address'
    html = """\
    <html>
      <head></head>
      <body>
    <pre>
    Hello Team,

    Below cron jobs failed on <Hostname> in the last 24 hours.



    For more details on ERROR/Exception please check hosts logs :
/opt/cops/cron_monitor_logs
    </pre>

        {0}

      <pre>





      </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
hostname,script_run_date,file_date,error,script_run_date,error_detail from
edhoperations.edh_cron_monitor where snapshottime IN (select distinct
snapshottime from edhoperations.edh_cron_monitor order by snapshottime desc
limit 1) ;' },
    dag=dag,
)
```