MURAL Operations & Troubleshooting Guide

Version: 5.0.2.rc1

Last Updated: 18-01-2019

Americas Headquarters

Cisco Systems, Inc.

170 West Tasman Drive

San Jose, CA 95134-1706 USA http://www.cisco.com

Tel: 408 526-4000

800 553-NETS (6387) Fax: 408 527-0883

MURAL Operations & Troubleshooting Guide

MURAL Operations & Troubleshooting Guide

Table Of Contents

1. Introducing MURAL

The MURAL software is a resilient, scalable, and configurable system that powers a variety of solution suites, each tailored to ingest particular kinds of data streams and deliver insights that enable you to take action or that trigger an automatic action. Specialized software components collect high-volume data streams, then fuse, extract, process, correlate, analyze, and store the data, making it available to the MURAL software user interface or to third-party tools and applications. This software is highly scalable and feature-rich distributed platform, that is useful for developing and deploying applications conveniently.

2. Understanding MURAL Job Processes

The system receives vast and variety of data from various sources. Such data gets collected and ingested into the system as it is. MURAL jobs classify the data into different categories and remove or blacklist the unwanted data as required.

MURAL adds defined categories to the ingested data hence grouping the data into different categories such as arts and entertainment, social entertainment and so on. The master_http job runs on the ingested data and categorizes the data into the mentioned categories. Next, the data is read by master_http job which black-lists and whitelists the data as defined in the job as required. By blacklisting the data, the user will not be able to see the list of data which the user wants to remove. For example, you will not be able to view any adult related data in the complete data box when master_http job runs. The agg_5min and agg jobs runs on the aggregated data and provides an output as required by the user. This out-put data is free from all the unwanted fields and data. The data is collected for every 5 min in 5min aggregation bins.

Read the following table to know the frequency and stage of the running jobs:

Job Name	Frequency	Stage
master_http	Streaming, 5-min batch	Enrichment
master_nonhttp	Streaming, 5-min batch	Enrichment
agg_5min	5-min	Aggregation
agg_1hour	1-hour	Aggregation
agg_1day-dimensionTable	1-day	Aggregation
agg_1month	1-month	Aggregation
talendviaspark_http	1-min	Ingestion
talendviaspark_nonhttp	1-min	Ingestion
DataQualityODSGDSDailyAgg_http	1-day	DQM Report
DataQualityODSGDSDailyAgg_nonhttp	1-day	DQM Report
DataQualityODSGDSHourlyAgg_http	1-hour	DQM Report

Job Name	Frequency	Stage
DataQualityODSGDSHourlyAgg_non-http	1-hour	DQM Report
DataQualityODSGDSWeeklyAgg_http	1-week	DQM Report
DataQualityODSGDSWeeklyAgg_non-http	1-week	DQM Report
uncatDailyAgg_http	1-day	UNCAT Report
uncatDailyAgg_nonhttp	1-day	UNCAT Report
uncatHeavyhitter_http	5-min batch	UNCAT Report
uncatHeavyhitter_nonhttp	5-min batch	UNCAT Report
uncatHourlyAgg_http	1-hour	UNCAT Report
uncatHourlyAgg_nonhttp	1-hour	UNCAT Report
uncatWeeklyAgg_http	1-week	UNCAT Report
uncatWeeklyAgg_nonhttp	1-week	UNCAT Report

Logs Location:

All the running jobs are available at the following location:

```
/opt/azkaban/solo-server/executions/<talend-execution-id>/
```

Perform the following steps via command prompt:

1. Run the following command to check pod name:

   ```
   kubectl get pods -o wide
   ```

2. Run the following command to log into Azkaban pod:

   ```
   kubectl exec -it <azkaban-pod-name> -- /bin/bash
   ```

Perform the following steps on UI:

1. Navigate via Azkaban UI < Executing tab < Currently Running tab < Get Execution ID to check the execution ID of the running job.
2. Navigate via Azkaban UI < History tab < Duplicate <job name> in Search Bar < Get Execution ID to check the execution ID of the past run job.

For more information about input and output path of jobs, refer "Appendix A: MURAL Jobs and Services Log" on page 77

3. Updating the MURAL Job Processes

This section states the steps to update the configurations and resource file in MURAL jobs:

3.1 Updating the Job Configurations

In each of the following files, update the spark properties as required to access them on your system:

Files to be updated

- /opt/tms/java/DataMediationEngine/WEB-INF/classes/spark.properties
- /opt/tms/java/DataMediationEngine2/WEB-INF/classes/spark.properties

Spark Properties to be added

```
spark.executor.instances <resource parameter>
spark.executor.cores <resource parameter>
spark.executor.memory <resource parameter>
spark.driver.memory <resource parameter>
```

The output may resemble as follows:

```
spark.executor.instances 1
spark.executor.cores 1
spark.executor.memory 2g
spark.driver.memory 2g
```

3.2 Updating the Job Resource

To update the job resource on your system, add the following line on both the namenodes:

/etc/hadoop/conf/fair-scheduler.xml - changes of mrx_hadoop.yml

Run the following command to restart YARN services:

systemctl restart hadoop-yarn-resourcemanager.service

3.3 Updating the Job Properties

Perform the following steps to update job properties via Azkaban UI:

1. Log into the Azkaban UI from your web browser using the provided credentials.

   Projects screen will show up by default.

2. Click the required job available on **Projects** screen.
3. From the list of **Flows** appearing on the screen, click the drop down button to select the job.
4. Click the **Edit** tab on the top- right side of the table and edit the command in the value column as required.
5. Click **Set/Change Job Description** button to save the changes or **Cancel** to discard the changes.

The job properties are updated.

4. Updating Azkaban User Details

Perform the following steps on both Azkaban nodes to change UI user details on setup created using provisioner:

1. Run the following command to log into Azkaban nodes as a root user:

```
ssh root@<node ip>
```

2. Run the following command to stop pgha-watchdog-azkaban process on both the Azkaban nodes:

```
service pgha-watchdog-azkaban stop
```

3. Update users details on both the namenodes in `/opt/azkaban/solo-server/conf/azkaban-users.xml` file:

   The following sample file illustrates the commands to change Azkaban user password as admin:

```
<azkaban-users>
<user username="azkaban" password="admin" roles="admin"
groups="azkaban" />
<user username="metrics" password="metrics" roles="metrics"/>
<role name="admin" permissions="ADMIN" />
<role name="metrics" permissions="METRICS"/>
</azkaban-users>
```

4. Update username and password in `/etc/azkacli/azkacli.cfg` file on both the namenodes.

   Enter username as *azkaban* and password as *admin*.

5. Run the following command to start pgha-watchdog-azkaban process on both the Azkaban nodes:

```
service pgha-watchdog-azkaban start
```

5. Understanding MURAL Cluster

Cisco MURAL enables operators to gain a greater understanding of subscribers' interests to strengthen their competitive position against digital providers and find new revenue opportunities to overcome flat ARPU. It understands interest of every subscriber in real time to enable personalized engagement and interactions that increase customer loyalty and help grow the business. MURAL incorporates behavioral, contextual, and business data across data sources to form a complete view of each subscriber.

The MURAL Platform uses grid-computing architecture, with collection engines located at each data collection site to collect and analyse data. It operates in a fault-tolerant mode where the failure of any component is fully sustained by an alternate component. If any component fails, all components in all platforms adjust their configuration to work with the new topology.

The MURAL Platform lets you install applications on the bare metal machine wherein cloudera RPMs are installed and on the top of it MURAL application will run.

The MURAL application expects EDR data. The data is provided by Cisco and data changes depending on various parameters. The MURAL platform inter-operates with your existing network infrastructure and data repositories and does not require you to distribute proprietary data collection devices or probes in your network. It is a seven layer information for subscribers which is viewed mostly in a particular segment or region.

All the data is collected by a node at one place before sending it to MURAL. This data is pulled by utility run and collected by central depository, which is Cisco.

The following section explains the three types of nodes, which allows access to MURAL:

- **Namenode Cluster:** The data is initially pushed to local disk, then to HDFS. There are Talend jobs instead of collectors. The data is read by Talend. Talend will translate the DPI data to a Cisco understandable data.
- **Ingestion Cluster:** The main function of this node is to collect and ingest

the data based on coded, customized and configurable filters. It then pushes the data to compute node.

- **Compute Cluster:** They process and aggregate the data based on business logic.

6. Understanding IB framework

Read the following table to know the use, state and frequency for each listed IB name in text file type.

IB Name	Local/HDFS	Used by/ Used for	Static/ Dynamic	Frequency
/opt/ddj/conf/ggsn.xml	Local & HDFS	master job, for enrichment of ggsn	Static	On demand
/opt/ddj/conf/location.xml	Local & HDFS	geographic location of Cell ID or Mobile tower	Static	On demand
/opt/ddj/conf/roamer.xml	Local & HDFS	MCC-MNC ID to mobile operator mapping	Static	On demand
/opt/ddj/conf/sgsn.xml	Local & HDFS	master job, for enrichment of sgsn	Static	On demand
/opt/ddj/IBs/snAppCat.map	Local & HDFS	master job, for traffic type categorization	Static	On demand
/opt/ddj/IBs/p2pCat.map	Local & HDFS	master job, for traffic type categorization	Static	On demand

IB Name	Local/HDFS	Used by/ Used for	Static/ Dynamic	Frequency
/opt/ddj/IBs/portProtoAppPath.map	Local & HDFS	master job, for traffic type categorization	Static	On demand
/opt/mrx/ingestion/etc/ratidtype.map	Local	rat annotation at ingestion level	Static	On demand
/opt/mrx/ingestion/etc/ipProto.map	Local	protocol annonation at ingestion level	Static	On demand
/opt/mrx/ingestion/etc/snAppIdStr.map	Local	snaAppId	Static	On demand
/opt/sevenflow/conf/rulesEngine.xml	Local & HDFS	to know the IBs path	Static	On demand
/opt/sevenflow/conf/service.xml	Local & HDFS	global data services (gds) rules	Static	On demand
/opt/sevenflow/conf/operatorService.xml	Local & HDFS	operator data services (ods) rules	Static	On demand

7. Viewing Hadoop Cluster Details

MURAL allows you to check the Hadoop cluster details via command prompt.

Run the following command on the master namenode to view and check the jib processes:

[root@mrx001-mst-02 ~]# ps -eaf | grep -i <jobname>

Ensure the following:

- All data nodes must be live. No dead node should exist.
- Safe mode must be enabled.
- DFS storage must be less than 80 percentile.
- Under replicated blocks and blocks with corrupt replicas must not be more than 1. Contact Cisco Support, if the value exceeds 1.

Read the following sample file for reference:

```
[root@mrx001-mst-01 log]# hadoop dfsadmin -report
DEPRECATED: Use of this script to execute hdfs command is
deprecated.
Instead use the hdfs command for it.
Configured Capacity: 1212657172480 (1.10 TB)
Present Capacity: 1210679600154 (1.10 TB)
DFS Remaining: 1168805761693 (1.06 TB)
DFS Used: 41873838461 (39.00 GB)
DFS Used%: 3.46%
Under replicated blocks: 1
Blocks with corrupt replicas: 1
Missing blocks: 1
Missing blocks (with replication factor 1): 0
-------------------------------------------------
Live datanodes (3):

Name: 192.168.133.125:50010 (mrx001-slv-01.gvs.ggn)
Hostname: mrx001-slv-01.gvs.ggn
```

```
Rack: /rack-8
Decommission Status : Normal
Configured Capacity: 242531434496 (225.88 GB)
DFS Used: 11947967524 (11.13 GB)
Non DFS Used: 0 (0 B)
DFS Remaining: 229863116089 (214.08 GB)
DFS Used%: 4.93%
DFS Remaining%: 94.78%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 14
Last contact: Thu Oct 04 09:13:48 UTC 2018

Name: 192.168.133.126:50010 (mrx001-slv-02.gvs.ggn)
Hostname: mrx001-slv-02.gvs.ggn
Rack: /rack-6
Decommission Status : Normal
Configured Capacity: 485062868992 (451.75 GB)
DFS Used: 14516404898 (13.52 GB)
Non DFS Used: 0 (0 B)
DFS Remaining: 470073306296 (437.79 GB)
DFS Used%: 2.99%
DFS Remaining%: 96.91%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 10
Last contact: Thu Oct 04 09:13:48 UTC 2018
```



```
Name: 192.168.133.127:50010 (mrx001-slv-03.gvs.ggn)
Hostname: mrx001-slv-03.gvs.ggn
Rack: /rack-5
Decommission Status : Normal
Configured Capacity: 485062868992 (451.75 GB)
DFS Used: 15409466039 (14.35 GB)
Non DFS Used: 0 (0 B)
DFS Remaining: 468869339308 (436.67 GB)
DFS Used%: 3.18%
DFS Remaining%: 96.66%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 14
Last contact: Thu Oct 04 09:13:48 UTC 2018

[root@mrx001-mst-01 log]#
```

8. Managing Kafka Cluster

Kafka cluster requires to be managed in order to keep it balanced and even. Expansion or reduction in Kafka cluster leads to discrepancies in the cluster and performance issues. Read the following section to know how to manage a Kafka cluster.

8.1 Expanding a Kafka Cluster

The partition reassignment tool is used to expand an existing Kafka cluster. Cluster expansion includes brokers with new broker IDs in a Kafka cluster. The tool must run to assign existing topics/ partitions to the new brokers in order to receive data from the existing Kafka- topic. The tool allows two options to make it easier to move Kafka-topics in bulk to the new brokers.

The previously mentioned options are as follows:

- topics to move
- list of newly added brokers

Using the above mentioned two options, the tool automatically figures out the placements of partitions for the topics on the new brokers and generates new JSON data which can be used in the next step to execute the move.

The following example moves two topics (foo1, foo2) to newly added brokers in a cluster (5,6,7):

```
$ ./bin/kafka-reassign-partitions.sh --topics-to-move-json-file
topics-to-move.json --broker-list "5,6,7" --generate
$ cat topics-to-move.json {"topics": [{"topic": "foo1"},{"topic":
"foo2"}], "version":1}
```

8.2 Reducing a Kafka Cluster

Perform the following steps to balance Kafka cluster in case of reduction:

1. Use partition reassignment *kafka-reassign-partition.sh* tool to generate the candidate assignment configuration. This shows the current and the

proposed replica assignments.

2. Copy the proposed assignment to a JSON file.
3. Execute the partition reassignment *kafka-reassign-partition.sh –execute* tool to update the metadata for balancing.

   Ensure you run the mentioned tool when there is less load on the clusters, as this moves the data between different available nodes.

   Wait for few minutes based on the amount of data has to be moved.
4. Click **Verify** button to verify the completion of balancing.
5. Once the partition reassignment is completed, run *"kafka-preferred-replica-election.sh"* tool to complete the balancing.

Note: Maximum one node can be down at a time or else Kafka cluster will not be able to perform.

The preceding section explains re-balancing a Kafka cluster automatically. If these steps fail to resolve the issue then, refer Managing Kafka Cluster to manually resolve the issue.

9. Managing Redis Node

This sections explains about managing and removing Redis node from clusters.

9.1 Prerequisites

Read the following section to ensure minimum requirements of removing Redis node from cluster:

- Redis cluster must be created using platform installer guide *MURAL 5.0.1.rc1 Installation Guide.*
- Ensure that a minimum number of 3 Redis nodes are available in a cluster.
- Setup management node and all Redis nodes must be up with all expected configuration and services comes with platform installer. Refer *MURAL 5.01.rc1 Installation Guide #VerifyPlatformInstallation* to verify the nodes status.
- Configuration of Redis master/slave nodes completed in circular loop with each node running one master and one slave.

Node 1	Node 2	Node 3
Master 1	Master 2	Master 3
Slave 3	Salve 1	Salve 2

- The master node that needs to be removed must be empty.

9.2 Removing Slave Node from Existing Master Node

Perform the following steps to remove slave node of deleting master node:

1. Log into the management node as root user.
2. Run the following command to remove Redis slave node (S1) from the Redis cluster:

```
# redis-trib del-node <ip address node1> <node name node1>
```

The output may resemble the following:

```
>>> Removing node <node name node1> from cluster <ip address
node1>
>>> Sending CLUSTER FORGET messages to the cluster...
>>> SHUTDOWN the node.
```

3. Run the following command to verify that Redis cluster check command does not show the removed node id:

```
# redis-trib check <ip address node1>
```

4. Login to slave node (<ip address node1>) as root user.
5. Run the following command to stop Redis slave service:

```
# systemctl stop redis_6380
```

6. Run the following command to check slave service status:

```
# systemctl status redis_6380
```

7. Run the following command to remove slave snapshot and node configuration files:

```
# rm -rf /var/lib/redis/6380/*
```

8. Run the following command to remove slave node logs:

```
# rm -rf /var/log/redis/node_6380.log*
```

For example,

Node 1	Node 2	Node 3	Node 4	Node 5	Node 6
1-M	2-M	3-M	4-M	5-M	6-M
6-S (DELETE REPLICA)	1-S	2-S	3-S	4-S	5-S

9.3 Migrating Slave Node of Existing Master

Before removing the node, it is necessary to migrate slave node to other existing master node.

Perform the following steps to migrate salve node of existing master:

1. Log into setup management node as root user.
2. Migrate Redis slave node from 192.168.133.19:6380 to 192.168.133.14:6380.
3. Run the following command to remove Redis slave (S: d92da175c221b-fc52bee3e6357007a28234881fe 192.168.133.19:6380) from the Redis cluster:

```
# redis-trib del-node 192.168.133.19:6380
d92da175c221bfc52bee3e6357007a28234881fe
```

   The output may resemble the following:

```
>>> Removing node d92da175c221bfc52bee3e6357007a28234881fe
from cluster 192.168.133.19:6380
>>> Sending CLUSTER FORGET messages to the cluster...
>>> SHUTDOWN the node.
```

4. Run the following command to verify that Redis cluster check command does not show the removed node id:

```
# redis-trib check 192.168.133.14:6379
```

5. Log in to slave node (192.168.133.19) using root user.
6. Run the following command to stop Redis slave service:

```
# systemctl stop redis_6380
# systemctl disable redis_6380
```

7. Run the following command to check slave service status:

```
# systemctl status redis_6380
```

8. Run the following command to remove slave snapshot and node configuration files:

```
# rm -rf /var/lib/redis/6380/*
```

9. Run the following command to remove slave node logs:


```
# rm -rf /var/log/redis/node_6380.log*
```

For Example,

192.168.1-33.14	192.168.1-33.15	192.168.1-33.16	192.168.1-33.17	192.168.1-33.18	192.168.1-33.19
1-M	2-M	3-M	4-M	5-M	6-M
	1-S	2-S	3-S	4-S	**5-S (DELETE)**

10. Log into slave node (<IP address>) as root user.
11. Run the following command to start slave service on it:

```
# systemctl start redis_6380
```

12. Run the following command to check slave service status:

```
# systemctl status redis_6380
```

13. Run the following command to verify `node.conf` file:

```
# cat /var/lib/redis/6380/nodes.conf
<name node> :0 myself,master - 0 0 0 connected
vars currentEpoch 0 lastVoteEpoch 0
```

14. Add slave node again in the cluster as slave of master node (M: c72918a56885cdded8740cd0a9c2bfcc5ffb3745 192.168.133.18:6379)
15. Run the following command to log into management node as root user:

```
# redis-trib add-node --slave --master-id
c72918a56885cdded8740cd0a9c2bfcc5ffb3745
```

For Example,

192.168.1-33.14	192.168.1-33.15	192.168.1-33.16	192.168.1-33.17	192.168.1-33.18	192.168.1-33.19
1-M	2-M	3-M	4-M	5-M	6-M
5-S (ADD REP-LICA)	1-S	2-S	3-S	4-S	

The output may resemble the following:

```
>>> Adding node 192.168.133.14:6380 to cluster
192.168.133.18:6379
>>> Performing Cluster Check (using node 192.168.133.18:6379)
M: c72918a56885cdded8740cd0a9c2bfcc5ffb3745
192.168.133.18:6379
slots:13607-16383 (2777 slots) master
0 additional replica(s)
S: 78954cba170ceafed1f85b23c543eddf1087c6b1
192.168.133.15:6380
slots: (0 slots) slave
replicates b87002614c51b5ceaba894fa0441627d621622e8
M: b87002614c51b5ceaba894fa0441627d621622e8
192.168.133.14:6379
slots:500-3276 (2777 slots) master
1 additional replica(s)
M: 2f7f6cff0b1f6323f8685fc492915483729dabba
192.168.133.15:6379
slots:10287-13106 (2820 slots) master
1 additional replica(s)
S: 510f73345809a56ebb75eed70d0693fe89c18a64
192.168.133.17:6380
slots: (0 slots) slave
replicates 7543151e59e1da79dc2b1d367a89a377a1f1f810
M: ef0bd5653797714ef965fb0cda96ac7a46984296
```

```
192.168.133.17:6379
slots:0-499,3277-4104,6554-7052,10114-10286,13107-13606 (2500
slots) master
1 additional replica(s)
S: 10b39fad6ed55888cc78119ff2b20ea71ed3fc26
192.168.133.16:6380
slots: (0 slots) slave
replicates 2f7f6cff0b1f6323f8685fc492915483729dabba
S: 986a7324547c0e5b5c0bea79c62add1569b378dc
192.168.133.18:6380
slots: (0 slots) slave
replicates ef0bd5653797714ef965fb0cda96ac7a46984296
M: 9007da35e4df889e0dc4a4f7f5a4ce98b7104f27
192.168.133.19:6379
slots:4105-6553,9830-10113 (2733 slots) master
0 additional replica(s)
M: 7543151e59e1da79dc2b1d367a89a377a1f1f810
192.168.133.16:6379
slots:7053-9829 (2777 slots) master
1 additional replica(s)
[OK] All nodes agree about slots configuration.
>>> Check for open slots...
>>> Check slots coverage...
[OK] All 16384 slots covered.
>>> Send CLUSTER MEET to node
>>> Configure node as replica of 192.168.133.18:6379.
[OK] New node added correctly.
```

16. Run the following command to verify that Redis cluster check command does show the added node id (c832428be52fcfc53212b-f68fd43b14da0f36793):

```
# redis-trib check 192.168.133.14:6379
```

The output may resemble as follows:


```
S: c832428be52fcfc53212bf68fd43b14da0f36793
192.168.133.14:6380
slots: (0 slots) slave
replicates c72918a56885cdded8740cd0a9c2bfcc5ffb3745
```

9.4 Removing Master Node

Once the slave node is migrated to other existing master node, then you can remove the master node from the cluster.

Perform the following steps to remove master node:

1. Remove Redis master node (M: 9007da35e4d-f889e0dc4a4f7f5a4ce98b7104f27 192.168.133.19:6379) from the Redis cluster.
2. Log in to management node as root user. In order to remove a master node it must be empty.
3. Run the following command to move all the shards from deleted master node id (9007da35e4df889e0dc4a4f7f5a4ce98b7104f27 192.168.133.19:6379) to any available master node id (M: 7543151e59e1da79dc2b1d367a89a377a1f1f810 192.168.133.16:6379):

   ```
   # redis-trib reshard --yes --from
   9007da35e4df889e0dc4a4f7f5a4ce98b7104f27 --to
   7543151e59e1da79dc2b1d367a89a377a1f1f810  --slots 16384
   192.168.133.14:6379
   ```

4. Run the following command to verify all slots of 192.168.133.19:6379 assigned to 192.168.133.16:6379 and all 16384 slots covered:

   ```
   # redis-trib check 192.168.133.14:6379
   ```

   The output may resemble the following:

   ```
   # redis-trib check 192.168.133.14:6379
   >>> Performing Cluster Check (using node 192.168.133.14:6379)
   ```



```
M: b87002614c51b5ceaba894fa0441627d621622e8
192.168.133.14:6379
slots:500-3276 (2777 slots) master
1 additional replica(s)
M: 9007da35e4df889e0dc4a4f7f5a4ce98b7104f27
192.168.133.19:6379
slots: (0 slots) master
0 additional replica(s)
S: 986a7324547c0e5b5c0bea79c62add1569b378dc
192.168.133.18:6380
slots: (0 slots) slave
replicates ef0bd5653797714ef965fb0cda96ac7a46984296
S: 10b39fad6ed55888cc78119ff2b20ea71ed3fc26
192.168.133.16:6380
slots: (0 slots) slave
replicates 2f7f6cff0b1f6323f8685fc492915483729dabba
M: 7543151e59e1da79dc2b1d367a89a377a1f1f810
192.168.133.16:6379
slots:4105-6553,7053-10113 (5510 slots) master
1 additional replica(s)
S: c832428be52fcfc53212bf68fd43b14da0f36793
192.168.133.14:6380slots: (0 slots) slave
replicates c72918a56885cdded8740cd0a9c2bfcc5ffb3745
S: 510f73345809a56ebb75eed70d0693fe89c18a64
192.168.133.17:6380
slots: (0 slots) slave
replicates 7543151e59e1da79dc2b1d367a89a377a1f1f810
M: c72918a56885cdded8740cd0a9c2bfcc5ffb3745
192.168.133.18:6379
slots:13607-16383 (2777 slots) master
1 additional replica(s)
M: ef0bd5653797714ef965fb0cda96ac7a46984296
```



```
192.168.133.17:6379
slots:0-499,3277-4104,6554-7052,10114-10286,13107-13606 (2500
slots) master
1 additional replica(s)
M: 2f7f6cff0b1f6323f8685fc492915483729dabba
192.168.133.15:6379
slots:10287-13106 (2820 slots) master
1 additional replica(s)
S: 78954cba170ceafed1f85b23c543eddf1087c6b1
192.168.133.15:6380
slots: (0 slots) slave
replicates b87002614c51b5ceaba894fa0441627d621622e8
[OK] All nodes agree about slots configuration.
>>> Check for open slots...
>>> Check slots coverage...
[OK] All 16384 slots covered.
```

5. Run the following command to remove Redis master node (M: 9007da35e4df889e0dc4a4f7f5a4ce98b7104f27 192.168.133.19:6379) from the Redis cluster:

```
# redis-trib del-node 192.168.133.19:6379
9007da35e4df889e0dc4a4f7f5a4ce98b7104f27
```

   The output may resemble the following:

```
>>> Removing node 9007da35e4df889e0dc4a4f7f5a4ce98b7104f27
from cluster 192.168.133.19:6379
>>> Sending CLUSTER FORGET messages to the cluster...
>>> SHUTDOWN the node.
```

6. Run the following command to verify that Redis cluster check command does not show the removed node id:

```
# redis-trib check 192.168.133.14:6379
```

7. Log in to Redis master node (192.168.133.19) using root user.

8. Run the following command to stop Redis master service:

```
# systemctl stop redis_6379
# systemctl disable redis_6379
```

9. Run the following command to check master service status:

```
# systemctl status redis_6379
```

For Example,

192.168.1-33.14	192.168.1-33.15	192.168.1-33.16	192.168.1-33.17	192.168.1-33.18	192.168.1-33.19
1-M	2-M	3-M	4-M	5-M	**6-M (DELETE)**
5-S	1-S	2-S	3-S	4-S	

The output may resemble the following:

```
# systemctl status redis_6379
● redis_6379.service - Redis persistent key-value database -
port 6379
Loaded: loaded (/usr/lib/systemd/system/redis_6379.service;
disabled; vendor preset: disabled)
Active: inactive (dead)
Oct 30 15:05:51 devops002-dn-06.gvs.ggn systemd[1]: Started
Redis persistent key-value database - port 6379.
Oct 30 15:05:51 devops002-dn-06.gvs.ggn systemd[1]: Starting
Redis persistent key-value database - port 6379...
Oct 31 09:16:35 devops002-dn-06.gvs.ggn systemd[1]: redis_
6379.service: control process exited, code=exited status=203
Oct 31 09:16:35 devops002-dn-06.gvs.ggn systemd[1]: Unit
redis_6379.service entered failed state.
Oct 31 09:16:35 devops002-dn-06.gvs.ggn systemd[1]: redis_
6379.service failed.
```



```
Oct 31 09:17:33 devops002-dn-06.gvs.ggn systemd[1]: Stopped
Redis persistent key-value database - port 6379.
```

10. Run the following command to remove master snapshot and node configuration files:

```
# rm -rf /var/lib/redis/6379/*
```

11. Run the following command to remove master node logs:

```
# rm -rf /var/log/redis/node_6379.log*
```

10. Monitoring MURAL

Mural provides access to various functions to the user. It is required to monitor MURAL to view the status of the software.

You can monitor jobs, nodes, and the health of the MURAL cluster by using the following interfaces provided by:

- Azkaban User Interface
- Hadoop User Interface
- Yarn User Interface

10.1 Using Azkaban User Interface

Azkaban is used to solve the problem of Hadoop job dependencies. It supports the jobs needs to run in order. It uses MySQL to store its state. Both the AzkabanWebServer and the AzkabanExecutorServer access the database.

10.1.1 Prerequisites

- All the name nodes must be up and running.
- Azkaban UI should be accessible with the required project uploaded.

10.1.2 Log in to Azkaban UI

To log into Azkaban:

1. Enter *URL- <Master NN IP>:8507* in your web browser.
2. Enter **azkaban** and **!4zk4b4n$** as username and password respectively.

The Azkaban user interface is launched.

10.1.3 Understanding the Azkaban Web UI

The following section explains the various measures displayed on the Azkaban UI:

Monitoring Projects

You can see list of configuring jobs. It retrieves project files from the database.

The following table describes the features available on the screen:

Field	Description
Quick Search	This option provides you to search from list of databases.
Create Project	This option provides you an option to create a new project.
Personal	This option displays the projects formed by you.
Group	This option displays the projects formed in a group.
All	This option displays all the projects available.

Monitoring Scheduling

This measure shows the state of the scheduled jobs. You can see the scheduled jobs by selecting this measure. To know about scheduling of jobs, refer "Appendix C: Scheduling Jobs Using User Interface" on page 83.

The following table describes the features available on the screen:

Field	Description
ID	This displays the ID of the job.
Schedule Status	This displays the status of the scheduled job.
Project	This displays the project of the job.
Flow	This displays the flow of the job.
Scheduled name	This displays the name of the scheduled job.
First Schedule	This displays the first schedule of the job.
Next Schedule	This displays the next schedule of the job, if any.
Next in	This displays the next job.

Field	Description
Repeats Every	This displays the repetition of every job.
Action	This displays the action of the job.

Monitoring Status

This measure keeps the track of executing flows and also shows the executor, who is running them. You can see the status of the jobs listed- running, finished or failed.

The following table describes the features available on the screen:

Field	Description
Execution ID	This displays the ID of the job.
Project	This displays the project of the job.
Flow	This displays the flow of the job.
User	This displays the name of the user.
Proxy	This displays the name of the proxy user.
Start Time	This displays the start time.
End Time	This displays the end time.
Elapsed Time	This displays the elapsed time.
Status	This displays the status of the job.
Action	This displays the action of the job.

Viewing History

This measure keeps the historical track of completed and successful jobs. You can see the history of the jobs which were running or success.

10.1.4 External Reference

For more information about using Azkaban user interface, refer *http://azkaban.-github.io/azkaban/docs/latest/*

10.2 Using Yarn User Interface

You can use Hadoop user interface for a periodic job, analyze its history logs and predict its resource requirement for the new run. It support various types of job logs. It scale to terabytes of job logs.

10.2.1 Prerequisites

- Hadoop must be up and running.
- It can be checked through user interface (port:50070).

10.2.2 Log in to YARN UI

To log into YARN UI:

- Enter *URL- <Master NN IP>:8088* in your web browser.

The YARN user interface is launched.

10.2.3 Understanding the YARN Web UI

The following table explains the various measures displayed on the YARN Web UI.

Field	Description
Cluster Metrics	It displays the status of the jobs.
Cluster Nodes Metrics	It displays the status of the nodes.
User Metrics	It displays the list of the IDs, User, Name, Application Type, Queue, Start Time, Finish Time, State, Final Status, Running Containers, Allocated CPU, Allocated Memory, Progress and Tracking UI os the job.

10.2.4 External Reference

For more information about using Hadoop user interface, refer *https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-*

hdfs/HdfsUserGuide.html

10.3 Using Hadoop User Interface

Hadoop is the primary distributed storage used by Hadoop applications. A Hadoop cluster primarily consists of a namenode that manages the file system metadata and datanodes that stores the actual data.

10.3.1 Prerequisites

- Hadoop must be up and running.
- Hadoop must be configured on all the machines.

10.3.2 Log into Hadoop UI

To log into Hadoop UI:

1. Enter *URL- <Master/Standby NN IP>:50070* in your web browser.

The Hadoop user interface is launched.

10.3.3 Understanding the Hadoop Web UI

The following table explains the various measures displayed on the Hadoop Web UI.

Field	Description
Overview	It displays various information about the platform. For example: Name space, Namenode id, Started, version, Compiled, Cluster Id and Block Pool Id.
Summary	It provides the summary and status of all the running jobs.
NameNode Journal Status	It provides the status of the journal manager. It also displays Current Transaction ID.

Field	Description
NameNode Storage	It provides the status and type of the storage directory.
DFS Storage Types	HDFS storage types can be used to assign data to different types of physical storage media. The following storage types are available: • DISK -- Disk drive storage (default storage type) • ARCHIVE -- Archival storage (high storage density, low processing resources)

10.3.4 External References

For more information about using HDFS user interface, refer *https://hadoop.apache.org/docs/r2.7.1/hadoop-project-dist/hadoop-hdfs/ArchivalStorage.html*

11. Monitoring System Metrics

The following section explains the naming conventions and list of counters available in the Rest APIs.

11.1 Naming Convention of Counters

The following section illustrates the naming convention of metrics standards in MURAL. The modules which sends the data to Grafana are Ingestion, and Master jobs.

```
<APP_FAMILY>.<APP_NAME>.<DATA_DOMAIN>.<METRIC_NAME>(.<METRIC_
BREAKDOWN.*>)
```

For example,

```
APP_FAMILY = mrx
APP_NAME = ingestion, ddj, dme, aggregation_5min, aggregation_
hourly, aggregation_daily
DATA_DOMAIN = edrhttp, edrnonhttp, guavus, radcom, httppdm, non-
httppdm
METRIC_NAME = input_records, output_records, invalid_reason1,
invalid_reason2
METRIC_BREAKDOWN = count, rate
```

Note: By following the METRIC_NAME convention, the user can aggregate all invalid records by using wild-cards invalid_* in grafana.

The following table lists the counter type, their description and the reason for their occurrence.

Counters	Description
sumSeriesWithWildcards(mrx.ingestion.att_edr.*. Drop_msisdn.count,3)	The record drops because the MSISDN (field #2) is either empty or zero.

Counters	Description
sumSeriesWithWildcards(mrx.ingestion.att_edr.*. Drop_recorsTime.count,3)	The record drops because all three of field #1, start (field #9) and end (field #10) timestamps are either invalid or empty.
sumSeriesWithWildcards(mrx.ingestion.att_edr.*. Invalid_ flowStartTime.count,3)	The start time (field #9) as empty or invalid. It was copied from field #1 or field #10.
sumSeriesWithWildcards(mrx.ingestion.att_edr.*. Invalid_ flowEndTime.count,3).	There are the following two possibilities: • The end time (field #10) is empty or invalid. It is copied from the field #1 or field #10. • The end time is less than the begin time.
sumSeriesWithWildcards(mrx.ingestion.att_edr.*. Invalid_upBytes.count,3)	Field #12 is empty or not a number. Replacing value by default value.
sumSeriesWithWildcards(mrx.ingestion.att_edr.*. Invalid_downBytes.count,3)	Field #11 is empty or not a number. Replacing value by default value.
sumSeriesWithWildcards(mrx.ingestion.att_edr.*. Invalid_ httpReturnCode.count,3)	Field #19 did not start with three digits. HTTP return code is set to null in the output.
sumSeriesWithWildcards(mrx.ingestion.att_edr.*. Invalid_subscriberIpAddress.count,3)	Subscriber IP address (field #14) is invalid. It had to be set to 0.0.0.0
sumSeriesWithWildcards(mrx.ingestion.att_edr.*. Invalid_serverIpAddress.count,3)	Server IP address (field #16) is invalid. It has to be set to 0.0.0.0
sumSeriesWithWildcards(mrx.ingestion.att_edr.*. Invalid_subscriberPort.count,3)	Subscriber port (field #15) is not an integer. It has to be set to 0.

Counters	Description
sumSeriesWithWildcards(mrx.ingestion.att_edr.*. Invalid_SGSNIP.-count,3)	Field #5 is invalid. It has to be set to 0.0.0.0
sumSeriesWithWildcards(mrx.ingestion.att_edr.*. Input_Records.count,3)	It shows the total number of records.
mrx.dataDomain.http.invalid_systemDroppedRecords.count	The DDJ http job drops due to invalid records.
mrx.dataDomain.http.output_uncatRecords.count	The DDJ http job reports uncategorised records.
mrx.dataDomain.http.input_records.count	It shows the total number of records.
mrx.dataMediation.http.output_blackListedRecords.count	The DME http job drops due to black listed records.
mrx.dataMediation.http.Invalid_httpsRecords.count	The DME http job drops due to http records.
mrx.dataMediation.http.input_records.count	It shows the total number of input records.

12. Starting and Stopping the Services

You can start, stop or restart the services via ansible script. Perform the following actions to start or stop the services.

Refer "Appendix E: List of Services" on page 91 to know the list of services.

12.1 Starting a Service

Run the following command to start a service via ansible script:

```
ssh root@<node-ip>
systemctl start <service-name>
```

You can also start all services at once. Run the following playbook with the command as follows to start all the services:

```
ansible-playbook -i inventory/generated/prod/mural001/hosts
playbooks/platform/ops/start-platform-ha.yml --user root --ask-pass
```

12.2 Stopping a Service

Run the following command to stop a service via ansible script:

```
ssh root@<node-ip>
systemctl stop <service-name>
```

You can also stop all services at once. Run the following playbook with the command as follows to stop all the services:

```
ansible-playbook -i inventory/generated/prod/mural001/hosts
playbooks/platform/ops/stop-platform-ha.yml --user root --ask-pass
```

12.3 Restarting a Service

Run the following command to restart a service via ansible script:

```
ssh root@<node-ip>
systemctl restart <service-name>
```


12.4 Viewing the Status of a Service

Run the following command to view the status of a service via ansible script:

ssh root@<node-ip>

systemctl status <service-name>

You can also view the status of all services at once. Run the following command via ansible script to view the status of all the services:

[root@mural001-mgt-01 reflex-provisioner]# ansible-playbook -i inventory/generated/prod/mural001/hosts playbooks/status_ services.yml --user root --ask-pass

The output may resemble as follows:

```
TASK [status_services : Print Status]
*****************************************
ok: [mural001-slv-02.gvs.ggn] => {
"msg": " hadoop-hdfs-datanode = RUNNING & ENABLED,  hadoop-hdfs-
journalnode = RUNNING & ENABLED,  hadoop-yarn-nodemanager = RUNNING
& ENABLED,  hbase-regionserver = RUNNING & ENABLED, impala-server =
||--- NOT RUNNING & ENABLED ---||, kafka-server = ||--- NOT RUNNING
& ENABLED ---||,  zookeeper-server = RUNNING & ENABLED,  docker =
RUNNING & ENABLED,  collectd-docker = RUNNING & ENABLED,  zabbix-
agent = RUNNING & ENABLED,  kubelet = RUNNING & ENABLED, "
}
ok: [mural001-mgt-01.gvs.ggn] => {
"msg": "kafka-server = ||--- NOT RUNNING & DISABLED ---||,
pacemaker = RUNNING & ENABLED,  corosync = RUNNING & ENABLED,  pcsd
= RUNNING & ENABLED,  docker = RUNNING & ENABLED,  collectd-docker
= RUNNING & ENABLED,  zabbix-agent = RUNNING & ENABLED,  kubelet =
RUNNING & ENABLED,  graphite-server = RUNNING & ENABLED,  grafana-
server = RUNNING & ENABLED,  zabbix-server = RUNNING & ENABLED, "
}
ok: [mural001-slv-01.gvs.ggn] => {
"msg": " hadoop-hdfs-datanode = RUNNING & ENABLED,  hadoop-hdfs-
```

```
journalnode = RUNNING & ENABLED,  hadoop-yarn-nodemanager = RUNNING
& ENABLED,  hbase-regionserver = RUNNING & ENABLED, impala-server =
||--- NOT RUNNING & ENABLED ---||,  kafka-server = RUNNING &
ENABLED,  zookeeper-server = RUNNING & ENABLED,  docker = RUNNING &
ENABLED,  collectd-docker = RUNNING & ENABLED,  zabbix-agent =
RUNNING & ENABLED,  kubelet = RUNNING & ENABLED,  etcd = RUNNING &
ENABLED, "
}
ok: [mural001-slv-03.gvs.ggn] => {
"msg": " hadoop-hdfs-datanode = RUNNING & ENABLED,  hadoop-hdfs-
journalnode = RUNNING & ENABLED,  hadoop-yarn-nodemanager = RUNNING
& ENABLED,  hbase-regionserver = RUNNING & ENABLED, impala-server =
||--- NOT RUNNING & ENABLED ---||, kafka-server = ||--- NOT RUNNING
& ENABLED ---||,  zookeeper-server = RUNNING & ENABLED,  docker =
RUNNING & ENABLED,  collectd-docker = RUNNING & ENABLED,  zabbix-
agent = RUNNING & ENABLED,  kubelet = RUNNING & ENABLED,  etcd =
RUNNING & ENABLED, "
}
ok: [mural001-slv-04.gvs.ggn] => {
"msg": " hadoop-hdfs-datanode = RUNNING & ENABLED,  hadoop-hdfs-
journalnode = RUNNING & ENABLED,  hadoop-yarn-nodemanager = RUNNING
& ENABLED,  hbase-regionserver = RUNNING & ENABLED, impala-server =
||--- NOT RUNNING & ENABLED ---||, kafka-server = ||--- NOT RUNNING
& ENABLED ---||, zookeeper-server = ||--- NOT RUNNING & ENABLED ---
||,  docker = RUNNING & ENABLED,  collectd-docker = RUNNING &
ENABLED,  zabbix-agent = RUNNING & ENABLED,  kubelet = RUNNING &
ENABLED,  etcd = RUNNING & ENABLED, "
}
ok: [mural001-mst-01.gvs.ggn] => {
"msg": " hadoop-hdfs-namenode = RUNNING & ENABLED,  hadoop-yarn-
resourcemanager = RUNNING & ENABLED,  hbase-master = RUNNING &
ENABLED, spark-history-server = ||--- NOT RUNNING & ENABLED ---||,
hive-metastore = ||--- NOT RUNNING & ENABLED ---||,  impala-state-
store = RUNNING & ENABLED,  impala-catalog = RUNNING & ENABLED,
```


schema-registry = RUNNING & ENABLED, hadoop-hdfs-zkfc = ||--- NOT RUNNING & ENABLED ---||, pacemaker = RUNNING & ENABLED, corosync = RUNNING & ENABLED, pcsd = RUNNING & ENABLED, docker = RUNNING & ENABLED, sentry-store = ||--- NOT RUNNING & ENABLED ---||, collectd-docker = RUNNING & ENABLED, zabbix-agent = RUNNING & ENABLED, kubelet = RUNNING & ENABLED, "
}
ok: [mural001-mst-02.gvs.ggn] => {
"msg": " hadoop-hdfs-namenode = RUNNING & ENABLED, hadoop-yarn-resourcemanager = RUNNING & ENABLED, hbase-master = RUNNING & ENABLED, spark-history-server = ||--- NOT RUNNING & ENABLED ---||, hive-metastore = ||--- NOT RUNNING & ENABLED ---||, impala-state-store = RUNNING & ENABLED, impala-catalog = RUNNING & ENABLED, schema-registry = RUNNING & ENABLED, hadoop-hdfs-zkfc = ||--- NOT RUNNING & ENABLED ---||, pacemaker = RUNNING & ENABLED, corosync = RUNNING & ENABLED, pcsd = RUNNING & ENABLED, docker = RUNNING & ENABLED, sentry-store = ||--- NOT RUNNING & ENABLED ---||, collectd-docker = RUNNING & ENABLED, zabbix-agent = RUNNING & ENABLED, kubelet = RUNNING & ENABLED, "
}
ok: [mural001-lb-01.gvs.ggn] => {
"msg": " pacemaker = RUNNING & ENABLED, corosync = RUNNING & ENABLED, pcsd = RUNNING & ENABLED, haproxy = RUNNING & ENABLED, docker = RUNNING & ENABLED, collectd-docker = RUNNING & ENABLED, zabbix-agent = RUNNING & ENABLED, kubelet = RUNNING & ENABLED, "
}
ok: [mural001-lb-02.gvs.ggn] => {
"msg": " pacemaker = RUNNING & ENABLED, corosync = RUNNING & ENABLED, pcsd = RUNNING & ENABLED, haproxy = RUNNING & ENABLED, docker = RUNNING & ENABLED, collectd-docker = RUNNING & ENABLED, zabbix-agent = RUNNING & ENABLED, kubelet = RUNNING & ENABLED, "

13. Starting and Stopping the Azkaban Jobs

You can start, stop, restart, or schedule the jobs via Azkaban UI. Perform the following actions to start or stop the jobs.

13.1 Starting Azkaban Job

Perform the following steps to stop the job on Azkaban UI:

1. Enter the *URL: http://192.168.134.4:8507/* in your web browser.
2. Log into the Azkaban UI using the credentials as username: azkaban and password: !4zk4b4n$.
3. Click the job name that you want to start from the list of jobs appearing on the **Projects** screen.
4. Click either the job name or the drop down arrow next to the job name to run the job.
    - If you click the job name, click **Schedule / Execute Flow** button on the top right corner of the screen.
    - If you click the drop-down arrow next to the job name, click the **Run Job** button corresponding to the job that you want to execute to run the job.
5. A dialogue box appears on the screen. Click **Execute** button available on the dialogue box to run the job or **Cancel** button to cancel the action.

    Click **Schedule** button to schedule the job, to run later. Enter the values in the dialogue box as required to schedule the job.
6. Click **Continue** button to confirm the action.

    The job is executed.

13.2 Stopping Azkaban Job

Perform the following steps to stop the job on Azkaban UI:

1. Enter the *URL: http://192.168.134.4:8507/* in your web browser.
2. Log into the Azkaban UI with the provided credentials.
3. Click **Executing** tab available on the screen to view the list of running and finished jobs.
4. Click the Execution Id Number in the **Execution Id** column corresponding to the job that you want to stop.
5. Click **Kill** button on the top right side of the graph sheet to kill the job permanently.

   The job is stopped.

13.3 Enabling and Disabling Azkaban Jobs

Perform the following steps to enable or disable the job on Azkaban UI:

1. Enter the *URL: http://192.168.134.4:8507/* in your web browser.
2. Log into the Azkaban UI with the provided credentials.
3. Click **Scheduling** tab available on the screen to view the list of scheduled jobs on the screen.
4. Click the **Disabled** button corresponding to the scheduled job that you require to stop. This action disables the running job.

   or

   Click **Enable** button corresponding to the stop job that you require to start. This action enables the disabled job.
5. Click **Remove** button in the **Action** column to delete the required scheduled job. This action deletes the scheduled job.

13.4 Scheduling Azkaban Jobs

MURAL allows you to schedule a job in the following two ways:

- Command Prompt. Read "Appendix B: Scheduling Jobs Using Command Prompt" on page 81 to know how to schedule jobs using command prompt.

- User Interface. Read "Appendix C: Scheduling Jobs Using User Interface" on page 83 to know how to schedule jobs using user interface.

14. MURAL Logs Location

MURAL provides you the following two ways for viewing the services' logs location:

- Via UI
- Via Command Prompt

Read the following section to view MURAL Services Logs Location via UI.

1. Enter the *URL: http://192.168.134.4:8507/* in your browser.
2. Click the **History** tab from the four tabs available on the screen.
3. Enter the required job name in the **Search Bar** available on the left top corner of the screen. A list will pop up as per your selection.
4. Select the required job from the list and click **Execution ID** tab.
5. Click **Details** tab to view the job logs and check for exceptions or errors.

Read the following table to know the different MURAL services and their logs locations:

Service Name	Log Location
azkaban process logs	The azkaban process logs are available at `kubectl logs azkaban-<do tab> azkaban-web`
azkaban job logs	The azkaban job logs are available at `/opt/azkaban/solo-server-/logs/` inside the conatiner path `kubectl exec -it azkaban-<kube podname> -- /bin/bash`

Service Name	Log Location

kubernetes container logs	• The kubernetes container logs are available at `cd /var/log/containers/` soflinks for docker container logs available here.
	• Run the following command to know more about specific hostname:
	`kubectl describe node <hostname>`
	Here, <hostnames> refers to the hostname for which you desire to know more.
	• Select the required id and run the following command to check the flannel logs for pod details:
	`/var/log/pods`

Service Name	Log Location

-Inges-tion jobs
 	
-Master jobs
-Aggreg-ation jobs
	• The logs for each job are available at Details icon.
	• The logs are available at the following paths:
	  • `/opt/azkaban/solo-server/logs/`
	  • `/op-t/guavus/-carereflex/-plat-form/default/logs/azkaban/platform/executions/` on `namenode`.
	• Enter *URL: http://<masterNN>:8088/cluster* in your browser and run the following command to know the application logs:
	`applicationId <applicationId>`
	Redirect the output to a file as the preceding command will return o/p on console.
	You can also get the yarn application ID from the following list of Rest APIs:
	• curl -X GET http://<Master NN IP>:8088/ws/v1/cluster/apps
	• curl -X GET http://<Master NN IP>:18082/api/v1/applications

UI/Tom-cat ser-vices logs	The UI/tomcat service logs are available at `cd /var/log/tomcat-mrxui` on namenode. The historical logs are available at `catalina/.<currentData-1>.log` • Check catalina.out (check queries) • Check localhost_access_log.<currentdate>.txt (check access logs)
Cron logs	The cron logs are available at `/var/log/cron`

Service Name	Log Location
Log down-loader logs	The log downloader logs are available at `/var/log/messages`

Hive logs	• Run the following command to find Hive errors:
	`/var/log/hive for metatstore log` (in the file)
	• "hive on spark" source file:
	  • Enter the following URL in browser to check application for yarn logs:
	  `http:<masterNN>:8088/cluster`
	  • Run the following command for yarn logs:
	  `applicationId <applicationId>`

HDFS logs	The HDFS logs are available at `/var/log/hadoop-hdfs` on both the namenode and datanode.
pgsql logs	The pgsql logs are available at `/var/log/postgresql/` Run the following command to check the status on master namenode: `pcs status`
Solution Installer Logs	The solution installer logs are available at `/etc/reflex-provisioner/logs/ansible.log` on management node.
kafka logs	The kafka logs are available at `/var/log/kafka/` on datanodes.
zoo-keeper logs	The zookeeper logs are available at `/var/log/zookeeper/`

Service Name	Log Location

job Xmls and IBs location	- No log file exists
	Run the following command on local or hdfs: (to check the availability of file)
	`ls /opt/sevenflow/conf/`
	`blackList.xml _DONE fileNameAttributeMapping operatorService.xml rulesEngine.xml searchKeyWords.xml service.xml`
	Run the following command to view the list files available in the directory:
	`ls /opt/ddj/conf/`
	`blackList.xml fileNameAttributeMapping ggsn.xml location.xml roamer.xml rulesEngine.xml sgsn.xml`
	Run the following command on IB location to view the list files available n the directory:
	`ls /opt/ddj/IBs/ iabsessionwhitelistfile.csv lte.txt p2pCat.map portProtoAppPath.map snAppCat.map websessionwhitelistfile.csv`

Impala logs	The impala logs are available at `/var/log/impala` on all data nodes.
UM edit-service oauth	The UM editservice oauth logs are available at: • `kubectl logs <oauth pod Id> oauth` • `kubectl logs <editservice pod Id> editservice` • `kubectl logs <fluentd pod Id> fluentd` • `kubectl logs <user-management-frontend pod Id> user-management-frontend` • `kubectl logs <user-management-backend pod Id> user-management-backend`

Service Name	Log Location
Redis logs	The redis logs are available at `/var/log/redis`

For more information about purposes of service logs, refer "Appendix A: MURAL Jobs and Services Log" on page 77

15. Troubleshooting MURAL User Interface

15.1 MURAL UI is not accessible

User interface does not show data on the screen post application of filters.

Solution

Hourly aggregation job must run with the data before launching the MURAL UI to view the data on the screen.

Run the following command to check the availability of data in the hourly_agg job:

1. Log onto the management node as a root user.
2. Run the following command to go to impala shell:

   ```
   impala-shell -i <lb_vip> -u impala
   ```

3. Run the following command to identify the database present on your system:

   ```
   use <db_name>;
   ```

4. Run the following command to select the required database from the available list:

   ```
   show partitions hourly_points;
   ```

5. Run the following command to check for that required bintag and its corresponding size. Ensure that the bintag size is more than 0bytes.

   ```
   [192.168.133.32:21000] > show partitions hourly_points;
   ```

   The output may resemble as follows:

   ```
   | bintag      | #Rows   | #Files | Size      | Bytes Cached |
   Cache Replication | Format   | Incremental stats | Location

                                     |
   | 1533279600 | 90585   | 400    | 15.29MB   | NOT CACHED   | NOT
   ```



```
CACHED        | PARQUET | true              | hdfs://reflex-
platform-
gdsLab/user/hivestore/warehouse/kafkaconnectdb14.db/hourly_
points/bintag=1533279600 |
```

15.2 Duplicating Entries in Hosts File

You require to duplicate host enteries in `/etc/hosts` file on all MURAL nodes. Also, remove external IP host entries from `/etc/hosts` file on all the nodes, if existing.

In case of multiple interface setups, `/etc/hosts` file must contain the internal IP for hostname mapping.

16. Troubleshooting Azkaban User Interface

16.1 Troubleshooting User Interface Access Problems

This topic describes the most common causes of trouble in accessing the MURAL UI from a browser.

Changing the Azkaban UI Password

Perform the following steps to change the password for Azkaban UI:

1. Change the password in the following Azkaban user configuration file:

   ```
   /opt/azkaban/solo-server/conf/azkaban-users.xml
   ```

2. Azkaban container must restart post changing the password.

Perform the following steps to reload the containers.

1. Run the following commands to delete the containers:

   kubectl delete deploy azkaban-platform --force --grace-period=0

2. Run the following commands to create the containers:

   kubectl create -f azkaban-dep.yml

3. Run the following commands to list and view all the pods:

   kubectl get pods

4. Run the following command to view the logs:

   kubectl logs -f azkaban-platform-408617990-zwx6n

5. Run the following command to know the detail of each pod present in the containers:

   kubectl describe pod azkaban-platform-408617990-zwx6n

16.2 Troubleshooting Health Check Failure

Prevent automatic restart of Azkaban Watchdog Solo Server

This topic provides information on how to prevent automatic restart of pgha-azkaban-watchdog solo server when no job is active. To prevent automatic server restart, you need to manually delete the list of schedules from the trigger table.

This could be due to one of the following conditions:

- Azkaban crashes frequently
- Azkaban UI access is not working
- Azkaban pod is up and running

Solution

Run the following commands to restart Azkaban pgha-watchdog solo-server:

1. Log onto the management node as a root user.
2. Run the following command to go to Postgres database to connect azkaban_platform.

   ```
   # psql -U postgres  -d azkaban_platform -h <lb-vip>;
   ```

   A screen will appear, enter the password as **postgres**.
3. Run the following command to list all the triggers:

   ```
   # select * from triggers;
   ```
4. Run the following command to delete the list of triggers:

   ```
   # delete from triggers;
   ```
5. Run the following command to quit:

   ```
   # \q
   ```

   Wait for 15 minutes for azkaban process to come up.
6. Verify Azkaban UI is now accessible.

   To verify the accessibility of UI, log into Azkaban UI and check whether Azkaban schedules are removed.

16.3 Troubleshooting the Different Timezones Error

Azkaban UI displaying inconsistent timezones

The Azkaban UI is showing different timezone on each screen. To change its default timezone, edit the `/opt/azkaban/solo-server-/conf/azkaban.properties` property in Azkaban configuration as required and then restart the Azkaban container.

For example,

Set the default timezone as `default.timezone.id=America/New_York`.

16.4 Troubleshooting Logs Error

Each log on Azkaban UI marked as error

Every log generates ERROR message in stderr, irrespective of it's actual status.

Solution

Azkaban UI is designed to mark log messages as ERROR by default. To confirm the type of logging, check info, warn or error message.

16.5 Executing agg_1month Job via Script

You require to execute agg_1month job on via script due to non functioning of Azkaban UI.

Solution

Run the following commands to run the job from the shell prompt (outside the container):

```
#!/bin/bash
#interval is set for 31 days
INTERVAL=2678400
JOB_CMD="/usr/lib/spark2/bin/spark-submit --master yarn-client --
queue default --name MonthlyAggregationJob --properties-file
/opt/sample_jobs/agg_1month/MonthlyAggregationJobConfig-spark --
verbose --jars /usr/lib/hive/lib/hive-jdbc-
standalone.jar,/opt/tms/java/pcsa/pcsaudf-with-dependencies.jar --
class
com.guavus.reflex.marketing.parquetstore.aggregation.AggregationJob
```



```
/opt/tms/java/aggregations-with-dependencies.jar
MonthlyAggregationJob
/data/streaming/MonthlyAggregationJobConfig.properties"
while ((1)); do
date1=`date +%Y%m01`;
date2=`date -d "$date1 +1 hour"`;
CUR_JOB_TIME=`date -d "$date2" +%s`;
NEW_JOB_TIME=$((CUR_JOB_TIME + ${INTERVAL}))
${JOB_CMD} ${CUR_JOB_TIME}
END_JOB_TIME=$(date +%s)
SLEEP_DUR=$((${NEW_JOB_TIME} - ${END_JOB_TIME}))
if (( ${SLEEP_DUR} > 0 )); then
sleep ${SLEEP_DUR}
fi
done
```

16.6 Troubleshooting the Jobs Failure

There is a possibility that the job fails or gets stuck in the process due to multiple reasons.

The following table explains the reasons of job failure:

Exceptions	Description	Reason
Jobs in which we have jobtime in command are getting failed on Azkaban UI	Flexibin module move ahead and our module was not supporting that due to which this error comes	The wrong module was compiled of flexibin.
DDJ job not writing data in Kafka with Kryo exception.	DDJ module is using urlCategorisation module. The data is not getting processed due to different versions of jars. The user can view an error such as, Encountered unregistered class ID: 68.	Mismatch in the jar compilation which results in failure in running of job.

Exceptions	Description	Reason
A record with missing mandatory field will be dropped at ingestion.	Ingestion API defines mandatory / optional fields to make sure records with no good information should be dropped.	The MSISDN must be absent or any of these are missing or cannot be determined and the record dropped.

Solution: Contact Cisco Support, if encountering such exceptions.

16.7 Replacing the Banner

You require to change the Azkaban UI banner on your system.

To change the existing UI banner with the CISCO banner, run the following commands:

```
Mst -1
kubectl edit cm azkaban-platform
++++++++++++++++++++++++++++++++++++
#Azkaban Personalization Settings
azkaban.name=Guavus
azkaban.label=platform
azkaban.color= #FF0000
++++++++++++++++++++++++++++++++++++
ssh mural001-mgt-01
cd
/opt/guavus/carereflex/platform/default/deployment/azkaban/platform
/Guavus Banner shouldn’t be present in Azkaban UI of Cisco solution
This is just to pick the correct name to redploy
kubectl delete deploy azkaban-platform --force --grace-period=0
kubectl create -f azkaban-dep.yml
kubectl get pods
kubectl logs -f azkaban-platform-408617990-zwx6n
kubectl describe pod azkaban-platform-408617990-zwx6n
```

17. Troubleshooting Postgresgl Node

This section explains the steps to troubleshoot the situation when postgres node is down. Read the following section to gain more understanding about its solution.

Run the following commands on the nodes to up the Postgres:

```
###
## /var/lib/pgsql/<version>/data.recovery
rm -rf /var/lib/pgsql/9.4/data.recovery
mv /var/lib/pgsql/9.4/data /var/lib/pgsql/9.4/data.recovery
su - postgres
pg_basebackup -h <VIP> -U postgres -D c -X stream -P
rm -rf /var/lib/pgsql/tmp/PGSQL.lock
cp /var/lib/pgsql/9.4/data.recovery/recovery.conf
/var/lib/pgsql/9.4/data/recovery.conf
chown postgres:postgres /var/lib/pgsql/9.4/data/recovery.conf
chmod 0600 /var/lib/pgsql/9.4/data/recovery.conf

### (most of the times, just clearing the failcount works, so first
check this) ###
pcs status
pcs resource clear pgsql <hostname_with_fqdn>
pcs resource failcount reset pgsql
pcs resource clear pgsql <hostname_with_fqdn>
pcs resource cleanup pgsql
```

This section explains steps to recover the Postgresgl node on your system. Read the following section to gain more understanding about its solution.

Note: These steps are applicable only when master namenode is running on the pgsql cluster with maximum two dead nodes out of three.

1. Run the following command to check the status of postgresgl nodes:

   ```
   crm_mon -Afr1
   ```


The output may resemble as follows:

```
- Node mgmtnode.devopslabs.gurugram.in:
  + master-pgsql : 1000
  + pgsql-data-status : LATEST
  + pgsql-master-baseline : 000000000F000090
  + pgsql-receiver-status : normal (master)
  + pgsql-status : PRI
  + pgsql-xlog-loc : 00000000130113E8

- Node namenode1.devopslabs.gurugram.in:
  + master-pgsql : 100
  + pgsql-data-status : STREAMING|SYNC
  + pgsql-receiver-status : normal
  + pgsql-status : HS:sync
  + pgsql-xlog-loc : 0000000013011218

- Node namenode2.devopslabs.gurugram.in:
  + master-pgsql : -INFINITY
  + pgsql-data-status : DISCONNECT
  + pgsql-receiver-status : normal
  + pgsql-status : STOP
```

2. Log onto the node which is down. Here, as an example, namenode2.devopslabs.gurugram.in is down.
3. Run the following command to remove `PGSQL.lock` file from the logged in node:

   `rm /var/lib/pgsql/tmp/PGSQL.lock`

4. Run the following command to set resource-stickiness as infinity:

   `pcs resource defaults resource-stickiness=INFINITY`

5. Run the following command to reset fail count to view the empty file:


```
pcs resource failcount reset pgsql
```

6. Wait for few minutes and run the following command to verify the status of the node:

```
crm_mon -Afr1
```

The output must resemble as follows:

```
- Node mgmtnode.devopslabs.gurugram.in:
  + master-pgsql : 1000
  + pgsql-data-status : LATEST
  + pgsql-master-baseline : 000000000F000090
  + pgsql-receiver-status : normal (master)
  + pgsql-status : PRI
  + pgsql-xlog-loc : 00000000130113E8

- Node namenode1.devopslabs.gurugram.in:
  + master-pgsql : 100
  + pgsql-data-status : STREAMING|SYNC
  + pgsql-receiver-status : normal
  + pgsql-status : HS:sync
  + pgsql-xlog-loc : 0000000013015440

- Node namenode2.devopslabs.gurugram.in:
  + master-pgsql : 10
  + pgsql-data-status : STREAMING|POTENTIAL
  + pgsql-receiver-status : normal
  + pgsql-status : HS:potential
  + pgsql-xlog-loc
```

Perform the following steps in case, the preceding output tends to differ.

1. Run the following command to take backup of data directory on down node:

```
mv /var/lib/pgsql/9.4/data /var/lib/pgsql/9.4/data_bkp
```

2. Run the following command to take backup from pgsql node:


```
su - postgres
pg_basebackup -h <loadbalancer VIP> -U postgres -D
/var/lib/pgsql/9.4/data -X stream -P
exit
```

3. Run the following command to copy `recovery.conf` file from backup data directory to data directory:

```
cp /var/lib/pgsql/9.4/data_bkp/recovery.conf
/var/lib/pgsql/9.4/data/
```

4. Run the following command to clean config cache:

```
pcs resource cleanup pgsql
```

5. Run the following command to reset pgsql failcount:

```
pcs resource failcount reset pgsql
```

6. Wait for few minutes and run the following command to check the status of postgresgl node:

```
crm_mon -Afr -1
```

The output may resemble as follows:

```
- Node mgmtnode.devopslabs.gurugram.in:
  + master-pgsql : 1000
  + pgsql-data-status : LATEST
  + pgsql-master-baseline : 000000000F000090
  + pgsql-receiver-status : normal (master)
  + pgsql-status : PRI
  + pgsql-xlog-loc : 00000000130113E8

- Node namenode1.devopslabs.gurugram.in:
  + master-pgsql : 100
  + pgsql-data-status : STREAMING|SYNC
  + pgsql-receiver-status : normal
```



```
+ pgsql-status : HS:sync
+ pgsql-xlog-loc : 0000000013015440
```

- Node namenode2.devopslabs.gurugram.in:

```
+ master-pgsql : 10
+ pgsql-data-status : STREAMING|POTENTIAL
+ pgsql-receiver-status : normal
+ pgsql-status : HS:potential
+ pgsql-xlog-loc
```

The node is now successfully up and running.

18. Troubleshooting a Flannel

This section explains steps to troubleshoot a flannel configuration. At its base, the flannel pods (as you install them) register everything with the etcd server running within the platform. Read the following section to gain more understanding about its solution.

1. Run the following command to reset the etcd configuration:

```
ansible-playbook -i inventory/generated/dev/devops022/hosts -u
root --ask-pass ../adhoc-playbooks/net_list.yml
```

The output may resembles as follows:

```
...
TASK [Rebuild flannel configuration from server facts]
************************
ok: [devops022-slv-03.devops.guavus.mtl] => {"msg": "etcdctl
set /srx.local/network/subnets/10.233.119.0-24 '
{\"PublicIP\":\"10.71.124.81\",\"BackendType\":\"vxlan\",\"Bac
kendData\":{\"VtepMAC\":\"8e:ad:03:29:08:e5\"}}' --ttl 0"}
ok: [devops022-slv-01.devops.guavus.mtl] => {"msg": "etcdctl
set /srx.local/network/subnets/10.233.73.0-24 '{
\"PublicIP\":\"10.71.124.79\",\"BackendType\":\"vxlan\",\"Back
endData\":{\"VtepMAC\":\"2a:8c:44:64:18:8d\"}}' --ttl 0"}
ok: [devops022-slv-02.devops.guavus.mtl] => {"msg": "etcdctl
set /srx.local/network/subnets/10.233.116.0-24 '{
\"PublicIP\":\"10.71.124.80\",\"BackendType\":\"vxlan\",\"Back
endData\":{\"VtepMAC\":\"26:4a:85:e2:b1:27\"}}' --ttl 0"}
ok: [devops022-mst-01.devops.guavus.mtl] => {"msg": "etcdctl
set /srx.local/network/subnets/10.233.66.0-24 '{
\"PublicIP\":\"10.71.124.77\",\"BackendType\":\"vxlan\",\"Back
endData\":{\"VtepMAC\":\"a6:a1:83:ba:3e:32\"}}' --ttl 0"}
ok: [devops022-mgt-01.devops.guavus.mtl] => {"msg": "etcdctl
set /srx.local/network/subnets/10.233.106.0-24 '{
```



```
\"PublicIP\":\"10.71.124.76\",\"BackendType\":\"vxlan\",\"Back
endData\":{\"VtepMAC\":\"3e:c4:84:d7:c9:38\"}}' --ttl 0"}
ok: [devops022-mst-02.devops.guavus.mtl] => {"msg": "etcdctl
set /srx.local/network/subnets/10.233.67.0-24 '{
\"PublicIP\":\"10.71.124.78\",\"BackendType\":\"vxlan\",\"Back
endData\":{\"VtepMAC\":\"7e:ce:da:22:77:0f\"}}' --ttl 0"}
ok: [devops022-lb-01.devops.guavus.mtl] => {"msg": "etcdctl
set /srx.local/network/subnets/10.233.81.0-24 '{
\"PublicIP\":\"10.71.124.84\",\"BackendType\":\"vxlan\",\"Back
endData\":{\"VtepMAC\":\"36:5a:9a:7f:fa:44\"}}' --ttl 0"}
ok: [devops022-lb-02.devops.guavus.mtl] => {"msg": "etcdctl
set /srx.local/network/subnets/10.233.79.0-24 '{
\"PublicIP\":\"10.71.124.85\",\"BackendType\":\"vxlan\",\"Back
endData\":{\"VtepMAC\":\"56:21:c9:d7:b7:05\"}}' --ttl 0"}
ok: [devops022-app-01.devops.guavus.mtl] => {"msg": "etcdctl
set /srx.local/network/subnets/10.233.125.0-24 '{
\"PublicIP\":\"10.71.124.82\",\"BackendType\":\"vxlan\",\"Back
endData\":{\"VtepMAC\":\"ce:77:b0:91:fd:29\"}}' --ttl
0"}
ok: [devops022-app-02.devops.guavus.mtl] => {"msg": "etcdctl
set /srx.local/network/subnets/10.233.78.0-24 '{
\"PublicIP\":\"10.71.124.83\",\"BackendType\":\"vxlan\",\"Back
endData\":{\"VtepMAC\":\"f6:a1:9e:ab:c9:42\"}}' --ttl
0"}
...
```

2. Run the following command to view and match the current configuration with the preceding expected configuration:

```
SUBNETS=`etcdctl --endpoint=https://127.0.0.1:2379 ls /reflex-
platform.local/network/subnets/` ; for x in $SUBNETS ; do echo
$x; etcdctl --endpoint=https://127.0.0.1:2379 get $x ; done
```

The output may resemble as follows:


```
/reflex-platform.local/network/subnets/10.233.66.0-24
{"PublicIP":"10.71.124.77","BackendType":"vxlan","BackendDat
a":{"VtepMAC":"a6:a1:83:ba:3e:32"}}/reflex-
platform.local/network/subnets/10.233.79.0-24
{"PublicIP":"10.71.124.85","BackendType":"vxlan","BackendDat
a":{"VtepMAC":"56:21:c9:d7:b7:05"}}
/reflex-platform.local/network/subnets/10.233.106.0-24
{"PublicIP":"10.71.124.76","BackendType":"vxlan","BackendDat
a":{"VtepMAC":"3e:c4:84:d7:c9:38"}}/reflex-
platform.local/network/subnets/10.233.125.0-24
{"PublicIP":"10.71.124.82","BackendType":"vxlan","BackendDat
a":{"VtepMAC":"ce:77:b0:91:fd:29"}}
/reflex-platform.local/network/subnets/10.233.78.0-24
{"PublicIP":"10.71.124.83","BackendType":"vxlan","BackendDat
a":{"VtepMAC":"f6:a1:9e:ab:c9:42"}}
/reflex-platform.local/network/subnets/10.233.116.0-24
{"PublicIP":"10.71.124.80","BackendType":"vxlan","BackendDat
a":{"VtepMAC":"26:4a:85:e2:b1:27"}}
/reflex-platform.local/network/subnets/10.233.81.0-24
{"PublicIP":"10.71.124.84","BackendType":"vxlan","BackendDat
a":{"VtepMAC":"36:5a:9a:7f:fa:44"}}
/reflex-platform.local/network/subnets/10.233.67.0-24
{"PublicIP":"10.71.124.78","BackendType":"vxlan","BackendDat
a":{"VtepMAC":"7e:ce:da:22:77:0f"}}
/reflex-platform.local/network/subnets/10.233.73.0-24
{"PublicIP":"10.71.124.79","BackendType":"vxlan","BackendDat
a":{"VtepMAC":"2a:8c:44:64:18:8d"}}
/reflex-platform.local/network/subnets/10.233.119.0-24
{"PublicIP":"10.71.124.81","BackendType":"vxlan","BackendDat
a":{"VtepMAC":"8e:ad:03:29:08:e5"}}
```


19. Common Troubleshooting

19.1 Unprocessed Files Remains in the Input Folder

The uncompressed files might not process and remains in input folder. This happens due to unavailability of gzip files in the system.

Solution

It is mandatory to have gzip compression files on south bound integration for MuralNg. It improves the system performance and the files will not remain in the input folder.

19.2 Viewing "Unknown" in Mobile Category

“Unknown” message might appear in mobile category from URLcat.

Solution

You need to check TAC database and imsi/msisdn fields.

20. Appendix A: MURAL Jobs and Services Log

Refer the following table to know about the input and output path of different Azkaban jobs:

Job Name	Input Data Path	Output Data
master_http	Talend Kafka Topic (talend-viaspark_http Output)	1. DDJ Kafka Topic 2. UnCat Output on HDFS
master_nonhttp	DDJ Kafka Topic	1. DME Kafka Topic 2. Hive 3. Granular Feed output on HDFS
agg_5min	Hive	Hive
talendviaspark_http	/user/mrx/ingestion/input (HDFS)	1. Processed file at HDFS: /user-/mrx/ingestion/ready/ 2. Talend Kafka Topic
talendviaspark_nonhttp	/user/mrx/ingestion2/input (HDFS)	1. Processed file at HDFS: /user-/mrx/ingestion2/ready/ 2. Talend Kafka Topic
HourlyAgg_http	Output of uncatHeavyhitter_http	HDFS
DailyAgg_http	Output of uncatHourlyAgg_http	HDFS
WeeklyAgg_http	Output of uncatDailyAgg_http	HDFS
DataQualityODSGDSDailyAgg_http	/data/streaming/outputPath/dataqualityodsgds/hourlyagg/	/data/streaming/outputPath/dataqualityodsgds/dailyagg/
DataQualityODSGDSDailyAgg_nonhttp	/data/streaming2/outputPath/dataqualityodsgds/hourlyagg/	/data/streaming2/outputPath/dataqualityodsgds/dailyagg/

Job Name	Input Data Path	Output Data
DataQu-alityODSGDSHo-urlyAgg_http	uncat_inputpath /data/stream-ing/out-putPath/dataqual-ityodsgds/dmeoutput/	uncat_outputpath /data/stream-ing/out-putPath/dataqual-ityodsgds/hourlyagg/
DataQu-alityODSGDSHo-urlyAgg_nonhttp	/data/stream-ing2/out-putPath/dataqual-ityodsgds/dmeoutput/	/data/stream-ing2/out-putPath/dataqual-ityodsgds/hourlyagg/
DataQu-alityODSGDSWe-eklyAgg_http	/data/stream-ing/out-putPath/dataqual-ityodsgds/dailyagg/	/data/stream-ing/out-putPath/dataqual-ityodsgds/weeklyagg/
DataQu-alityODSGDSWe-eklyAgg_nonhttp	/data/stream-ing2/out-putPath/dataqual-ityodsgds/dailyagg/	/data/stream-ing2/out-putPath/dataqual-ityodsgds/weeklyagg/
uncatDailyAgg_http	uncat_inputpath /data/d-dj/out-putPath/hourlyAg-gregateUncat/	/data/d-dj/out-putPath/dailyAggregateUncat/
uncatDailyAgg_nonhttp	/data/d-dj2/out-putPath/hourlyAg-gregateUncat/	/data/d-dj2/out-putPath/dailyAggregateUncat/
uncatHeavy-hitter_http	/data/d-dj/un-knownOut-putPath/heavyHitterPath	/data/d-dj/out-putPath/heavyHitterUncat/
uncatHeavy-hitter_nonhttp	/data/d-dj2/un-knownOut-putPath/heavyHitterPath	/data/d-dj2/out-putPath/heavyHitterUncat/

Job Name	Input Data Path	Output Data
uncatHourlyAg-g_http	/data/d-dj/out-putPath/heavyHitterUncat/	/data/d-dj/out-putPath/hourlyAg-gregateUncat/
uncatHourlyAg-g_nonhttp	/data/d-dj2/out-putPath/heavyHitterUncat/	/data/d-dj2/out-putPath/hourlyAg-gregateUncat/
uncatWeeklyAg-g_http	/data/d-dj/out-putPath/dailyAggregateUncat/	/data/d-dj/out-putPath/weeklyAg-gregateUncat/
uncatWeeklyAg-g_nonhttp	/data/d-dj2/out-putPath/dailyAggregateUncat/	/data/d-dj2/out-putPath/weeklyAg-gregateUncat/

Refer the following table to know about the purposes of different services log:

Service Name	Purpose
kafka-connect	All the kafka-connect logs appears
azkaban pro-cess logs	To show pgha logs for azkaban process
azkaban job logs	• These logs contain job Id execution logs • To web access logs for azkaban • To access logs for azkaban http process
kubernetes con-tainer logs	• To know if any node evict from kube cluster or kubelet ser-vice is down • Run the following command to check evicted nodes: `kubectl get pods -o wide -n kube-system|grep "kube-proxy"`

Service Name	Purpose
• Ingestion jobs • Master jobs • Aggreg-ation jobs	To debug jobs, if any failure occurred
NLE streaming job	• To check NLE counters • To check NLE job errors and exceptions
UI / Tomcat services logs	• To check queries to check access logs • To check errors/exceptions
SPE/tomcat ser-vice logs	• To check queries • To check access logs • To check errors/exceptions
Cron logs	All cron execution logs appears in cron logs
Log down-loader logs	All logdownloader logs appears in /var/log/messages
Hive logs	To find hive queries , hive configurations and hive running state here
HDFS logs	To find all the debug logs related to HDFS in this location
pgsql logs	-
Solution Installer Logs	For solution installer processing and failed playbook steps
Kafka logs	To find kafka status and configuration errors from logs
Zookeeper logs	To find zookeeper errors here
Job Xmls and IBs location	Xmls and IBs location for DDJ and DME workflow
Impala logs	To debug Impala logs
UM editservice oauth	-
Redis logs	-

21. Appendix B: Scheduling Jobs Using Command Prompt

Schedule the jobs in the same sequence as mentioned in the following section:

1. **master_http**

   Perform the following steps to run master_http job:

   1. Log into as a root user on one of the namenodes and run the following command:

      ```
      cd /root/streamingJobs/ddj_http
      nohup ./run_ddj_httpJob.sh &
      ```

   2. Run the following command to view the job progress:

      ```
      tail -f nohup.out
      ```

   3. Before running the next job, ensure the following mentioned counters are present in the `nohup.out` log file:

      ```
      Some(totalRecords) - 0
      Some(invalidRecords) - 0
      Some(unknownRecords) - 0
      ```

2. **master_nonhttp**

   Perform the following steps to run master_nonhttp job:

   1. Log into as a root user on one of the name node and run the following command:

      ```
      cd /root/streamingJobs/ddj_nonhttp
      nohup ./run_ddj_nonhttpJob.sh &
      ```

   2. Run the following command to view the job progress:

      ```
      tail -f nohup.out
      ```

   3. Before running the next job, ensure the following mentioned counters

are present in the `nohup.out` log file:

```
Some(totalRecords) - 0
Some(invalidRecords) - 0
Some(unknownRecords) - 0
```

3. **talendviaspark_http**: Schedule http ingestion job in every 1 minute.
4. **talendviaspark_nonhttp**: Schedule nonhttp ingestion job in every 1 minute.
5. **agg-5min job**: Schedule http aggregation job in every 5 minutes.
6. **agg_1hour job**: Schedule http aggregation job every hour.
7. **agg_1day job**: Schedule daily aggregation job at 06:00 hour of the day.
8. **agg_1month job**: Schedule monthly aggregation job at 1st day at 23:00 hour, every month.

   After the successful completion of monthly agg job, run the following command to remove the `/data/streaming/monthly-aggregation-ts` file from HDFS

   ```
   ssh namenode
   hdfs dfs -rm /data/streaming/monthly-aggregation-ts
   ```

Notes:

- Due to kubernetes container issue, launch the jobs outside the container on Master Namenode.
- Do not schedule the jobs from Azkaban UI until any notification from MURAL Support.
- HA is not be available for the jobs outside the container.
- Launch the jobs in same the way as they will be running on production site.

22. Appendix C: Scheduling Jobs Using User Interface

Ensure you schedule the jobs as explained in this topic:

22.1 Logging into Azkaban

1. Enter *URL: http://<VIP>:8507/haproxy_stats* in your web browser.
2. Log into the Azkaban UI using your credentials as provided.

22.2 Scheduling Jobs

Read the following section to know how to schedule jobs:

1. Enter *URL: http://lib-vip:8507/* in your web browser.
2. Enter the provided credentials to log into the Azkaban UI.
3. To schedule a job, perform the following steps. This step uses Data Domain Job as an example.
    1. Click **MRX_5_6_2_rc1** project.
    2. Click master job's drop-down arrow < master http.
    3. Click **Run Job** to run the job.
4. Similarly, schedule the following jobs in the listed order and the properties file for the jobs are configured in the dedicated folder.
    1. Talend http Ingestion Job
    2. Talend non http Ingestion Job
    3. Master http Job
    4. Master non http Job
    5. 5min Aggregation Job
    6. Hourly Aggregation Job
    7. Daily Aggregation Job

8. Dimension Impala To Postgress
9. Monthly Aggregation Job

   After the job is scheduled, the epoch time is passed to the Daily aggregation job, automatically.

10. HeavyHitter Uncat Job

    After the job is scheduled, DDJ creates a folder in hdfs (Hadoop Distributed File System) `/data/d-dj/unknownOutputPath/heavyHitterPath/yyyy/mm/dd/hh/mm`. Pass the epoch value of yyyymmddhhmm to HeavyHitter Uncat job.

11. HourlyAgg Uncat Job

    After the job is scheduled, HourlyAgg Uncat creates a folder in hdfs `/data/d-dj/outputPath/hourlyAggregateUncat/yyyy/mm/dd/hh/mm`.

12. DailyAgg Uncat Job

    After the job is scheduled, HourlyAgg Uncat creates a folder in hdfs `/data/d-dj/outputPath/hourlyAggregateUncat/yyyy/mm/dd/hh/mm`.

13. WeeklyAgg Uncat Job

    After the job is scheduled, DailyAgg Uncat creates a folder in hdfs `/data/d-dj/outputPath/dailyAggregateUncat/yyyy/mm/dd/hh/mm`.

14. Cleanup Job

    The primary task of cleanup job is to clear all the output of all scheduled jobs as per defined configurations in cleanup `config.xml`.

23. Appendix D: Scheduling Jobs via Script

You can schedule jobs via script as mentioned in this section:

Run the following commands as stated for each job in the following section:

1. **Talend**

```
#!/bin/bash
INTERVAL=60
while ((1)); do
CUR_JOB_TIME=$(date +%s)
#talend job expects job time aligned to 1-min boundary
JOB_TIME=$((${CUR_JOB_TIME} / 60) * 60))
NEW_JOB_TIME=$((CUR_JOB_TIME + ${INTERVAL}))
/usr/lib/spark2/bin/spark-submit --master yarn-client --class
com.guavus.run.LaunchTalend
/opt/tms/java/azkaTalend/SparkTalendJob-1.0-SNAPSHOT-jar-with-
dependencies.jar "/opt/tms/java/azkaTalend/mrx_extract_
Talend/mrx_extract_run_parallel/mrx_extract_run_parallel_
run.sh --context_param context_dir=/opt/mrx/ingestion/etc/  --
context_param spark_file_array="
"/opt/mrx/ingestion/etc/extract.conf"
/opt/tms/java/azkaTalend/spark.properties ${JOB_TIME}
END_JOB_TIME=$(date +%s)
SLEEP_DUR=$((${NEW_JOB_TIME} - ${END_JOB_TIME}))
if (( ${SLEEP_DUR} > 0 )); then
sleep ${SLEEP_DUR}
fi
done
```

2. **5min_agg job**

```
#!/bin/bash
INTERVAL=300
JOB_CMD="/usr/lib/spark2/bin/spark-submit --master yarn-client
```

```
--queue jobs.5min --properties-file /opt/sample_jobs/agg_
5min/PdmAggregationJobConfig-spark --class
com.guavus.reflex.marketing.parquetstore.hourlyjob.PDMAggregat
eJob /opt/tms/java/aggregations-with-dependencies.jar
5minAggregateJob /data/streaming/pdm-aggregation.config "
while ((1)); do
CUR_JOB_TIME=$(date +%s)
# 5-min aggregation job expects job time aligned to 5-min
boundary
JOB_TIME=$((${CUR_JOB_TIME} / 300) * 300))
NEW_JOB_TIME=$((CUR_JOB_TIME + ${INTERVAL}))
${JOB_CMD} ${JOB_TIME}
END_JOB_TIME=$(date +%s)
SLEEP_DUR=$((${NEW_JOB_TIME} - ${END_JOB_TIME}))
if (( ${SLEEP_DUR} > 0 )); then
sleep ${SLEEP_DUR}
fi
done
```

3. **hourly_agg job**

```
#!/bin/bash
INTERVAL=3600
JOB_CMD="/usr/lib/spark2/bin/spark-submit --master yarn-client
--queue jobs.hourly --name HourlyAggregationJob --properties-
file /opt/sample_jobs/agg_1hour/HourlyAggregationJobConfig-
spark --verbose --jars /usr/lib/hive/lib/hive-jdbc-
standalone.jar,/opt/tms/java/pcsa/pcsaudf-with-
dependencies.jar --class
com.guavus.reflex.marketing.parquetstore.aggregation.Aggregati
onJob /opt/tms/java/aggregations-with-dependencies.jar
HourlyAggregationJob
/data/streaming/HourlyAggregationJobConfig.properties"
while ((1)); do
CUR_JOB_TIME=$(date +%s)
```



```
# 1-hour aggregation job expects job time aligned to hour
boundary
JOB_TIME=$((((${CUR_JOB_TIME} / 3600) * 3600)+600))
NEW_JOB_TIME=$((CUR_JOB_TIME + ${INTERVAL}))
${JOB_CMD} ${JOB_TIME}
END_JOB_TIME=$(date +%s)
SLEEP_DUR=$((${NEW_JOB_TIME} - ${END_JOB_TIME}))
if (( ${SLEEP_DUR} > 0 )); then
sleep ${SLEEP_DUR}
fi
done
```

4. **daily_agg job**

```
#!/bin/bash
INTERVAL=86400
JOB_CMD="/usr/lib/spark2/bin/spark-submit --master yarn-client
--queue jobs.daily --name DailyAggregationJob --properties-
file /opt/sample_jobs/agg_1day/DailyAggregationJobConfig-spark
--verbose --jars /usr/lib/hive/lib/hive-jdbc-
standalone.jar,/opt/tms/java/pcsa/pcsaudf-with-
dependencies.jar --class
com.guavus.reflex.marketing.parquetstore.aggregation.Aggregati
onJob /opt/tms/java/aggregations-with-dependencies.jar
DailyAggregationJob
/data/streaming/DailyAggregationJobConfig.properties"
JOB_CMD2="/usr/lib/spark2/bin/spark-submit --master yarn-
client --queue jobs.hourly --name ImpalaToPostgresJob --
properties-file /opt/sample_
jobs/dimensionImpalaToPostgres/sparkConfigs.txt --driver-
class-path /opt/tms/java/aggregations-with-dependencies.jar --
verbose --class
com.guavus.reflex.marketing.parquetstore.aggregation.ImpalaToP
ostgres_DimensionTables /opt/tms/java/aggregations-with-
```



```
dependencies.jar /opt/sample_
jobs/dimensionImpalaToPostgres/ImpalaToPostgres_DB.properties
/opt/sample_
jobs/dimensionImpalaToPostgres/MappingsPostgresToImpalaColumn.
conf"
while ((1)); do
CUR_JOB_TIME=$(date +%s)
# 1-day aggregation job expects job time aligned to hour
boundary
JOB_TIME=$((((${CUR_JOB_TIME} / 86400) * 86400)+3600))
NEW_JOB_TIME=$((CUR_JOB_TIME + ${INTERVAL}))
${JOB_CMD} ${JOB_TIME}
sleep 120
${JOB_CMD2} ${JOB_TIME} Daily
END_JOB_TIME=$(date +%s)
SLEEP_DUR=$((${NEW_JOB_TIME} - ${END_JOB_TIME}))
if (( ${SLEEP_DUR} > 0 )); then
sleep ${SLEEP_DUR}
fi
done
```

5. **monthly_agg job**

```
#!/bin/bash
#interval is set for 31 days
INTERVAL=2678400
JOB_CMD="/usr/lib/spark2/bin/spark-submit --master yarn-client
--queue default --name MonthlyAggregationJob --properties-file
/opt/sample_jobs/agg_1month/MonthlyAggregationJobConfig-spark
--verbose --jars /usr/lib/hive/lib/hive-jdbc-
standalone.jar,/opt/tms/java/pcsa/pcsaudf-with-
dependencies.jar --class
com.guavus.reflex.marketing.parquetstore.aggregation.Aggregati
onJob /opt/tms/java/aggregations-with-dependencies.jar
```



```
MonthlyAggregationJob
/data/streaming/MonthlyAggregationJobConfig.properties"
while ((1)); do
date1=`date +%Y%m01`;
date2=`date -d "$date1 +1 hour"`;
CUR_JOB_TIME=`date -d "$date2" +%s`;
NEW_JOB_TIME=$((CUR_JOB_TIME + ${INTERVAL}))
${JOB_CMD} ${CUR_JOB_TIME}
END_JOB_TIME=$(date +%s)
SLEEP_DUR=$((${NEW_JOB_TIME} - ${END_JOB_TIME}))
if (( ${SLEEP_DUR} > 0 )); then
sleep ${SLEEP_DUR}
fi
done
```

24. Appendix E: List of Services

Service	Service Name	Service Manager	Nodes on which service is running
Zookeeper Server	zookeeper-server.service	systemd	datanode1, datanode2, datanode3
ZKFC	hadoop-hdfs-zkfc.service	systemd	namenode1, namenode2
Spark History Server	spark-history-server.service	systemd	namenode1, namenode2
Sentry Store	sentry-store.service	systemd	namenode1, namenode2
Schema Registry	schema-registry.service	systemd	namenode1, namenode2
Rsyslog	rsyslog.service	systemd	Rsyslog rsyslog.service systemd all nodes
PCSD	pcsd.service	systemd	management, namenode1, namenode2, loadbalancer1, loadbalancer2
Pacemaker	pacemaker.service	systemd	management, namenode1, namenode2, loadbalancer1, loadbalancer2
NTP	ntpd.service	systemd	all nodes
Kafka Server	kafka-server.service	systemd	datanode1, datanode2, datanode3
Impala State Store	impala-state-store.service	systemd	namenode1,namenode2
Impala Server	impala-server.service	systemd	datanode1, datanode2, datanode3
Impala Catalog	impala-catalog.service	systemd	namenode1,namenode2
HTTP Server	httpd.service	systemd	management
Hive Metastore	hive-metastore.service	systemd	namenode1,namenode2

Service	Service Name	Service Manager	Nodes on which service is running
Hbase Region Server	hbase-region-server.service	systemd	datanode1, datanode2, datanode3
Hbase Master	hbase-master.service	systemd	namenode1, namenode2
Haproxy	haproxy.service	systemd	loadbalancer1, loadbalancer2
Hadoop Resource-manager	hadoop-yarn-resource-manager.service	systemd	namenode1, namenode2
Hadoop Node Manager	hadoop-yarn-node-manager.service systemd all data nodes	systemd	all data nodes
Hadoop Name Node	hadoop-hdfs-namenode.service	systemd	namenode1, namenode2
Hadoop Journal Node	Hadoop Journal Node hadoop-hdfs-journalnode.service	systemd	datanode1, datanode2, datanode3
Hadoop Data Node	hadoop-hdfs-datanode.service	systemd	all data nodes
Corosync	corosync.service	systemd	namenode1, namenode2, loadbalancer1, loadbalancer2