Hortonworks Data Platform

Falcon Quick Start Guide

(Apr 13, 2015)

docs.hortonworks.com

Hortonworks Data Platform : Falcon Quick Start Guide

The Hortonworks Data Platform, powered by Apache Hadoop, is a massively scalable and 100% open source platform for storing, processing and analyzing large volumes of data. It is designed to deal with data from many sources and formats in a very quick, easy and cost-effective manner. The Hortonworks Data Platform consists of the essential set of Apache Hadoop projects including MapReduce, Hadoop Distributed File System (HDFS), HCatalog, Pig, Hive, HBase, Zookeeper and Ambari. Hortonworks is the major contributor of code and patches to many of these projects. These projects have been integrated and tested as part of the Hortonworks Data Platform release process and installation and configuration tools have also been included.

Unlike other providers of platforms built using Apache Hadoop, Hortonworks contributes 100% of our code back to the Apache Software Foundation. The Hortonworks Data Platform is Apache-licensed and completely open source. We sell only expert technical support, training and partner-enablement services. All of our technology is, and will remain free and open source.

Please visit the Hortonworks Data Platform page for more information on Hortonworks technology. For more information on Hortonworks services, please visit either the Support or Training page. Feel free to Contact Us directly to discuss your specific needs.

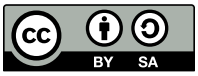

Table of Contents

1. Data Replication with Falcon Quick Start Guide

The Data Replication Quick Start Guide gets you started replicating your data with Falcon by addressing the following tasks:

- Creating Source Clusters
- Creating Target Clusters
- Defining Datasets for Replication
- Submitting Datasets for Replication

1.1. Prerequisites

Before you begin setting up Data Replication, this guide assumes you have the following installed on your cluster:

- **HDP.** Installed on your cluster (using Ambari or a Manual Install)
- **Falcon.** Installed on your cluster and the Falcon Service is running.
- **Oozie Client and Server.** Installed on your cluster and the Oozie Service is running on your cluster.

1.2. Define the Data Source: Set Up a Source Cluster Entity

Define where data and processes are stored in the cluster entity.

1. Create an XML file for the Cluster entity. This file contains all properties for the cluster. Include the XML version:

```
<?xml version="1.0"?>
```

2. Define the `colo` and `name` attributes for the cluster.

```
<?xml version="1.0"?>
<cluster colo="<MyDataCenter>" description="description"
            name="<MyDataCenterFilename>">
</cluster>
```

Note

`colo` is the name of the data center.

`name` is the file name of the data center.

3. Define the interfaces for the cluster. For each interface specify type of interface, endpoint, and Apache version.

   For example:

```
<cluster colo="<MyDataCenter>" description="description"
          name="<MyDataCenterFilename>">
     <interfaces>

          <!-- Required for distcp for replications. -->
          <interface type="readonly" endpoint="hftp://nn:50070" version="2.
4.0" />

          <!-- Needed for writing to HDFS-->
          <interface type="write" endpoint="hdfs://nn:8020" version="2.4.
0" />

          <!-- Needed to write to jobs as MapReduce-->
          <interface type="execute" endpoint="rm:8050" version="2.4.0" />

          <!-- Required. Submits Oozie jobs.-->
          <interface type="workflow" endpoint="http://os:11000/oozie/"
 version="4.0.0" />

          <!--Register/deregister partitions in the Hive Metastore and get
 events on partition availability-->
          <interface type="registry" endpoint="thrift://hms:9083" version=
"0.13.0" />

          <!--Needed for alerts-->
          <interface type="messaging" endpoint="tcp://mq:61616?daemon=true"
 version="5.1.6" />
     </interfaces>
</cluster>
```

4. Provide the locations for the HDFS paths to files.

   For example:

```
<cluster colo="<MyDataCenter>" description="description"
          name="<MyDataCenter>">
     <interfaces>

          <!-- Required for distcp for replications. -->
          <interface type="readonly" endpoint="hftp://nn:50070" version="2.
4.0" />

          <!-- Needed for writing to HDFS-->
          <interface type="write" endpoint="hdfs://nn:8020" version="2.4.
0" />

          <!-- Needed to write to jobs as MapReduce-->
          <interface type="execute" endpoint="rm:8050" version="2.4.0" />

          <!-- Required. Submits Oozie jobs.-->
          <interface type="workflow" endpoint="http://os:11000/oozie/"
 version="4.0.0" />

          <!--Register/deregister partitions in the Hive Metastore and get
 events on partition availability-->
```

```
          <interface type=”registry" endpoint="thrift://hms:9083" version=
"0.13.0" />

          <!--Needed for alerts-->
          <interface type="messaging" endpoint="tcp://mq:61616?daemon=true"
 version="5.1.6" />
     </interfaces>

     <locations>

          <!--HDFS directories used by the Falcon server-->
          <location name="staging" path="/apps/falcon/prod-cluster/
staging" />
          <location name="temp" path="/tmp" />
          <location name="working" path="/apps/falcon/prod-cluster/
working" />
     </locations>
</cluster>
```

The cluster entity is complete if you are using a non-secure environment. If you are using a kerberized environment, continue on with the next step.

5. For secure clusters, define the following properties in all your cluster entities as shown below:

```
<cluster colo="<MyDataCenter>" description="description"
          name="<MyDataCenter>">

     <interfaces>

          <!-- Required for distcp for replications. -->
          <interface type="readonly" endpoint="hftp://nn:50070" version="2.
4.0" />

          <!-- Needed for writing to HDFS-->
          <interface type="write" endpoint="hdfs://nn:8020" version="2.4.
0" />

          <!-- Needed to write to jobs as MapReduce-->
          <interface type="execute" endpoint="rm:8050" version="2.4.0" />

          <!-- Required. Submits Oozie jobs.-->
          <interface type="workflow" endpoint="http://os:11000/oozie/"
 version="4.0.0" />

          <!--Register/deregister partitions in the Hive Metastore and get
 events on partition availability-->
          <interface type=”registry" endpoint="thrift://hms:9083" version=
"0.13.0" />

          <!--Needed for alerts-->
          <interface type="messaging" endpoint="tcp://mq:61616?daemon=true"
 version="5.1.6" />
     </interfaces>

     <locations>

          <!--HDFS directories used by the Falcon server-->
          <location name="staging" path="/apps/falcon/prod-cluster/
staging" />
```

```
          <location name="temp" path="/tmp" />
          <location name="working" path="/apps/falcon/prod-cluster/
working" />
     </locations>

     <properties>
          <property name="dfs.namenode.kerberos.principal" value="nn/$my.
internal@EXAMPLE.COM"/>
          <property name="hive.metastore.kerberos.principal" value="hive/
$my.internal@EXAMPLE.COM"/>
          <property name="hive.metastore.uris" value="thrift://$my.
internal:9083"/>
          <property name="hive.metastore.sasl.enabled" value="true"/>
     </properties>
</cluster>
```

Replace **$my.internal@EXAMPLE.COM** and **$my.internal** with your own values.

Important

Make sure hadoop.security.auth_to_local in core-site.xml is consistent across all clusters. Inconsistencies in rules for hadoop.security.auth_to_local can lead to issues with delegation token renewals.

1.3. Create the Replication Target: Define a Cluster Entity

Replication targets must also be defined as cluster entities. These entities include:

- `colo` and `name` attributes for the cluster.
- Interfaces for the cluster.
- Locations for the HDFS paths to files.
- (For secure clusters only) security properties.

1.4. Create the Feed Entity

The feed entity defines the data set that Falcon replicates. Reference your cluster entities to determine which clusters the feed uses.

1. Create an XML file for the Feed entity.

   ```
   <?xml version="1.0"?>
   ```

2. Describe the feed.

   ```
   <?xml version="1.0"?>
   <feed description="$rawInputFeed" name=”testFeed” xmlns="uri:falcon:feed:0.
   1">
   </feed>
   ```

3. Specify the frequency of the feed.

   ```
   <?xml version="1.0"?>
   ```

```
<feed description="$rawInputFeed" name=”testFeed” xmlns="uri:falcon:feed:0.
1">

<!--Feed run frequency-->
<frequency>hours(1)</frequency>

</feed>
```

4. Choose a retention policy for the data to remain on the cluster.

   For example:

```
<?xml version="1.0"?>
<feed description="$rawInputFeed" name=”testFeed” xmlns="uri:falcon:feed:0.
1">

<!--Feed run frequency-->
<frequency>hours(1)</frequency>

</feed>
```

5. (Optional) Set a late-arrival cut-off policy. The supported policies for late data handling are backoff, exp-backoff (default), and final.

   For example, to set the policy to a late cutoff of 6 hours:

```
<?xml version="1.0"?>
<feed description="$rawInputFeed" name=”testFeed” xmlns="uri:falcon:feed:0.
1">

<!--Feed run frequency-->
<frequency>hours(1)</frequency>

<!-- Late arrival cut-off -->
<late-arrival cut-off="hours(6)”/>

</feed>
```

6. Define your source and target clusters for the feed.

   For example, for two clusters, MyDataCenter and MyDataCenter-secondary cluster:

```
<?xml version="1.0"?>
<feed description="$rawInputFeed" name=”testFeed” xmlns="uri:falcon:feed:0.
1">

<!--Feed run frequency-->
<frequency>hours(1)</frequency>

<!-- Late arrival cut-off -->
<late-arrival cut-off="hours(6)”/>

<!-- Target clusters for retention and replication. -->
<clusters>
     <cluster name="<MyDataCenter>" type="source">
          <validity start="$date" end="$date"/>

          <!--Currently delete is the only action available -->
          <retention limit="days($n)" action="delete">
     </cluster>
```

```
    <cluster name="$MyDataCenter-secondary" type="target">
         <validity start="2012-01-01T00:00Z" end="2099-12-31T00:00Z"/>
         <location type="data” path="/churn/weblogs/${YEAR}-${MONTH}-
${DAY}-${HOUR} "/>
       <retention limit="days(7)" action="delete"/>
    </cluster>
</clusters>
</feed>
```

7. **Specify the HDFS weblogs path locations or Hive table locations.** For example to specify the HDFS weblogs location:

```
<?xml version="1.0"?>

<feed description="$rawInputFeed" name=”testFeed” xmlns="uri:falcon:feed:0.
1">

<!--Feed run frequency-->
<frequency>hours(1)</frequency>

<!-- Late arrival cut-off -->
<late-arrival cut-off="hours(6)”/>

<!-- Target clusters for retention and replication. -->
<clusters>
    <cluster name="<MyDataCenter>" type="source">
         <validity start="$date" end="$date"/>

         <!--Currently delete is the only action available -->
         <retention limit="days($n)" action="delete">
    </cluster>

    <cluster name="$MyDataCenter-secondary" type="target">
         <validity start="2012-01-01T00:00Z" end="2099-12-31T00:00Z"/>
         <location type="data” path="/churn/weblogs/${YEAR}-${MONTH}-
${DAY}-${HOUR} "/>
         <retention limit="days(7)" action="delete"/>
    </cluster>
 </clusters> <locations>

<!-- Global location across clusters - HDFS paths or Hive tables -->
<location type="data" path="/weblogs/${YEAR}-${MONTH}-${DAY}-${HOUR} "/>
</locations>
</feed>
```

8. **Specify HDFS ACLs.** Set the owner, group, and level of permissions for HDFS. For example:

```
<?xml version="1.0"?>
<feed description="$rawInputFeed" name=”testFeed” xmlns="uri:falcon:feed:0.
1">

<!--Feed run frequency-->
<frequency>hours(1)</frequency>

<!-- Late arrival cut-off -->
<late-arrival cut-off="hours(6)”/>

<!-- Target clusters for retention and replication. -->
```

```
<clusters>
     <cluster name="<MyDataCenter>" type="source">
     <validity start="$date" end="$date"/>

     <!--Currently delete is the only action available -->
     <retention limit="days($n)" action="delete">
     </cluster>

     <cluster name="$MyDataCenter-secondary" type="target">
     <validity start="2012-01-01T00:00Z" end="2099-12-31T00:00Z"/>
     <location type="data" path="/churn/weblogs/${YEAR}-${MONTH}-${DAY}-
${HOUR} "/>
     <retention limit="days(7)" action="delete"/>
     </cluster>
</clusters>

<!-- Global location across clusters - HDFS paths or Hive tables -->
<locations>
     <location type="data" path="/weblogs/${YEAR}-${MONTH}-${DAY}-${HOUR} "/
>
</locations>

<!-- Required for HDFS. -->
<ACL owner="hdfs" group="users" permission="0755"/>

</feed>
```

9. **Specify the location of the schema file for the feed as well as the the provider of the schema like protobuf, thrift etc.** For example:

```
<?xml version="1.0"?>
<feed description="$rawInputFeed" name="testFeed" xmlns="uri:falcon:feed:0.
1">

<!--Feed run frequency-->
<frequency>hours(1)</frequency>

<!-- Late arrival cut-off -->
<late-arrival cut-off="hours(6)"/>

<!-- Target clusters for retention and replication. -->
<clusters>
     <cluster name="<MyDataCenter>" type="source">
     <validity start="$date" end="$date"/>

     <!--Currently delete is the only action available -->
     <retention limit="days($n)" action="delete">
     </cluster>

     <cluster name="$MyDataCenter-secondary" type="target">
     <validity start="2012-01-01T00:00Z" end="2099-12-31T00:00Z"/>
     <location type="data" path="/churn/weblogs/${YEAR}-${MONTH}-${DAY}-
${HOUR} "/>
     <retention limit="days(7)" action="delete"/>
     </cluster>
</clusters>

<!-- Global location across clusters - HDFS paths or Hive tables -->
<locations>
```

```
    <location type="data" path="/weblogs/${YEAR}-${MONTH}-${DAY}-${HOUR} "/
>
</locations>

<!-- Required for HDFS. -->
<ACL owner="hdfs" group="users" permission="0755"/>
<schema location="/schema" provider="protobuf"/>
</feed>
```

1.5. Submit and Validate the Entities

- Submit your cluster entities. For example:

  ```
  falcon entity -type cluster -submit -file <YourCluster>.xml
  ```

  For each entity, you should see the following success message for submit:

  ```
  falcon/default/Submit successful ($entity type) $yourEntityFile
  ```

- Submit your feed entity. For example:

  ```
  falcon entity -type feed -submit -file <YourFeed>.xml
  ```

  For each feed entity, you should see the following success message for submit:

  ```
  falcon/default/Submit successful (feed) <YourFeed>
  ```

1.6. Confirm Results

To confirm your results, check your target cluster and review your Oozie jobs.