

APACHE FALCON FEED MANAGEMENT AND DATA PROCESSING PLATFORM

Bin Jiang

04/22/2017

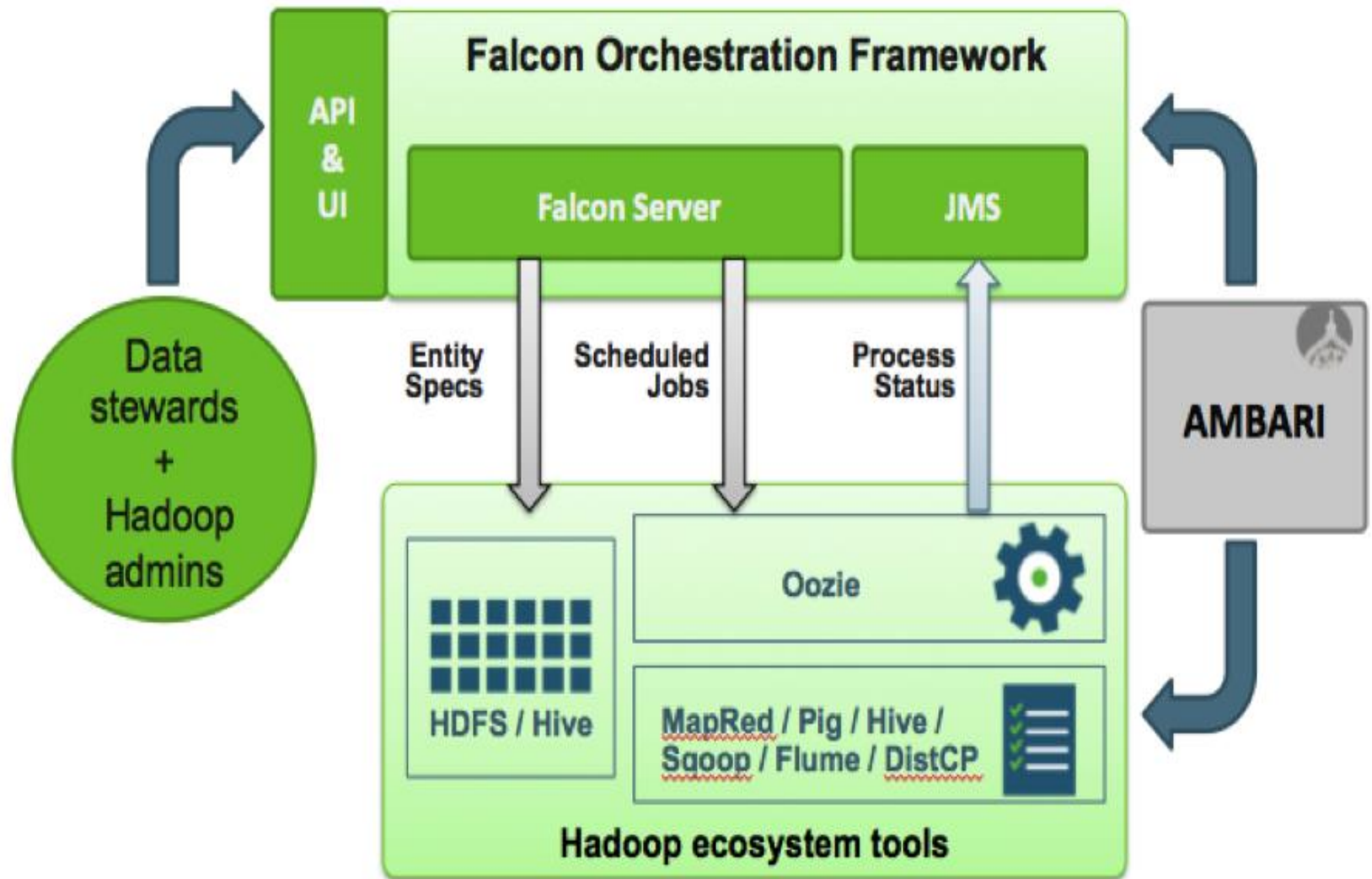
What is Apache Falcon

Apache Falcon is a data governance engine that defines, schedules, and monitors data management policies. Falcon allows Hadoop administrators to centrally define their data pipelines, and then Falcon uses those definitions to auto-generate workflows in Apache Oozie

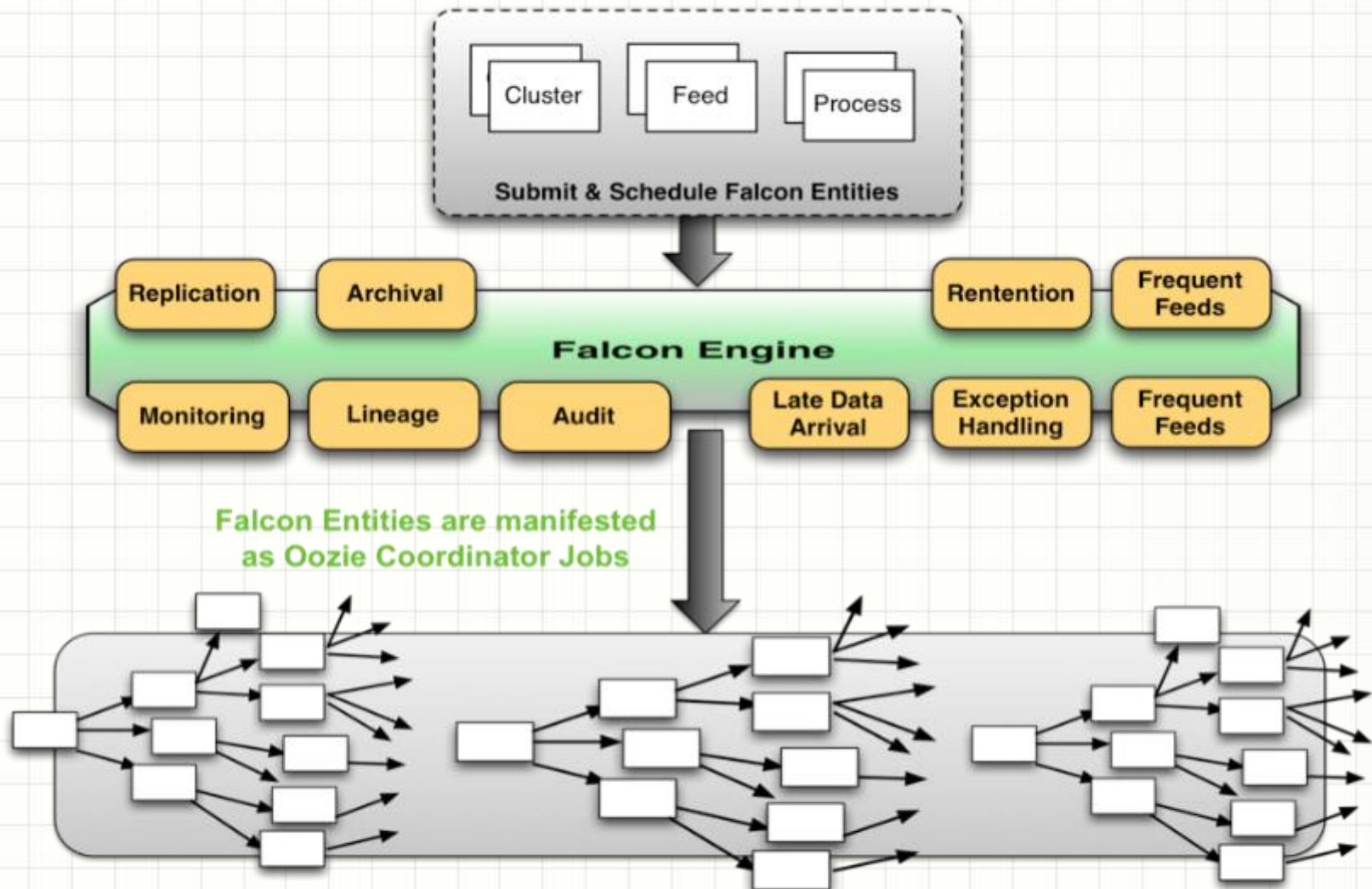
What Does Apache Falcon Do

- Simplifies complicated data management workflows into generalized entity definitions
- Define data pipelines Easily
- Monitor data pipelines in coordination with Ambari
- Trace pipelines for dependencies, tagging, audits and lineage

Falcon Architecture



Apache Falcon Workflow



Why Apache Falcon

- Apache Oozie Challenges:
 - Difficult to manage so many data set and process definitions
 - Processes might use the wrong copies of data sets
 - Data sets and processes may be duplicated
 - More difficult to track down where a particular data set originated

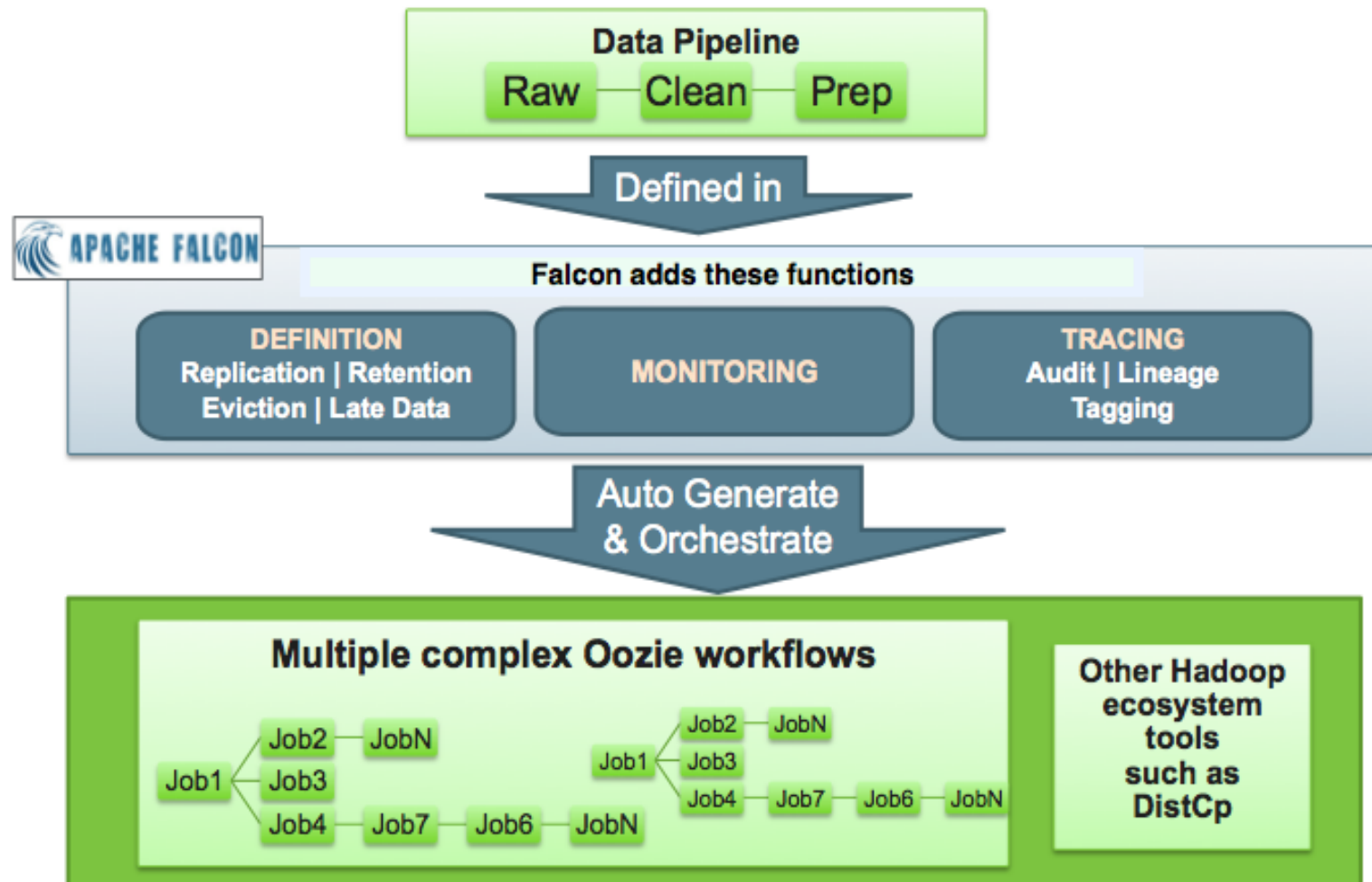
Why Apache Falcon

- How do Apache Falcon Address:
 - Falcon addresses these data governance challenges with high-level
 - Reusable “entities” that can be defined once and re-used many times
 - Data management policies are defined in Falcon entities and manifested as Oozie workflows
 - Compliance and audit
 - Database replication and archival
 - Leverage other HDP components, such as Pig, HDFS, and Oozie

Data Pipeline

- Falcon entities are configured to form a data pipeline
- A pipeline consists of a dataset and the processing that acts on the dataset across the HDFS cluster
- Configure data pipelines for data replication and mirroring
- Define cluster storage locations and interfaces, dataset feeds, and the processing logic to be applied to the datasets
- Entities act together to provide a dynamic flow of information to load, clean, and process data
- Each entity is defined separately and then linked together to form a data pipeline

Data Pipeline Flow

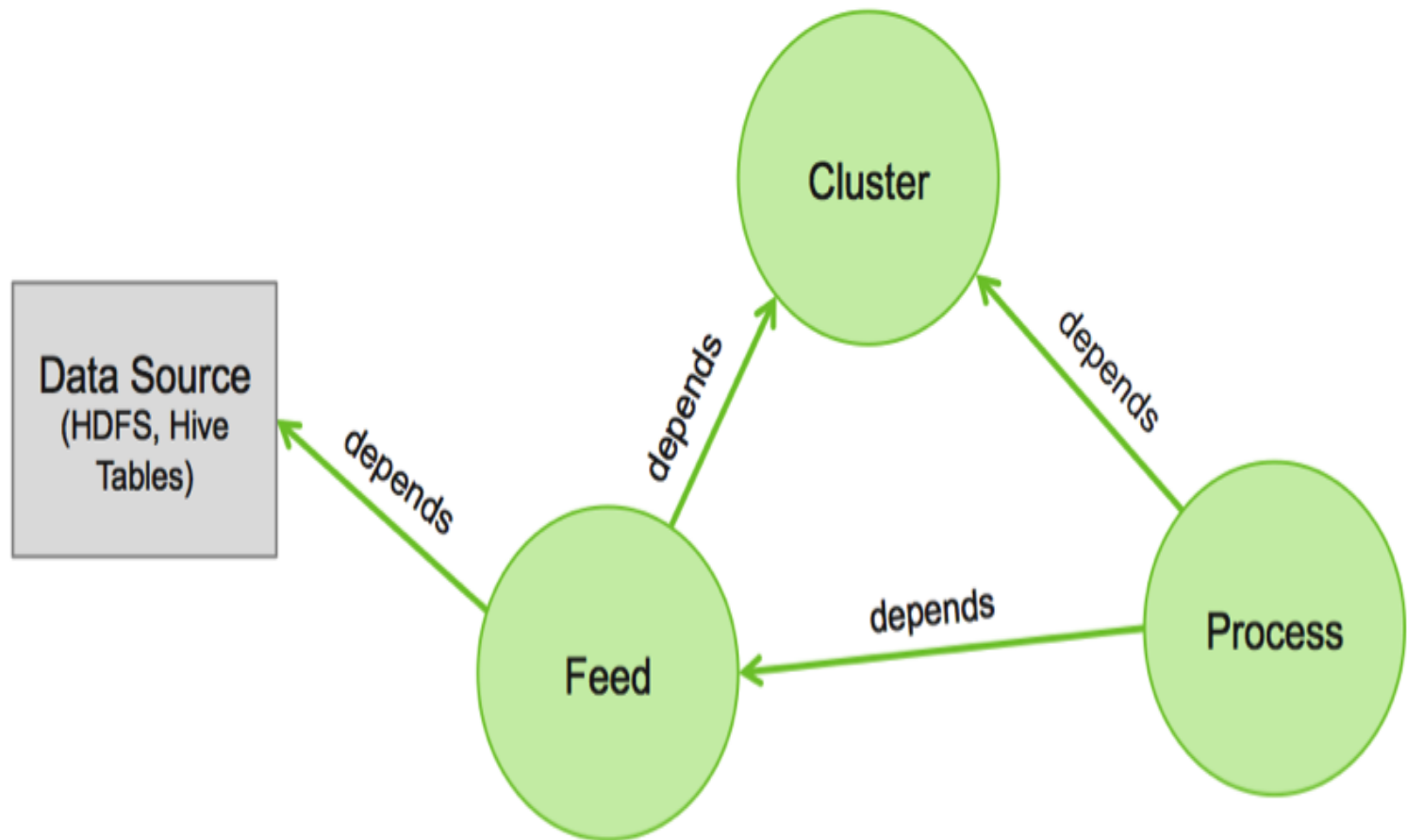


Falcon Entities

Entities that can be combined to describe all data management policies and pipelines:

- ☐ Cluster
- ☐ Feed (i.e. Data Set)
- ☐ Process
- ☐ Mirror
- ☐ Datasource

Falcon Entities



Falcon Policies

- Falcon provides predefined policies for data replication, retention, late data handling, retry, and replication
- These sample policies are easily customized to suit the needs
- Entities can be reused many times to define data management policies for Oozie jobs, Spark jobs, Pig scripts, and Hive queries

Falcon ACL

- Defines the cluster, including its interfaces, where data and processes are stored
Using the Falcon graphical user interface, accessible from Apache Ambari either stand-alone or as an Ambari View
- Using the Falcon CLI
- Using the Falcon APIs, as a service

Cluster Entity

- Defines the cluster, including its interfaces, where data and processes are stored

Creating a Cluster Entity Definition Using the Web UI

- **Cluster Entity General Properties**
 - **Cluster Name**
 - **Data Center or Colo Name and Description**
 - **Tags**

Creating a Cluster Entity Definition Using the Web UI

- **Cluster Entity Interface Properties**
 - **File System Read Endpoint Address** - Required for DistCp
 - **File System Default Address** - Falcon uses this interface to export data from HDFS to the data source
 - **YARN Resource Manager Address**
 - **Workflow Address** - Required to submit workflow jobs, e.g. Oozie Server URL
 - **Message Broker Address** - Required to send alerts
 - **Metadata Catalog Registry** - Use to register or deregister partitions in the Hive Metastore and to fetch events about partition availability
 - **Spark** – Yarn Client or Cluster

Creating a Cluster Entity Definition Using the Web UI

- Cluster Entity Properties & Location Properties
 - **Properties** - Specify a name and value for each property
 - **Location** - Specify the HDFS locations for the staging, temp, and working directories
 - **Access Control List**

Creating a Cluster Entity Definition Using the Web UI

NEW CLUSTER

Cluster Name*	Data Center or Colo Name*
<input type="text"/>	<input type="text"/>

You need to specify a name

Description

Tags

<input type="text" value="key"/>	<input type="text" value="value"/>	+ add tag
----------------------------------	------------------------------------	-----------

INTERFACES

Type	Endpoint
File System Read Endpoint Address	<input type="text" value="http://<hostname>:50070"/>
File System Default Address	<input type="text" value="hdfs://<hostname>:8020"/>
Yarn Resource Manager Address	<input type="text" value="<hostname>:8050"/>
Workflow Address	<input type="text" value="http://<hostname>:11000/oozie/"/>
Message Broker Address	<input type="text" value="tcp://<hostname>:61616?daemon=true"/>
<input type="checkbox"/> Metadata Catalog Registry	<input type="text" value="thrift://<hostname>:9083"/>
<input type="checkbox"/> Spark	<div><input type="radio"/> Yarn Cluster <input type="radio"/> Yarn Client <input type="radio"/> Local <input type="radio"/> Custom</div> <input type="text"/>

PROPERTIES

Property Name	Value
<input type="text" value="name"/>	<input type="text" value="value"/>

+ add property

LOCATION

Location Name	Path
---------------	------

Creating a Cluster Entity Definition Using the CLI

```
<?xml version="1.0" encoding="UTF-8"?>
<cluster colo="toronto" description="Primary Cluster"
name="primary-cluster"
xmlns="uri:falcon:cluster:0.1">
<tags>class=production,site=canada-east</tags>
<interfaces>
<interface type="readonly" endpoint="hdfs://sandbox.hortonworks.com:8020" version="2.4.0"/>
<interface type="write" endpoint="hdfs://sandbox.hortonworks.com:8020" version="2.4.0"/>
<interface type="execute" endpoint="sandbox.hortonworks.com:8050" version="2.4.0"/>
<interface type="workflow" endpoint="http://sandbox.hortonworks.com:11000/oozie/"
version="4.1.0"/>
<interface type="registry" endpoint="thrift://sandbox.hortonworks.com:9083" version="0.13.0"/>
<interface type="messaging" endpoint="tcp://sandbox.hortonworks.com:61616?daemon=true"
version="5.4.3"/> </interfaces>
<locations>
<location name="staging" path="/tmp"/>
<location name="working" path="/tmp"/>
<location name="temp" path="/tmp"/>
</locations>
</cluster>
```

Cluster Entity

Submit the Cluster Entity to Falcon:

```
falcon entity -type cluster -submit -file primary-  
cluster.xml
```

Feed Entity

- Defines the datasets to be cleaned and processed
- The Feed entities have policies attached to them that need to be explicitly scheduled by Falcon
- Falcon takes the retention, replication, feed frequency, and delays and creates Oozie Coordinator jobs to automate all of these actions for you
- To process data, define two feed entities: One for data input and one for data output

Creating a Feed Entity Definition Using the Web UI

- **General Feed Properties**
 - **Feed Name and Description**
 - **Tags**
 - **Feed Groups**
 - **Type** - Hive, HDFS, RDBMS Import, and RDBMS Export

Creating a Feed Entity Definition Using the Web UI

- **Hive Source and Target Feed Properties**
 - **Cluster**
 - **Table URI**
 - **Start and End Times**
 - **Retention**
 - **Frequency**

Creating a Feed Entity Definition Using the Web UI

- **HDFS Source and Target Feed Properties**
 - **Cluster**
 - **Data Path**
 - **Statistics Path**
 - **Start and End Times**
 - **Retention**
 - **Frequency**

Creating a Feed Entity Definition Using the Web UI

- **RDBMS Import Source and Target Feed Properties**
 - **Datasource** - Connection information for a remote data source
 - **Table**
 - **Extract Type** - Full or Incremental
 - **Merge Type** - Snapshot or Append
 - **Columns** - Include or Exclude
 - **Location** - Hive or HDFS
 - **Cluster**
 - **Data Path** - HDFS only
 - **Statistics Path**
 - **Table URI** - Hive only
 - **Start and End Times**
 - **Retention**
 - **Frequency**

Creating a Feed Entity Definition Using the Web UI

- **RDBMS Export Source and Target Feed Properties**
 - **Location** - HDFS or Hive
 - **Cluster**
 - **Data Path** - HDFS only
 - **Statistics Path**
 - **Table URI** - Hive only
 - **Start and End Times**
 - **Retention**
 - **Datasource** - Connection information for a remote data source
 - **Table**
 - **Load Method** - Update Only or Allow Insert
 - **Columns**
 - **Frequency**

Creating a Feed Entity Definition Using the Web UI

- **Advanced Feed Properties**
 - **Queue Name** - Hadoop job queue
 - **Job Priority** - Hadoop job priority
 - **Late Arrival Cutoff** - The timeframe within which a dataset feed can arrive late and still be processed
 - **Availability Flag** - Specifies the name of a file that when present in a feed's data directory, determines that the feed is available
 - **Max Map Jobs** - The maximum number of maps used during replication
 - **Max Bandwidth** - The bandwidth in MB/s used by each mapper during replication
 - **Parallel** - The concurrent replication instances that can run at any given time
 - **Access Control List**
 - **Feed Schema**

Creating a Feed Entity Definition Using the Web UI

Feed Name*

You need to specify a name

Description

Tags

+ add tag

Feed Groups (comma separated)

Type*

-Select feed type- ▼

Frequency

Repeat Every*

hours ▼

Timezone*

UTC ▼

ADVANCED OPTIONS ▲

Job Details

Queue Name

Job Priority

Normal ▼

Late Arrival Cutoff

hours ▼

Availability Flag

Feed Entity

Key Parameter:

- Feed Frequency
 - How often this feed will land
- Cluster
 - The cluster(s) where this feed will land
- Retention Policy
 - How long the data will remain on the cluster
- Data Set Locations
 - The HDFS path to the location of the files

Creating a Feed Entity Definition Using the CLI

```
<feed description="raw input feed" name="raw-input-feed"  
xmlns="uri:falcon:feed:0.1">  
<tags>owner=landing,pipeline=adtech,category=click-logs</tags>  
<frequency>minutes(60)</frequency>  
<clusters>  
<cluster name="primary-cluster" type="source">  
<validity start="2014-03-01T00:00Z" end="2016-01-01T00:00Z"/>  
<retention limit="days(90)" action="delete"/>  
</cluster>  
</clusters>  
<locations>  
<location type="data" path="/landing/raw-input-feed/${YEAR}-${MONTH}-  
${DAY}-${HOUR}"/>  
</locations>  
<ACL owner="adam" group="landing" permission="0x755"/>  
<schema location="/none" provider="none" />  
</feed>
```

Feed Entity

- Submit these entities:

falcon entity -type feed -submit -file raw-input-feed.xml

- Schedule the Feeds:

falcon entity -type feed -schedule -name raw-input-feed

Process Entity

- Defines how the process (such as Hive jobs) works with the dataset on top of a cluster
- A process consumes feeds, invokes processing logic (including late data handling), and produces further feeds
- A process also defines the configuration of the Oozie workflow and defines when and how often the workflow should run

Creating a Process Entity Definition Using the Web UI

- **General Process Properties**

- **Process Name**

- **Tags**

Creating a Process Entity Definition Using the Web UI

- **Process Detail and Engine Properties**
 - **Engine** - Spark, Oozie, Pig, and Hive
 - **Workflow Name**
 - **Workflow Path** - The path to the workflow engine on HDFS
 - **Cluster**
 - **Inputs** - The input data for the workflow. Each input maps to a feed
 - **Outputs** - The output data that is generated by the workflow. Each output is mapped to a feed and the output path is picked up from the feed definition
 - **Name** - Name of the Spark application
 - **Application** - Spark only, Specifies the .jar or Python file to be executed by Spark
 - **Run Duration Start and End**
 - **Frequency**
 - **Timezone**

Creating a Process Entity Definition Using the Web UI

- **Advanced Process Properties**
 - **Retry Policy Type** - Periodic, Exponential Backup, and None
 - **Delay Up to** - The time period after which a retry attempt is made
 - **Attempts**
 - **Max Parallel Instances**
 - **Order** - FIFO, LIFO or Last_Only
 - **Properties**
 - **Access Control List**

Creating a Process Entity Definition Using the Web UI

Tags

key value [+ ADD](#)

Details

Engine* [+ ADD](#) INPUT(S) [+ ADD](#) OUTPUT(S) [+ ADD](#)

Workflow Name*

Workflow Path*

Cluster*

Run Duration*

Start :

End :

Frequency

Repeat Every*

Timezone*

ADVANCED OPTIONS ^

Retry Policy

Process Entity

Built-in actions:

- Any Oozie workflow
- A Hive script
- A Pig script

Process Entity

Key Parameters:

- Clusters
 - The site or sites where the process is executed
- Frequency
 - The frequency of the job execution.
- Inputs
 - Specify the name of the Feed entity for input and the time range of the data to use in the process.
- Outputs
 - specify the name of the Feed entity for output
- Retry
 - control the number of retry attempts and the delay between each attempt.

Creating a Process Entity Definition Using the CLI

```
<process name="filter-process" xmlns="uri:falcon:process:0.1">
<tags>owner=landing,pipeline=adtech,category=click-logs</tags>
<clusters>
<cluster name="primary-cluster">
<validity start="2014-03-02T00:00Z" end="2016-04-30T00:00Z"/>
</cluster>
</clusters>
<parallel>1</parallel>
<order>FIFO</order>
<frequency>days(1)</frequency>
<inputs>
<input name="input" feed="raw-input-feed" start="yesterday(0,0)" end="today(-
1,0)" />
</inputs>
<outputs>
<output name="output" feed="filtered-feed" instance="yesterday(0,0)" />
</outputs>
<workflow engine="pig" path="/landing/scripts/simplefilter.pig" />
<retry policy="periodic" delay="minutes(30)" attempts="2" />
</process>
```

Process Entity

Submit and Schedule the process:

```
falcon entity -type process -submit -file filter-process.xml
```

```
falcon entity -type process -schedule -name filter-process
```


Mirroring Data with Falcon

- Mirror data between on-premise clusters
- Mirror data between an on-premises HDFS cluster and a cluster in the cloud using Microsoft Azure or Amazon S3

Mirror File System Data Using the Web UI

- **General HDFS Mirror Properties**
 - **Mirror Name**
 - **Tags**

Mirror File System Data Using the Web UI

- **Source and Target Mirror Properties**
 - **Source Location** - HDFS, Azure or S3
 - **Source Cluster**
 - **Source Path**
 - **Target Location** - HDFS, Azure or S3
 - **Target Cluster**
 - **Target Path**
 - **Run job here**
 - **Validity Start and End**
 - **Frequency**
 - **Timezone**
 - **Send alerts to**

Mirror File System Data Using the Web UI

- **Advanced HDFS Mirror Properties**
 - **Max Maps for DistCp** - The maximum number of maps used during replication
 - **Max Bandwidth**
 - **Retry Policy** - Periodic, Exponential Backoff, and Final
 - **Delay**
 - **Attempts**
 - **Access Control List**

Mirror File System Data Using the Web UI

NEW HDFS MIRROR

Mirror Name*

You need to specify a name

Tags

key

value

+ add tag

Source

☒ Run job here

Target

☐ Run job here

Location

☒ HDFS ☐ Azure ☐ S3

Cluster*

-Select cluster-

Path*

Path

Location

☒ HDFS ☐ Azure ☐ S3

Cluster*

-Select cluster-

Path*

Path

Validity

Start*

04/27/2017

Begin Time*

09

:

49

PM

End*

12/31/2099

End Time*

11

:

59

AM

Frequency

Repeat Every

1

days

Timezone*

UTC

Mirror Hive Data Using the Web UI

- **General Hive Mirror Properties**

- **Mirror Name**

- **Tags**

Mirror Hive Data Using the Web UI

- **Source and Target Hive Mirror Properties**
 - **Cluster, Source & Target**
 - **HiveServer2 Endpoint, Source & Target**
 - **Hive2 Kerberos Principal, Source & Target**
 - **Meta Store URI, Source & Target**
 - **Kerberos Principal, Source & Target**
 - **Run job here**
 - **I want to copy** - copy one or more Hive databases or copy one or more tables
 - **Validity Start and End**
 - **Frequency**
 - **Timezone**
 - **Send alerts to**

Mirror Hive Data Using the Web UI

- **Advanced Hive Mirror Properties**
 - **TDE Encryption** - Enables encryption of data at rest
 - **Max Maps for DistCp**
 - **Max Bandwidth**
 - **Retry Policy**
 - **Delay**
 - **Attempts**
 - **Access Control List**

Mirror Hive Data Using the Web UI

NEW HIVE MIRROR

Mirror Name*

You need to specify a name

Tags

Mirror

File System

Hive

Snapshot

Data Source

Source

☐ Run job here

Target

☒ Run job here

Cluster*

HiveServer2 endpoint

I want to copy* ☒ Entire databases ☐ Tables in a single database

Databases (comma separated)*

Cluster*

HiveServer2 endpoint

Validity

Start*

Begin Time*

:

Mirror Data Using Snapshots

- Snapshot-based mirroring is an efficient data backup method because only updated content is actually transferred during the mirror job
- Mirror snapshots from a single source directory to a single target directory. The destination directory is the target for the backup job

Mirror Data Using Snapshots

- **Source and Target Snapshot Mirror Properties**
 - **Source, Cluster**
 - **Target, Cluster**
 - **Source, Directory**
 - **Source, Delete Snapshot After** - Specify the time period after which the mirrored snapshots are deleted from the source cluster
 - **Source, Keep Last** - Specify the number of snapshots to retain on the source cluster
 - **Target, Directory**
 - **Target, Delete Snapshot After**
 - **Target, Keep Last**
 - **Run job here**
 - **Run Duration Start and End**
 - **Frequency**
 - **Timezone**

Mirror Data Using Snapshots

- **Advanced Snapshot Mirror Properties**

- **TDE Encryption**
- **Retry Policy**
- **Delay**
- **Attempts**
- **Max Maps**
- **Max Bandwidth**
- **Send alerts to**
- **Access Control List**

Mirror Data Using Snapshots

NEW SNAPSHOT BASED MIRROR

Name*

You need to specify a name

Tags

name

value

+ add tag

Source

☐ Run job here

Target

☒ Run job here

Cluster*

-Select source cluster-

Source Directory*

You need to provide a path

Delete Snapshot After

14

days

Keep Last

90

snapshots

Cluster*

-Select target cluster-

Target Directory*

Delete Snapshot After

14

days

Keep Last

90

snapshots

Run Duration

Start

04/27/2017

Begin Time

11

:

42

PM

Datasource

NEW DATA SOURCE

Connection Name*

Data Center or Colo Name*

Description

Tags

key

value

+ add tag

Database Manager*

- Select Database Manager -

Connection String*

Driver Jar*

[+ Add](#)

Driver Class*

Type*

☒ Read only ☐ Write

Credential Type*

☐ Username / Password ☐ Password File ☐ Password Alias

Sqoop Properties

☐ Direct ☐ Verbose

[+ Add Property](#)

Sqoop Parameter File

ADVANCED OPTIONS ▼

TEST

Late Data Handling

- Late data handling in Falcon defines how long data can be delayed and how that late data is handled
- For example, a late arrival cut-off of hours(6) in the feed entity means that data for the specified hour can delay as much as 6 hours later
- Policies for late data handling:
 - backoff: Take the maximum late cut-off and check every specified time
 - exp-backoff (default): Recommended. Take the maximum cut-off date and check on an exponentially determined time
 - final: Take the maximum late cut-off and check once

Late Data Handling - Example

- Specify the cut-off time in your feed entity.
 - For example, to set a cut-off of 4 hours:
<late-arrival cut-off="hours(4)"/>
- Specify a check for late data in all your process entities that reference that feed entity.
 - For example, to check each hour until the cut-off time with a specified policy of backoff and a delay of 1 hour:

```
<late-process policy="exp-backoff" delay="hours(1)">  
  <late-input input="input" workflow-  
    path="/apps/clickstream/late" />  
</late-process>
```

Setting a Retention Policy

Falcon kicks off the retention policy on the basis of the time value you specify:

- Less than 24 hours: Falcon kicks off the retention policy every 6 hours
- More than 24 hours: Falcon kicks off the retention policy every 24 hours
- When a feed is scheduled: Falcon kicks off the retention policy immediately

Setting a Retention Policy

```
<clusters>
```

```
  <cluster name="corp" type="source">
```

```
    <validity start="2012-01-30T00:00Z" end="2013-03-31T23:59Z"
```

```
      timezone="UTC" />
```

```
    <retention limit="$unitOfTime($n)" action="delete" /> <!--Retention  
policy. -->
```

```
  </cluster>
```

```
</clusters>
```

Setting a Retry Policy

```
<process name="[sample-process]">
```

```
...
```

```
  <retry policy="periodic" delay="minutes(10)" attempts="3"/>
```

```
...
```

```
</process>
```


Process Engine - Spark

NEW PROCESS

Process Name*

You need to specify a name

Tags

+ ADD

Details

Engine*

Workflow Path*

Cluster*

Name*

INPUT(S)

+ ADD

OUTPUT(S)

+ ADD

Application*

Main Class

Runs On

Mode*

☒ Cluster ☐ Client

Spark Options

Spark Arguments

Falcon REST API

- REST Call on Entity Resource
- REST Call on Feed/Process Instances
- REST Call on Admin Resource
- REST Call on Lineage Graph Resource
- REST Call on Metadata Resource
- REST Call on Extension artifact
- REST Call on Extension Job Management

Falcon EL Expression

Falcon expression language can be used in process definition for giving the start and end instance for various feeds

- `now(hours,minutes)`
- `today(hours,minutes)`
- `yesterday(hours,minutes)`
- `lastYear(month,day,hour,minute)`
- `currentMonth(day,hour,minute)`
- `lastMonth(day,hour,minute)`
- `currentYear(month,day,hour,minute)`
- `lastYear(month,day,hour,minute)`
- `latest(number of latest instance)`
- `currentWeek(weekDayName,hour,minute)`
- `lastWeek(weekDayName,hour,minute)`

Apache Falcon Futures

- Distributed Cluster Management with Apache Falcon Prism
- Management UI
- Data Lineage
- Data Management Alerting and Monitoring
- Falcon Integration and Extensibility via the REST API
- Secure Data Management