Data Pipeline testing now made easy!

With Apache Falcon
$whoami

- Pallavi Rao
  - Architect, InMobi
  - Committer, Apache Falcon
  - Contributor, Apache PIG

- Pavan Kumar Kolamuri
  - Sr. Software Engineer, InMobi
  - Contributor, Apache Falcon and Oozie
What is in store for you?

❖ Some history and introduction to Apache Falcon
❖ Falcon Unit - A new feature in v0.7
❖ Falcon Unit - How it simplifies testing pipelines
❖ Demo, Q&A
Once upon a time...
What kept us up at night?

- Failures
- Data arriving late
- Re-processing
- Varied Data Replication
- Varied Data Retention
- Data Archival
- Lineage
- SLA monitoring
The pattern
The concoction
Concoction.. distributed
Some maladies cured

- Process Management
  - Relays
  - Late Data Handling
  - Failure Retries
  - Reruns

- Data Management
  - Data Import/Export
  - Retention
  - Replication
  - Archival

- Data Governance
  - Lineage
  - Audit
  - SLA
Sample pipeline in Falcon

- **Click Logs** → **Click Enhancer** → **Enhanced Clicks** → **Hourly Aggregation** → **Hourly Clicks**
- **Metadata** with Retention: 2 hours, Frequency: 5 mins, Late Data arrival
- **Daily Aggregation** with Retention: 1 day
- **Daily clicks** with Retention: 7 days, Replication required

Falcon Feed

Falcon Process
Cluster Specification

<cluster colo="default" description="" name="corp" xmlns="uri:falcon:cluster:0.1">
<tags>consumer=consumer@xyz.com, owner=producer@xyz.com, _department_type=forecasting</tags>
<interfaces>
    <interface type="readonly" endpoint="webhdfs://localhost:14000" version="1.1.2"/>
    <interface type="write" endpoint="hdfs://localhost:9000" version="1.1.2"/>
    <interface type="execute" endpoint="localhost:8032" version="1.1.2"/>
    <interface type="workflow" endpoint="http://localhost:11000/oozie/" version="4.1.0"/>
    <interface type="registry" endpoint="thrift://localhost:12000" version="0.11.0"/>
</interfaces>
<location name="staging" path="/projects/falcon/staging"/>
<location name="temp" path="/tmp"/>
<location name="working" path="/projects/falcon/working"/>
<properties>
    <property name="field1" value="value1"/>
</properties>
</cluster>
Feed specification

```xml
<feed
description="enhanced clicks replication feed"
name="repl-feed"
xmlns="uri:falcon:feed:0.1">
    <frequency>minutes(5)</frequency>
    <late-arrival cut-off="hours(1)"/>
    <sla slaLow="hours(2)" slaHigh="hours(3)"/>
    <clusters>
        <cluster
            name="corp"
type="source">
            <validity
                start="2013-01-01T00:00Z"
end="2030-01-01T00:00Z"/>
            <retention
                limit="days(2)"
action="delete"/>
        </cluster>
        <cluster
            name="secondary"
type="target">
            <validity
                start="2013-11-15T00:00Z"
end="2030-01-01T00:00Z"/>
            <retention
                limit="days(2)"
action="delete"/>
            <locations>
                <location
                    type="data"
path="/data/clicks/repl-enhanced/${YEAR}/${MONTH}/${DAY}/${HOUR}/${MINUTE}"/>
            </locations>
        </cluster>
    </clusters>
</feed>
```
Process specification

```xml
<process name="clicks-hourly" xmlns="uri:falcon:process:0.1">
  <clusters>
    <cluster name="corp">
      <validity start="2011-11-02T00:00Z" end="2011-12-30T00:00Z"/>
    </cluster>
  </clusters>
  <parallel>1</parallel>
  <order>LIFO</order>
  <frequency>hours(1)</frequency>
  <inputs>
    <input name="click" feed="clicks-enhanced" start="yesterday(0,0)" end="latest(0)" partition="*/US"/>
  </inputs>
  <outputs>
    <output name="clicksummary" feed="click-hourly" instance="today(0,0)"/>
  </outputs>
  <workflow name="test" version="1.0.0" engine="oozie" path="/user/guest/workflow" lib="/user/guest/workflowlib"/>
  <retry policy="periodic" delay="hours(10)" attempts="3"/>
  <late-process policy="exp-backoff" delay="hours(1)"
    <late-input input="click" workflow-path="hdfs://clicks/late/workflow"/>
  </late-process>
</process>
```

Where should the process run?

How should the process run?

What to consume?

What to produce?

Processing logic

Late Data processing
Why Falcon Unit?
Before Falcon Unit

Unit Tests for each module using either PigUnit or MRUnit or JUnit.

Integration Tests executed by bringing up AWS instances with the entire stack.
Before Falcon Unit

Falcon Feed
Falcon Process

OK
Invalid
Input
Invalid
Output
OK
Spec.
Invalid
Improper
Replication

OK
OK
OK
Motivation for Falcon Unit

- User errors caught only at deploy time.
  - Input/Output feeds and paths not getting resolved.
  - Errors in specification.
- Integration Tests require environment setup/tearDown.
  - Messy deployment scripts.
  - Time consuming.
- Debugging was cumbersome.
  - Logs scattered.
Falcon Unit

In Process execution env.
- Local Oozie
- Local File System
- Local Job Runner
- Local Message Queue

Actual cluster
- Oozie
- HDFS
- YARN
- Active MQ
What you can test

Process Management

- Validation of definition
- Entity scheduling and status verification
- Correctness of data window being picked up.
- Reruns
- Missing dependencies/properties

Data Management

- Data creation
- Data injection
- Retention
- Replication

Data Governance

- Lineage
- Data availability for verification
After Falcon Unit

Falcon Feed

Falcon Process

OK

TESTED OK
Falcon Unit Illustrated
Capabilities with example

❖ Entity creation and data flow validation.
❖ Data Injection.
❖ Data Retention and Replication.
❖ Seamless API for cluster and local mode.
Example Pipeline

- Hourly Clicks
- Deferred Clicks
- Daily Aggregation
- Daily clicks

Consumes

Produces
Cluster Creation

Cluster Creation :

→ Local Mode

\[
\text{submit(EntityType.Cluster, coloName, clusterName, propsMap);}
\]
\[
\text{submitCluster(); - Uses defaults}
\]

→ Cluster Mode

\[
\text{submit(EntityType.Cluster, <Path to Cluster XML>);}
\]
Feed Creation

Submit Feed

submit(EntityType.Feed, <Path to Hourly Clicks XML>);

Inject Data

createData("HourlyClicks", "local", scheduleTime, <test data path>, numinstances);
Process Submission:

submit(EntityType.Process, <Path to Daily clicks Agg XML>); → Local
submit(EntityType.Process, <Path to Daily clicks Agg XML>); → Cluster Mode

Process Scheduling:

scheduleProcess("daily_clicks_agg", startTime, numInstances, clusterName);

Process Verification:

getInstanceStatus(EntityType.Process, "daily_clicks_agg", scheduleTime);
Data Retention:

- Data retention can be validated by scheduling feed in both cluster mode and local mode
  
  ```java
  createData("HourlyClicks", "local", timeStamp, <test data path>);
  schedule(EntityType.FEED, "HourlyClicks", "local");
  status = getInstanceStatus(EntityType.FEED, "HourlyClicks");
  ```

- Falcon Unit provides APIs for validation of existence of paths.
Data Replication

Data Replication can also be tested using Falcon Unit:

```java
submitCluster(coloName, srcCluster, propsMap);
submitCluster(coloName, targetCluster, propsMap);
createData("HourlyClicks", "srcCluster", timeStamp, <test data path>);
schedule(EntityType.FEED, "HourlyClicks", targetCluster);
status = getInstanceStatus(EntityType.FEED, feed, targetCluster);
Assert.assertEquals(status, WorkflowStatus.SUCCEEDED);
```
Going forward ...

- Improved data injection
  - Generation of test data from template
  - Sampling of production data for testing
- Support for other data lifecycle operations
  - Data ingestion, export
- Maven plugin for build time validation of definitions.
Questions?

If you want to ask later - user@falcon.apache.org
If you want to contribute - dev@falcon.apache.org