任务调度框架Oozie

谭唐华

课程大纲

- Oozie介绍
- Oozie部署
- Oozie案例

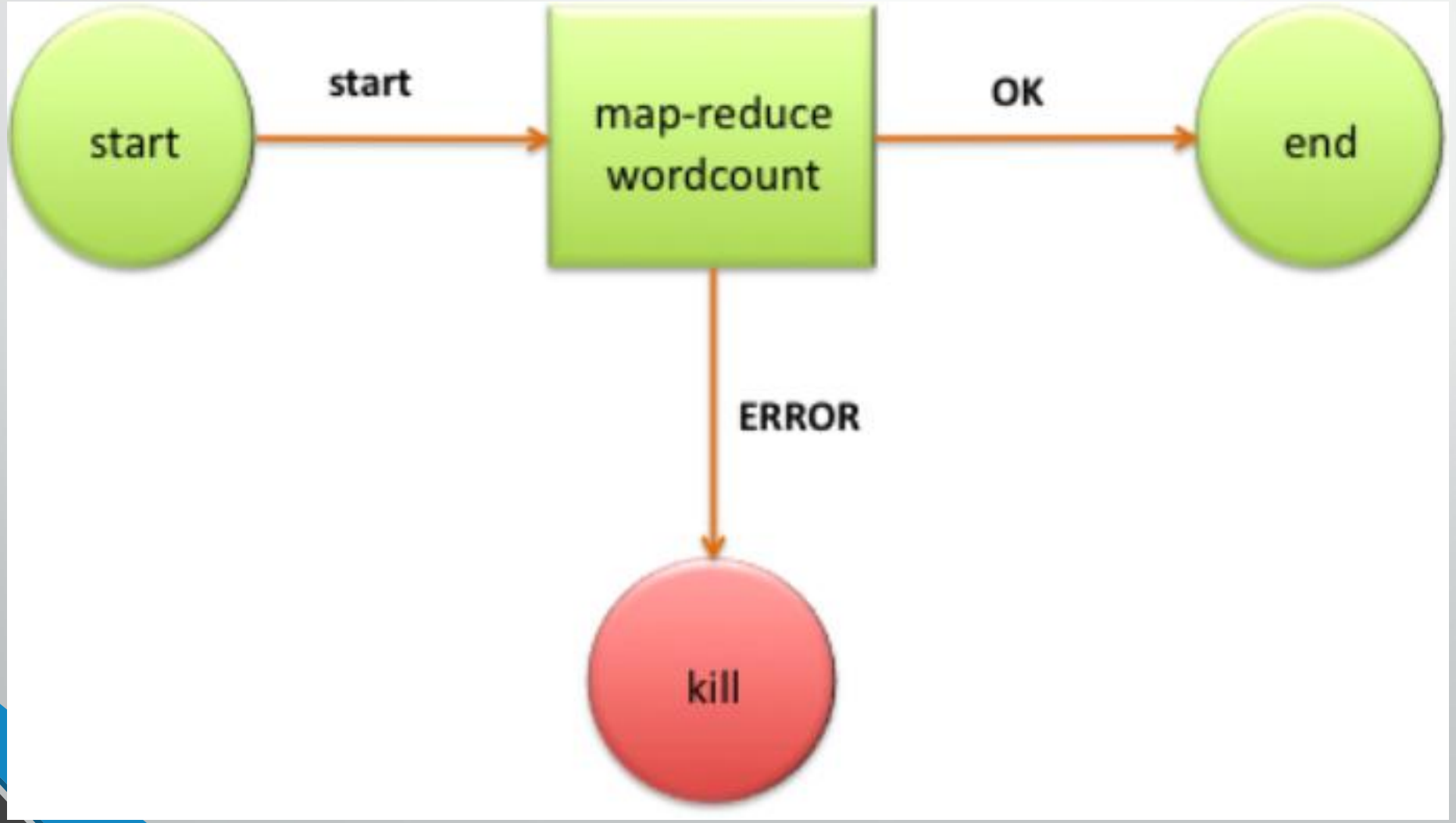
Oozie简介

- 一个基于工作流引擎的开源框架，是由Cloudera公司贡献给Apache的，它能够提供对Hadoop MapReduce和Pig Jobs的**任务调度与协调**。Oozie需要**部署到Java Servlet容器**中运行。
- Oozie工作流定义，同JBoss jBPM提供的jPDL一样，也提供了类似的**流程定义语言hPDL**，通过XML文件格式来实现流程的定义。对于工作流系统，一般都会有很多不同功能的节点，比如分支、并发、汇合等等。
- Oozie定义了**控制流节点（Control Flow Nodes）和动作节点（Action Nodes），**其中控制流节点定义了流程的开始和结束，以及控制流程的执行路径（Execution Path），如decision、fork、join等；而动作节点包括Hadoop map-reduce、Hadoop文件系统、Pig、SSH、HTTP、eMail和Oozie子流程。

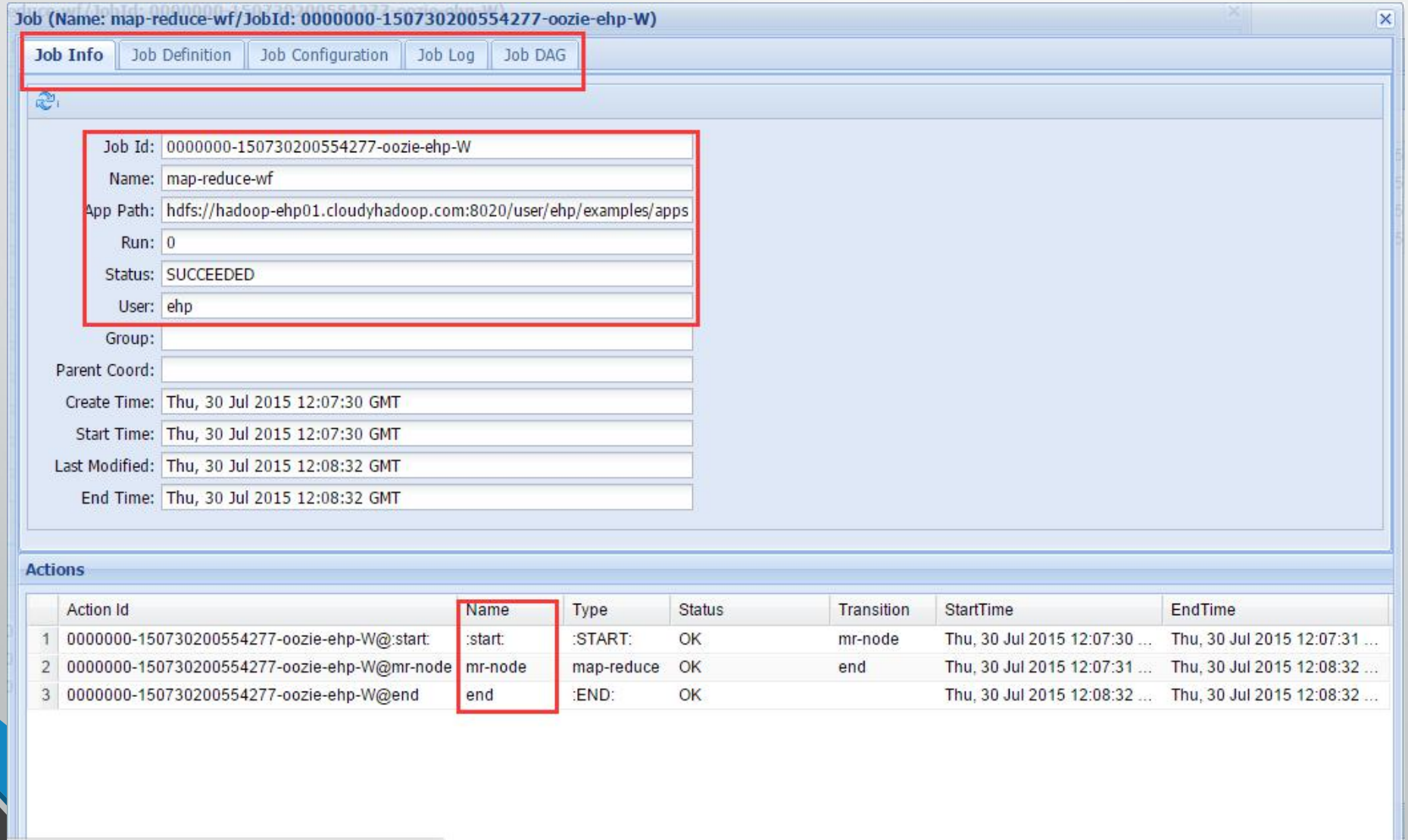
Oozie版本

- oozie 1.x 运行mapreduce，pig任务的工作流job。
- oozie 2.x 加入基于时间和数据触发的协调器引擎，根据配置的时间间隔和数据是否可用连续不断的运行。
- oozie 3.x 提供了更高级的抽象，能够批量运行捆绑在一起的协调应用。可以很容易的对一坨应用执行开始，暂停，继续，结束，重跑等动作。
- 课程演示版本：oozie-4.0.0-cdh5.3.6.tar.

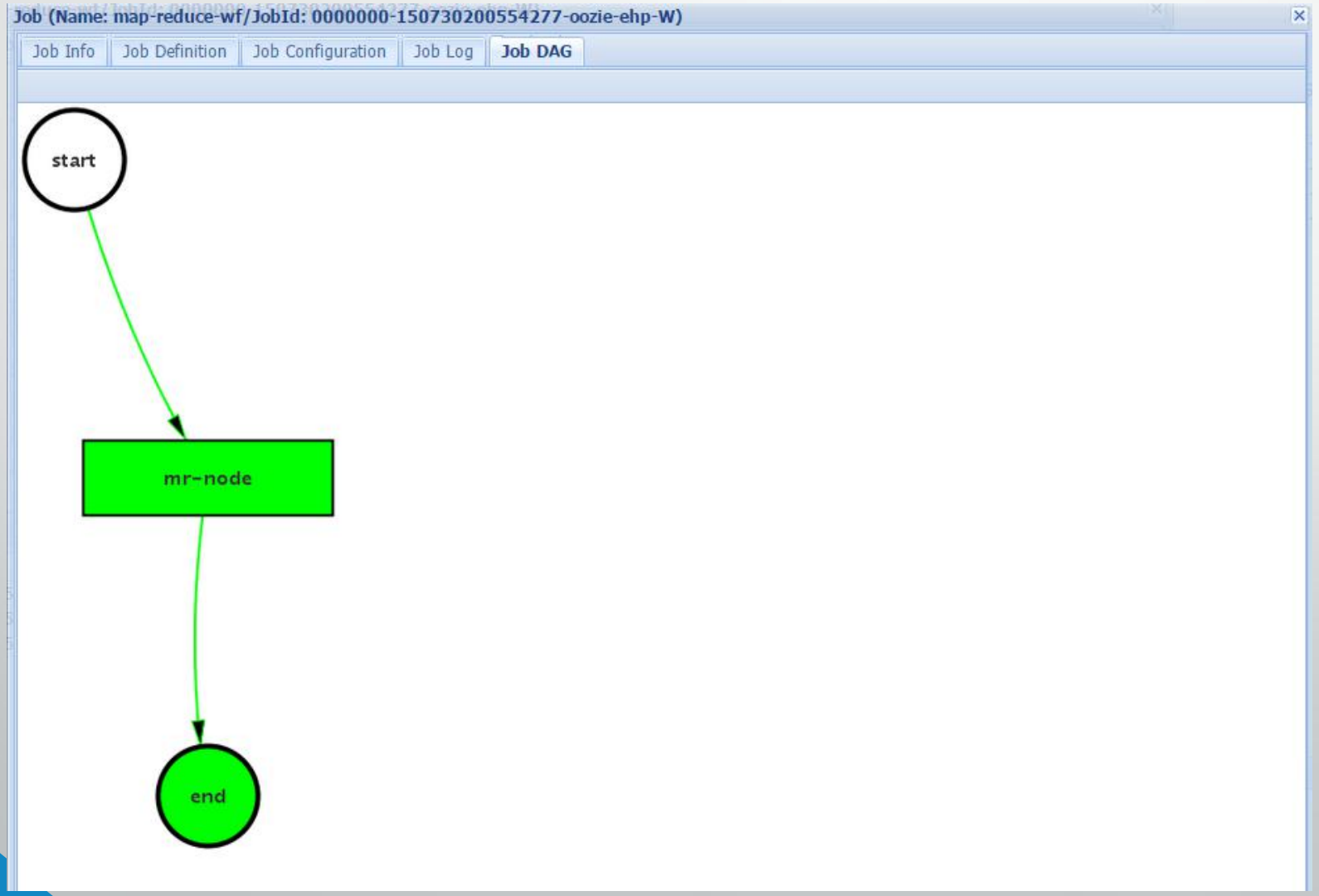
WordCount Workflow Example



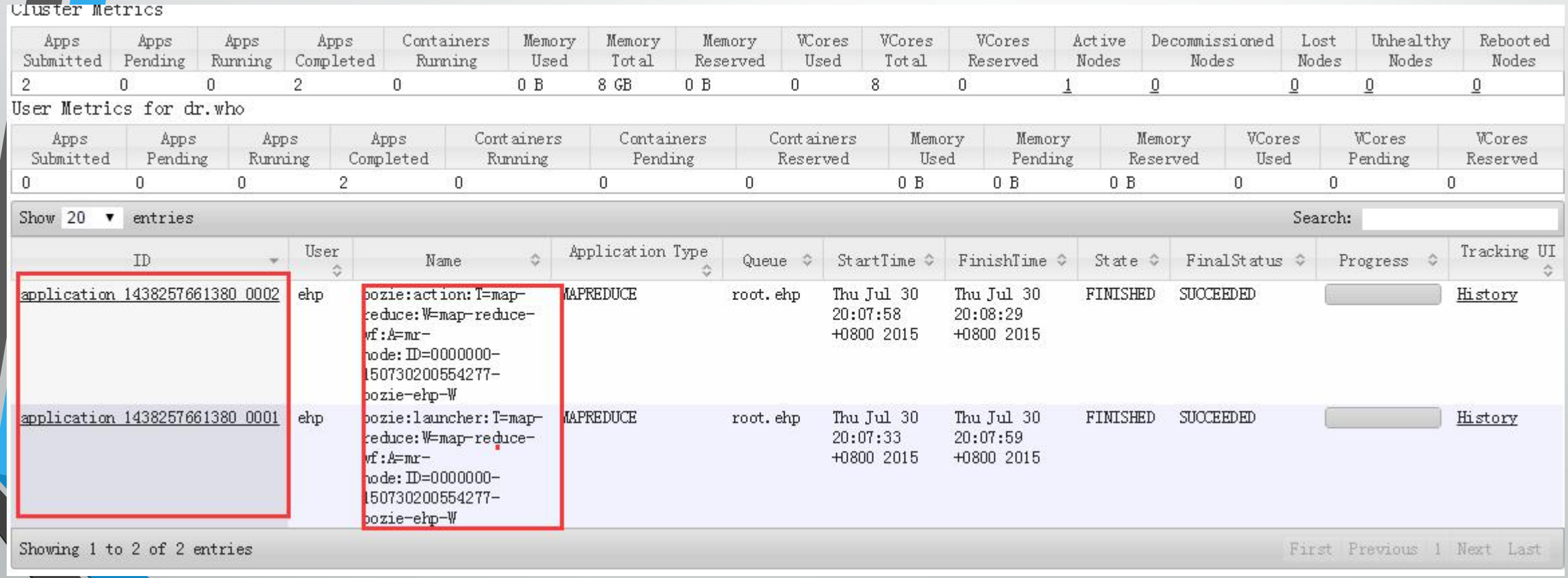
Oozie Examples



Oozie Examples



Oozie Examples



Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
2	0	0	2	0	0 B	8 GB	0 B	0	8	0	1	0	0	0	0

User Metrics for dr.who

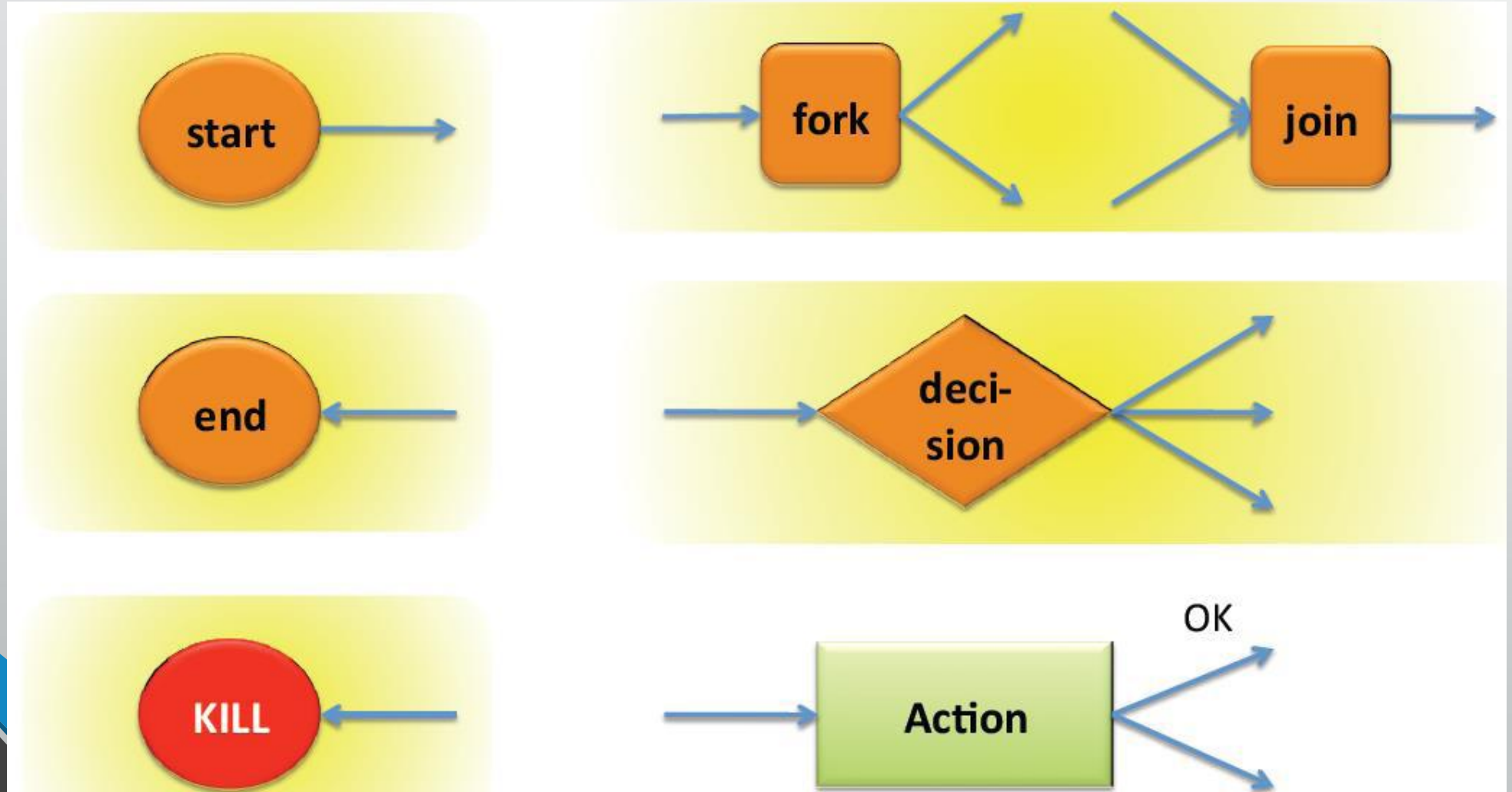
Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Containers Pending	Containers Reserved	Memory Used	Memory Pending	Memory Reserved	VCores Used	VCores Pending	VCores Reserved
0	0	0	2	0	0	0	0 B	0 B	0 B	0	0	0

Show 20 entries Search:

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI
application_1438257661380_0002	ehp	oozie:action:T=map-reduce:W=map-reduce-wf:A=mr-node:ID=0000000-150730200554277-oozie-ehp-W	MAPREDUCE	root.ehp	Thu Jul 30 20:07:58 +0800 2015	Thu Jul 30 20:08:29 +0800 2015	FINISHED	SUCCEEDED		History
application_1438257661380_0001	ehp	oozie:launcher:T=map-reduce:W=map-reduce-wf:A=mr-node:ID=0000000-150730200554277-oozie-ehp-W	MAPREDUCE	root.ehp	Thu Jul 30 20:07:33 +0800 2015	Thu Jul 30 20:07:59 +0800 2015	FINISHED	SUCCEEDED		History

Showing 1 to 2 of 2 entries First Previous 1 Next Last

Workflow Nodes



Start Control Node

The `start` node is the entry point for a workflow job, it indicates the first workflow node the workflow job must transition to.

When a workflow is started, it automatically transitions to the node specified in the `start` .

A workflow definition must have one `start` node.

Syntax:

```
<workflow-app name="[WF-DEF-NAME]" xmlns="uri:oozie:workflow:0.1">
  ...
  <start to="[NODE-NAME]"/>
  ...
</workflow-app>
```

The `to` attribute is the name of first workflow node to execute.

End Control Node

The end node is the end for a workflow job, it indicates that the workflow job has completed successfully.

When a workflow job reaches the end it finishes successfully (SUCCEEDED).

If one or more actions started by the workflow job are executing when the end node is reached, the actions will be killed. In this scenario the workflow job is still considered as successfully run.

A workflow definition must have one end node.

Syntax:

```
<workflow-app name="[WF-DEF-NAME]" xmlns="uri:oozie:workflow:0.1">
    ...
    <end name="[NODE-NAME]"/>
    ...
</workflow-app>
```

The name attribute is the name of the transition to do to end the workflow job.

Kill Control Node

The `kill` node allows a workflow job to kill itself.

When a workflow job reaches the `kill` it finishes in error (KILLED).

If one or more actions started by the workflow job are executing when the `kill` node is reached, the actions will be killed.

A workflow definition may have zero or more `kill` nodes.

Syntax:

```
<workflow-app name="[WF-DEF-NAME]" xmlns="uri:oozie:workflow:0.1">
    ...
    <kill name="[NODE-NAME]">
        <message>[MESSAGE-TO-LOG]</message>
    </kill>
    ...
</workflow-app>
```

The `name` attribute in the `kill` node is the name of the Kill action node.

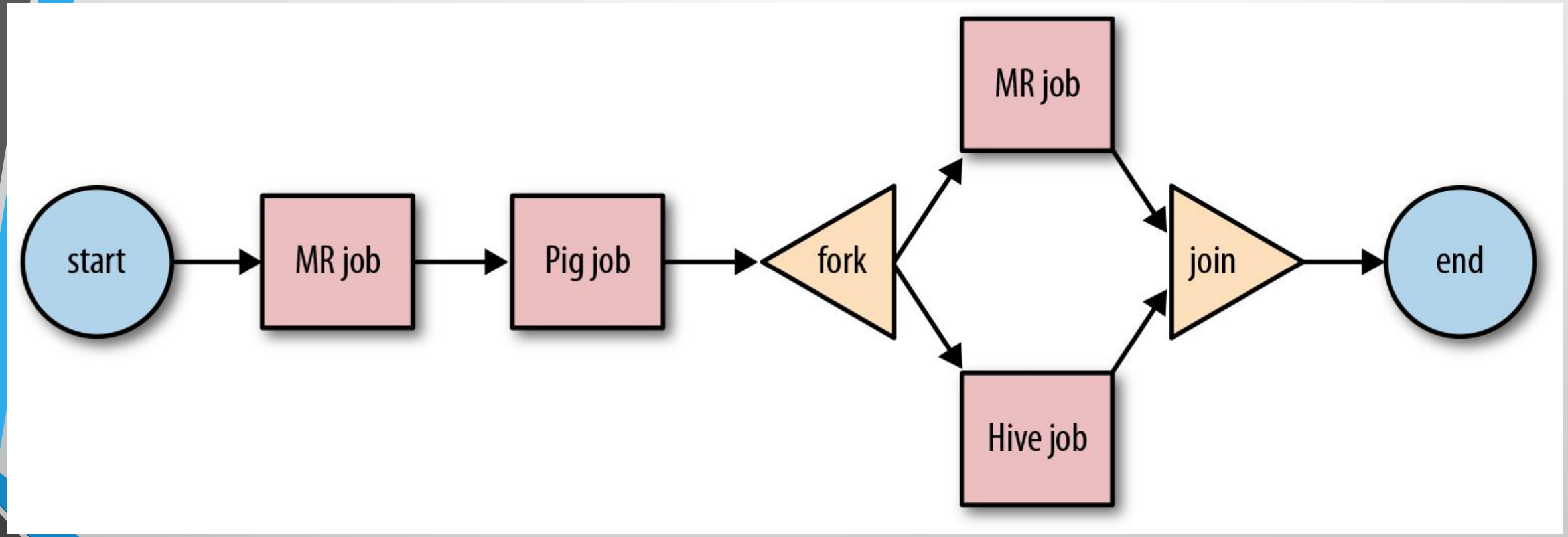
The content of the `message` element will be logged as the kill reason for the workflow job.

A `kill` node does not have transition elements because it ends the workflow job, as

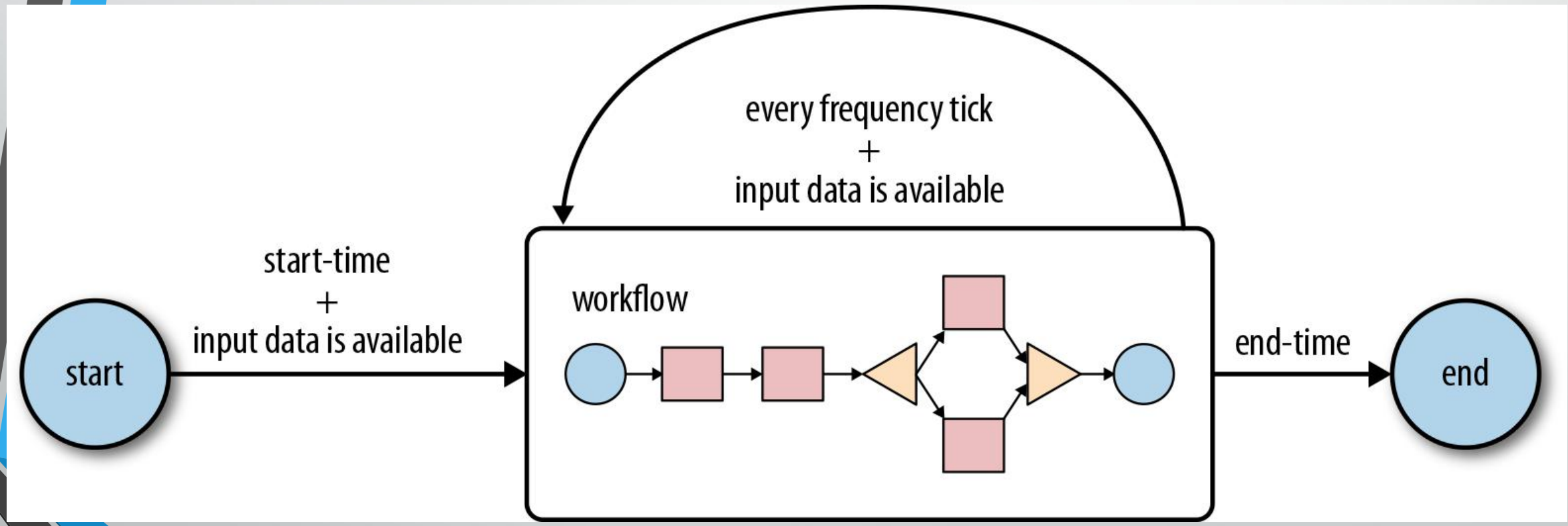
WordCount Workflow Example

```
<workflow-app name='wordcount-wf' xmlns="uri:oozie:workflow:0.1">
    <start to='wordcount' />
    <action name='wordcount'>
        <map-reduce>
            <job-tracker>${jobTracker}</job-tracker>
            <name-node>${nameNode}</name-node>
            <configuration>
                <property>
                    <name>mapred.mapper.class</name>
                    <value>org.myorg.WordCount.Map</value>
                </property>
                <property>
                    <name>mapred.reducer.class</name>
                    <value>org.myorg.WordCount.Reduce</value>
                </property>
                <property>
                    <name>mapred.input.dir</name>
                    <value>${inputDir}</value>
                </property>
                <property>
                    <name>mapred.output.dir</name>
                    <value>${outputDir}</value>
                </property>
            </configuration>
        </map-reduce>
        <ok to='end' />
        <error to='end' />
    </action>
    <kill name='kill'>
        <message>Something went wrong: ${wf:errorCode('wordcount')}</message>
    </kill/>
    <end name='end' />
</workflow-app>
```

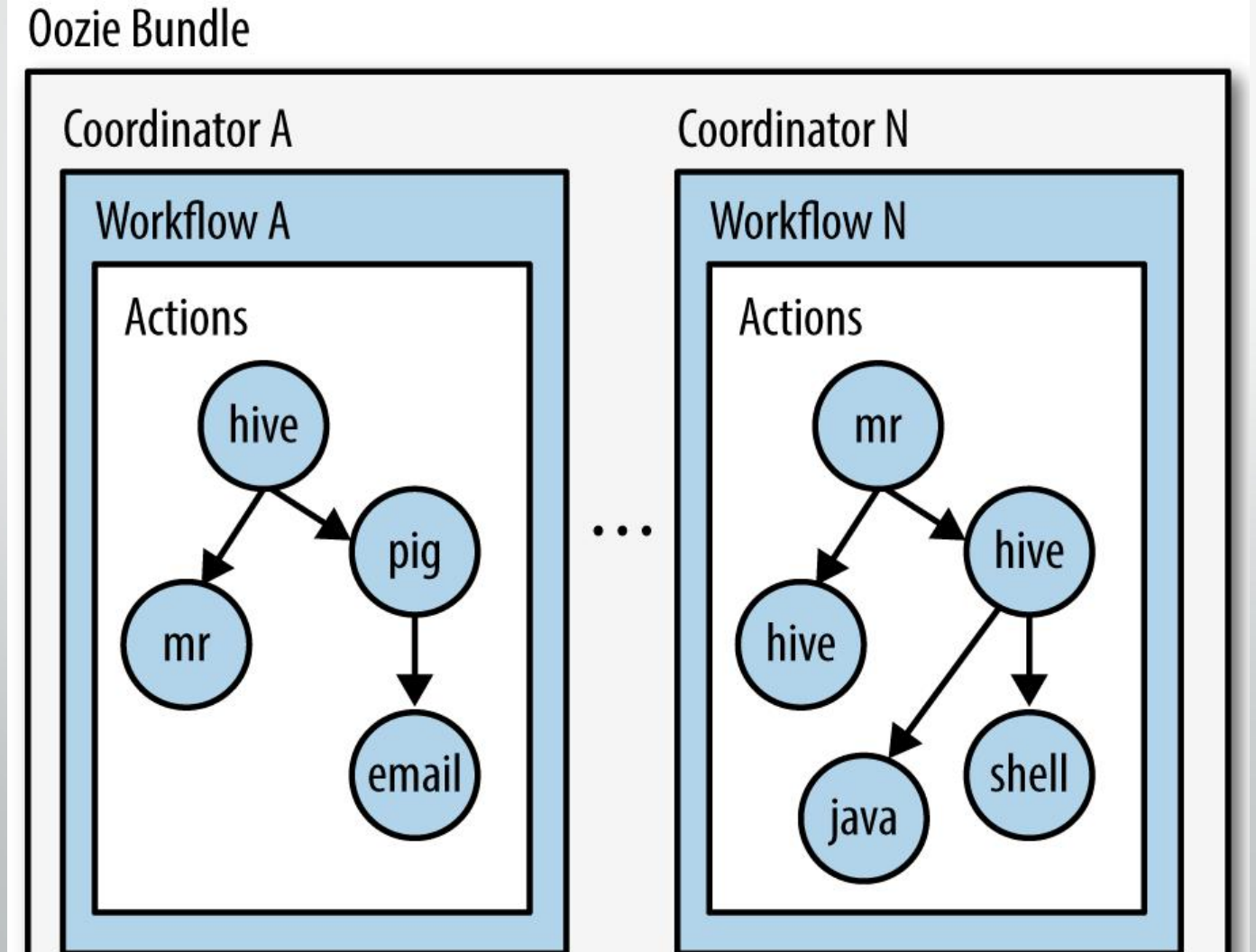
Oozie Workflow



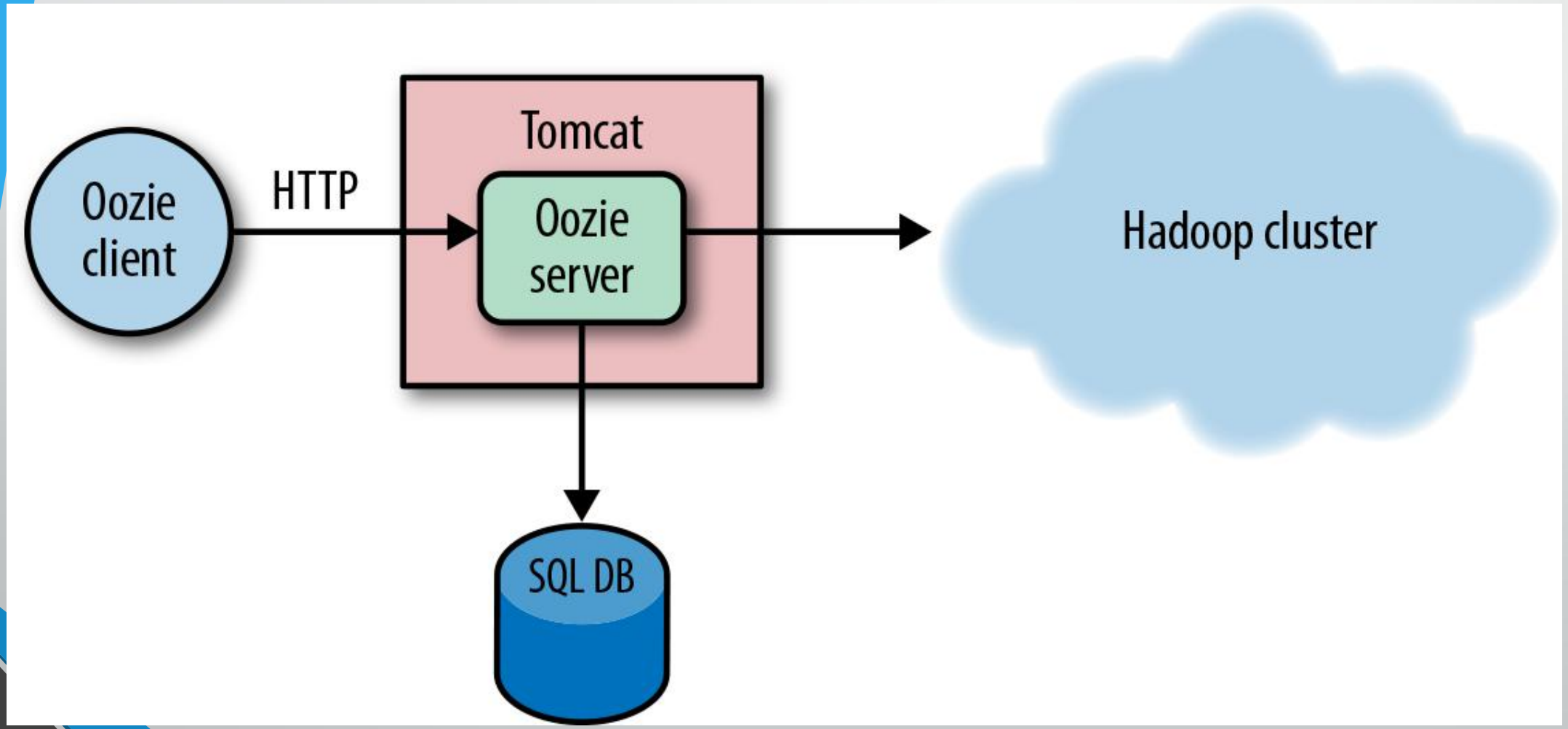
Lifecycle of an Oozie coordinator



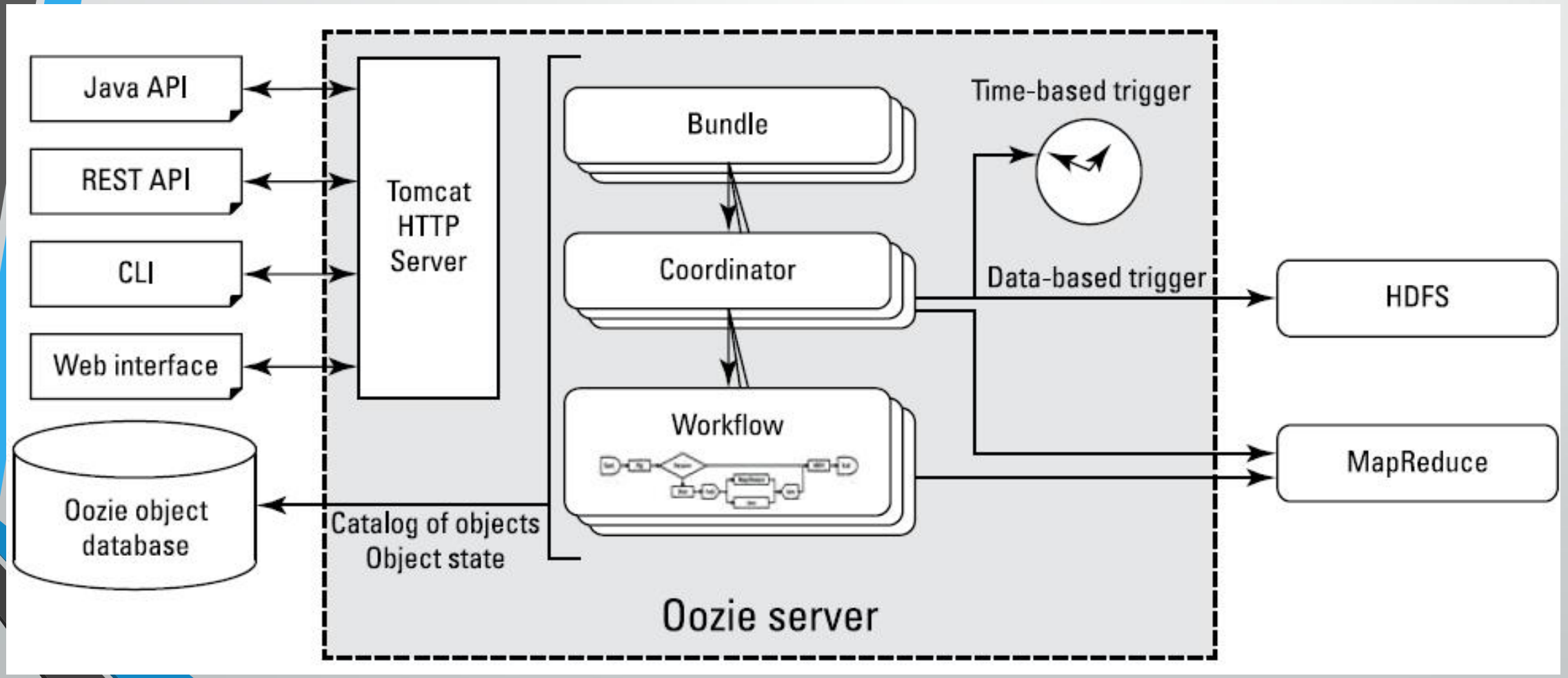
Oozie Bundle



Oozie Server Architecture



Oozie server components



课程大纲

- Oozie介绍
- Oozie部署

oozie安装配置

- 1、Hadoop已安装并启动
- 2、解压oozie软件包
- $ tar zxf /opt/softwares/oozie-4.0.0-cdh5.3.6.tar.gz -C ./

- 3、配置Hadoop的core-site.xml文件的proxyuser
- <property>
- <name>hadoop.proxyuser.beifeng.hosts</name>
- <value>*</value>
- </property>
- <property>
- <name>hadoop.proxyuser.beifeng.groups</name>
- <value>*</value>
- </property>

重启Hadoop:$ sbin/stop-dfs.sh ;sbin/start-dfs.sh

oozie安装配置

- 4、解压目录,生成libext
- $ cd /opt/modules/oozie-4.0.0-cdh5.3.6
- $ tar zxf oozie-hadooplibs-4.0.0-cdh5.3.6.tar.gz -C /opt/modules/
- 解压后目录：oozie-4.0.0-cdh5.3.6\hadooplibs

- $ mkdir libext/
- $ cp -ra hadooplibs/hadooplib-2.5.0-cdh5.3.6.oozie-4.0.0-cdh5.3.6/* libext/
- $ cp /opt/softwares/ext-2.2.zip libext/

- 5、安装Mysql
- $ mysql -uroot -p123456
- mysql> create database oozie ;

- !!!!拷贝mysql驱动jar包到libext!!!!

$ cp mysql-connector-java-5.1.27-bin.jar /opt/modules/cdh/oozie-4.0.0-cdh5.3.6/libext/

oozie安装配置

6、修改配置文件oozie-site.xml

```
<property>
    <name>oozie.service.JPAService.jdbc.driver</name>
    <value>com.mysql.jdbc.Driver</value>
</property>
<property>
    <name>oozie.service.JPAService.jdbc.url</name>
    <value>jdbc:mysql://192.168.242.128:3306/oozie</value>
</property>
<property>
    <name>oozie.service.JPAService.jdbc.username</name>
    <value>root</value>
</property>
<property>
    <name>oozie.service.JPAService.jdbc.password</name>
    <value>123456</value>
</property>

  <!--让oozie关联Hadoop-->
<property>
    <name>oozie.service.HadoopAccessorService.hadoop.configurations</name>
    <value>*=/opt/modules/cdh/hadoop-2.5.0-cdh5.3.6/etc/hadoop</value>
</property>
```

oozie安装配置

- 7、依次执行以下几个命令
  - 第一条：上传压缩包里面的文件到/user/beifeng/share/目录下面去
    - bin/oozie-setup.sh sharelib \
    - create -fs hdfs://192.168.242.128:8020 \
    - -locallib oozie-sharelib-4.0.0-cdh5.3.6-yarn.tar.gz
  - 第二条：去mysql数据库中oozie库里面创建默认的数据表
    - bin/oozie-setup.sh db \
    - create -run \
    - -sqlfile oozie.sql
  - 第三条：生成项目war包
  - bin/oozie-setup.sh prepare-war

oozie安装配置

- 8、启动oozie
- $ bin/oozied.sh start

- 9、浏览器浏览oozie界面
- http://192.168.242.128:11000/oozie

课程大纲

- Oozie介绍
- Oozie部署
- Oozie案例

案例1、oozie调度shell脚本

- oozie调度shell脚本 ：配置一个shell action

- 1、在oozie home解压案例
- $ tar zxf oozie-examples.tar.gz

- 2、配置job.properties workflow.xml
- $ mkdir oozie-apps --用来存放指定的配置
- $ cp -r examples/apps/shell/ oozie-apps/

案例1、 oozie调度shell脚本

- 1、修改job.properties和workflow.xml
- **job.properties**
- nameNode=hdfs://bigdata.beifeng.com:8020
- jobTracker=bigdata.beifeng.com:8032
- queueName=default
- examplesRoot=oozie-apps/shell

- oozie.wf.application.path=${nameNode}/user/beifeng/${examplesRoot}/
- EXEC=meminfo.sh
- **workflow.xml**
    - <exec>${EXEC}</exec>
    - <file>/user/beifeng/oozie-apps/shell/${EXEC}#${EXEC}</file>

- 2、创建脚本mem.sh
- #!/bin/bash
- /usr/bin/free -m >> /tmp/meminfo
- 3、上传工程目录到HDFS

$ bin/hdfs dfs -put /opt/modules/oozie-4.0.0-cdh5.3.6/oozie-apps/ /user/beifeng/

提交、结束任务

4、提交oozie任务
bin/oozie job -oozie http://192.168.242.128:11000/oozie -config oozie-apps/shell/job.properties -run

杀死某个oozie job
$ bin/oozie job -oozie http://192.168.242.128:11000/oozie -kill 0000003-160908105642617-oozie-beif-W

如果workflow.xml被二次修改以后,一定要记得重新上传到HDFS对应路径下面！！！

Map-Reduce Action

- A map-reduce action can be configured to **perform file system cleanup and directory creation before starting the map reduce job**.
- The workflow job will wait until the Hadoop map/reduce job completes before continuing to the next action in the workflow execution path.
- The counters of the Hadoop job and job exit status **(=FAILED=, KILLED or SUCCEEDED**) must be available to the workflow job after the Hadoop jobs ends.
- **The map-reduce action has to be configured with all the necessary Hadoop JobConf properties to run the Hadoop map/reduce job.**

Map-Reduce Action

```
<workflow-app name="foo-wf" xmlns="uri:oozie:workflow:0.1">
    ...
    <action name="myfirstHadoopJob">
        <map-reduce>
            <job-tracker>foo:8021</job-tracker>
            <name-node>bar:8020</name-node>
            <prepare>
                <delete path="hdfs://foo:8020/usr/tucu/output-data"/>
            </prepare>
            <job-xml>/myfirstjob.xml</job-xml>
            <configuration>
                <property>
                    <name>mapred.input.dir</name>
                    <value>/usr/tucu/input-data</value>
                </property>
                <property>
                    <name>mapred.output.dir</name>
                    <value>/usr/tucu/input-data</value>
                </property>
            </configuration>
        </map-reduce>
        <ok to="myNextAction"/>
        <error to="errorCleanup"/>
    </action>
    ...
</workflow-app>
```

案例2：配置一个mapreduce action

1、运行测试jar的正确性

```
$ bin/yarn jar /opt/softwares/mr-wordcount.jar /input /output
$ bin/hdfs dfs -text /output/par*
```

2、拷贝模版,并修改

```
$ cp -r map-reduce/ /opt/modules/oozie-4.0.0-cdh5.3.6/oozie-apps/
```

注意：

（1）、在hdfs上面map-reduce/下建立lib目录放countword.jar

（2）、传wc.txt文件到hdfs下面input-data/

3、上传本地mapreduce目录到HDFS

```
$ bin/hdfs dfs -put /opt/modules/oozie-4.0.0-cdh5.3.6/oozie-apps/ /user/beifeng/
```

4、提交任务

```
$ bin/oozie job -oozie http://192.168.242.128:11000/oozie -config oozie-apps/mapreduce/job.properties -run
```

Time Trigger

Time-based triggers are easy to explain and resembles the Unix cron utility. In a time-aware coordinator, a workflow is executed at fixed intervals or frequency. A user typically specifies a time trigger in the coordinator using three attributes:

Start time (ST)

Determines when to execute the first instance of the workflow

Frequency (F)

Specifies the interval for the subsequent executions

End time (ET)

Bounds the last execution start time (i.e., no new execution is permitted on or after this time)

In other words, the first execution occurs at the *ST* and subsequent executions occur at *(ST + F)*, *(ST + 2F)*, *(ST + 3F)* , and so on until the *ET* is reached.

时区

- 为什么会将地球分为不同时区呢？因为地球总是自西向东自转，东边总比西边先看到太阳，东边的时间也总比西边的早。东边时刻与西边时刻的差值不仅要以时计，而且还要以分和秒来计算。整个地球分为二十四时区，每个时区都有自己的本地时间。在国际无线电通信场合，为了统一起见，使用一个统一的时间，称为通用协调时(UTC, Universal Time Coordinated)。UTC与格林尼治平均时(GMT, Greenwich Mean Time)一样，都与英国伦敦的本地时相同。

CST：中国标准时间（China Standard Time），这个解释可能是针对RedHat Linux。

UTC：协调世界时，又称世界标准时间，简称UTC，从英文国际时间/法文协调时间”Universal Time/Temps Cordonné”而来。中国大陆、香港、澳门、台湾、蒙古国、新加坡、马来西亚、菲律宾、澳洲西部的时间与UTC的时差均为+8，也就是UTC+8。

GMT：格林尼治标准时间（旧译格林威治平均时间或格林威治标准时间；英语：Greenwich Mean Time，GMT）是指位于英国伦敦郊区的皇家格林尼治天文台的标准时间，因为本初子午线被定义在通过那里的经线。

系统时区查看

1：使用date命令查看时区

```
[root@db-server ~]# date -R
Sun, 11 Jan 2015 07:10:28 -0800
[root@db-server ~]#
```

如上RFC 2822 format所示，上面命令输出了-0800表示西八区，是美国旧金山所在的时区，下面表示我们国家的东八区（+0800）

```
[root@lnx01 ~]# date -R
Sun, 11 Jan 2015 23:06:02 +0800
```

修改时区

- 修改链接/etc/localtime的时区文件

```
[root@db-server ~]# ln /usr/share/zoneinfo/Asia/Shanghai /etc/localtime

ln: creating hard link `/etc/localtime' to `/usr/share/zoneinfo/Asia/Shanghai': File exists

[root@db-server ~]# rm /etc/localtime

rm: remove regular file `/etc/localtime'? y

[root@db-server ~]# ln -sf /usr/share/zoneinfo/Asia/Shanghai /etc/localtime

[root@db-server ~]# date -R

Mon, 12 Jan 2015 10:56:10 +0800
```

Oozie Coordinator

EL Constant	Value	Example
${coord:minutes(int n)}	*n*	${coord:minutes(45)} --> 45
${coord:hours(int n)}	*n * 60*	${coord:hours(3)} --> 180
${coord:days(int n)}	*variable*	${coord:days(2)} --> minutes in 2 full days from the current date
${coord:months(int n)}	*variable*	${coord:months(1)} --> minutes in a 1 full month from the current date
${cron syntax}	*variable*	${0, 10 15 * * 2-6} --> a job that runs every weekday at 3:00pm and 3:10pm UTC time

Cron expressions are comprised of 5 required fields. The fields respectively are described as follows:

Field name	Allowed Values	Allowed Special Characters
Minutes	0-59	, - * /
Hours	0-23	, - * /
Day-of-month	1-31	, - * ? / L W
Month	1-12 or JAN-DEC	, - * /

Oozie Coordinator

Examples:

Cron Expression	Meaning
10 9 * * *	Runs everyday at 9:10am
10,30,45 9 * * *	Runs everyday at 9:10am, 9:30am, and 9:45am
0 * 30 JAN 2-6	Runs at 0 minute of every hour on weekdays and 30th of January
0/20 9-17 * * 2-5	Runs every Mon, Tue, Wed, and Thurs at minutes 0, 20, 40 from 9am to 5pm
1 2 L-3 * *	Runs every third-to-last day of month at 2:01am
1 2 6W 3 ?	Runs on the nearest weekday to March, 6th every year at 2:01am
1 2 * 3 3#2	Runs every second Tuesday of March at 2:01am every year
0 10,13 * * MON-FRI	Runs every weekday at 10am and 1pm

Oozie Coordinator

```
<coordinator-app name="cron-coord" frequency="0/10 1/2 * * *" start="${start}" end="${end}" timezone="UTC"
                 xmlns="uri:oozie:coordinator:0.2">
        <action>
        <workflow>
            <app-path>${workflowAppUri}</app-path>
            <configuration>
                <property>
                    <name>jobTracker</name>
                    <value>${jobTracker}</value>
                </property>
                <property>
                    <name>nameNode</name>
                    <value>${nameNode}</value>
                </property>
                <property>
                    <name>queueName</name>
                    <value>${queueName}</value>
                </property>
            </configuration>
        </workflow>
    </action>
</coordinator-app>
```

案例3：使用Coordinator周期性调度shell workflow

1、配置时区

```
$ date -R
Thu, 08 Sep 2016 15:31:37 +0800
```

2、在配置文件oozie-site.xml添加如下内容：

```
<property>
  <name>oozie.processing.timezone</name>
  <value>GMT+0800</value>
</property>
```

3、修改js时区

```
$ vi oozie-server/webapps/oozie/oozie-console.js
    function getTimeZone() {
      Ext.state.Manager.setProvider(new Ext.state.CookieProvider());
      return Ext.state.Manager.get("TimezoneId","GMT+0800");
    }
```

案例3：使用Coordinator周期性调度shell workflow

- 4、重启oozie服务,并清空浏览器缓存,重新打开页面

- 5、配置coordinator周期性调度
- $ pwd
- /opt/modules/cdh/oozie-4.0.0-cdh5.3.6/oozie-apps
- $ cp -r ../examples/apps/cron ./

案例3：使用Coordinator周期性调度shell workflow

- 6、修改配置
- job.properties 里面的start开始时间需要指定一个未来时间
- coordinator.xml
- ** frequency="${coord:minutes(5)}"
- ** frequency="*/5 * * * *"
- workflow.xml 使用前面的额shell action里面的文件
- 7、测试
- $ cat mem.sh
- #!/bin/bash
- /usr/bin/free -m >> /tmp/mem.log
- /bin/date >> /tmp/mem.log

案例3：使用Coordinator周期性调度shell workflow

- 8、上传本地mapreduce目录到HDFS
- $ bin/hdfs dfs -put /opt/modules/oozie-4.0.0-cdh5.3.6/oozie-apps/ /user/beifeng/

- 9、提交
- bin/oozie job -oozie http://192.168.242.128:11000/oozie -config oozie-apps/cron/job.properties -run

Oozie 时区问题

oozie默认使用UTC时区，而服务器上可能是CST，建议统一使用GMT+0800。

不要修改oozie-default.xml，无效。在oozie-site.xml中添加：

```
1   <property>
2       <name>oozie.processing.timezone</name>
3       <value>GMT+0800</value>
4   </property>
```

可以用 `oozie info --timezones` 来查看支持的时区

使用GMT+0800后，时间不可以再使用形如 `2014-01-24T13:40Z` 的格式，要使用对应的形如 `2014-01-24T13:40+0800` 的格式

还有一点比较重要，即oozie web console的TimeZone设置要和上述一致，否则你在web console中看到的时间在感官上都是不正确的

$OOZIE_HOME/oozie-server/webapps/oozie/oozie-console.js

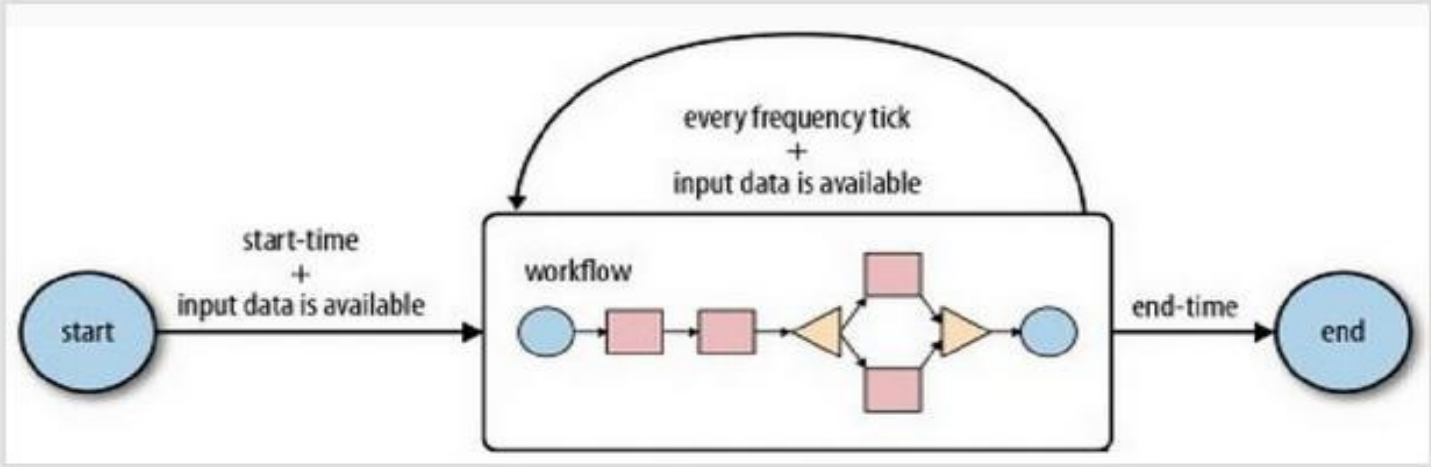
```
177 function getTimeZone() {
178     Ext.state.Manager.setProvider(new Ext.state.CookieProvider());
179     return Ext.state.Manager.get("TimezoneId","GMT+0800");
```

Oozie总结

Oozie 是用于管理 Hadoop Job 的工作流调度平台，工作流创建好之后，就需要 `Coordinator` 来进行调度了。Coordinator 进行调度依赖时间条件和数据是否可用。Coordinator 中几个关键的概念：

- ***Coordinator Action***，即是一个 Workflow Job ，Job 在满足一系列条件后开始执行；
- ***Coordinator Application***，定义了 Coordinator Action 的执行条件，包括 Coordinator Action 执行的时间频率，以及 Coordinator Action 需要被执行的开始时间和不再被执行的结束时间；
- ***Coordinator Job***，即一个 Coordinator Application 的运行实例。

一个 Coordinator Application 的定义包括开始时间、结束时间、执行频率、输入数据、输出数据和被执行的工作流。一个 Coordinator Application 在开始时间起周期性的执行工作流，如下图所示。



开始执行时，Coordinator Job 首先会检查是否输入数据能够获取到。当输入数据存在可用时，Workflow Job 处理输入数据，并产生对应的输出数据。如果输入数据不存在不可用时，Workflow Job 的执行过程推迟，直到输入数据可用。若结束时间条