

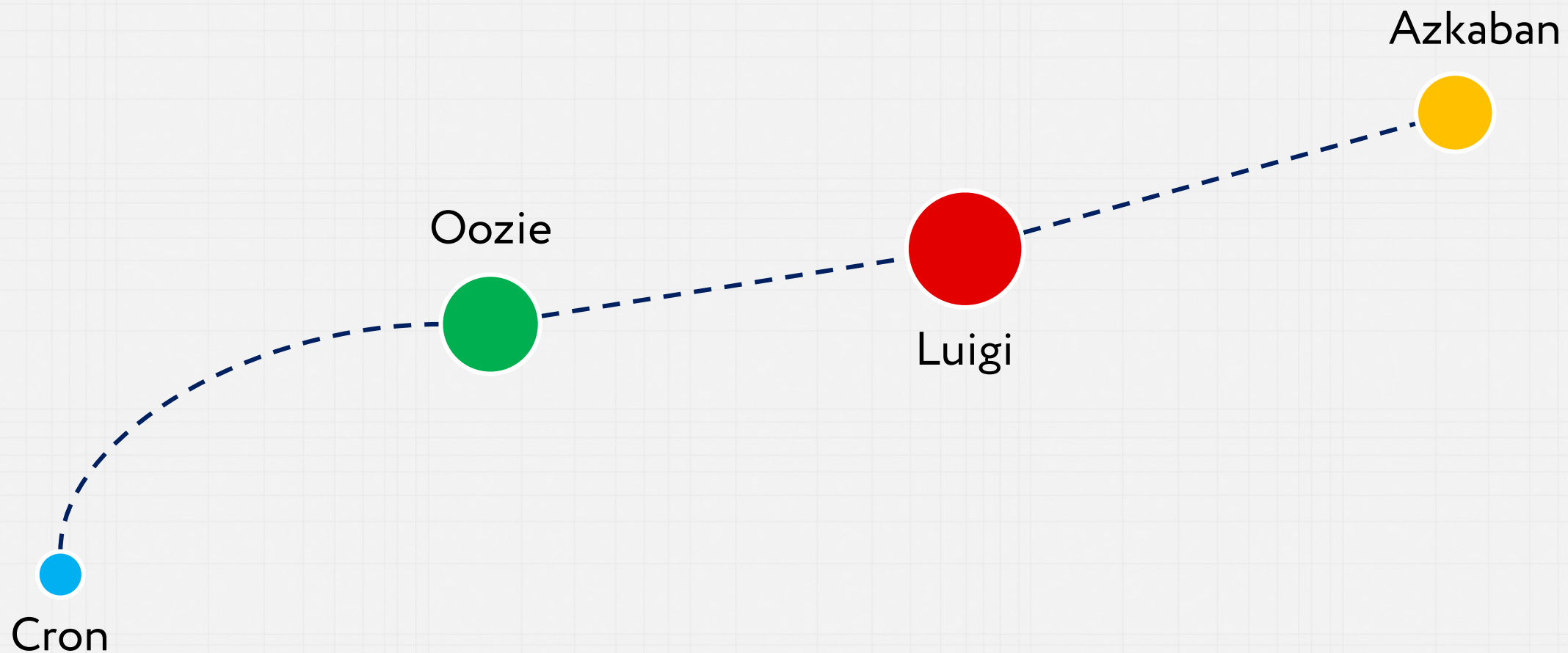


# ШЕДУЛЕРЫ

CRON, OOOIE, LUIGI, AZKABAN

АРТЕМ ПИЧУГИН, HEAD OF DATA-RELATED PROGRAMS

# План



# Cron

## Встроенная тулза в Linux

Подходит для простой логики выполнения команд в рамках одной машины.



minute (0-59)  
hour (0-23)  
day of month (1-31)  
month (1-12)  
day of week (0-6) (Sunday is 0)

\* \* \* \* \* <command to execute>



# Практика №1

Качать последние обновления из репозитория каждый день в 20:20.



# Практика №1



Качать последние обновления из репозитория каждый день в 20:20.

```
$ git config --global credential.helper 'cache --timeout=360000'
```

```
$ crontab -e
```

```
20 20 * * * cd /home/ubuntu/content-dataengineer2 | git pull >> /home/ubuntu/log.txt
```



## Практика №2

Каждые 5 минут записывать в лог состояние процесса `kibana`.



## Практика №2



Каждые 5 минут записывать в лог состояние процесса `kibana`.

```
*/5 * * * * ps aux | grep "kibana" >> /home/ubuntu/kibana-log.txt
```





## Практика №3

Каждый год 5 ноября присылать на почту сообщение  
“Winter is coming”.



# Практика №3

Каждый год 5 ноября присылать на почту сообщение  
“Winter is coming”.

```
MAILTO="apichugin@newprolab.com"
```

```
* * 5 11 * echo "Winter is coming"
```



# Полезные ссылки

- [Newbie: Intro to cron](#)
- [Scheduling Tasks with Cron Jobs](#)

# Oozie

# Oozie



1. Есть уже в дистрибутиве Hadoop.
2. Поддерживает работу с MapReduce, Pig, Hive, Sqoop.
3. Триггеры: не только время, но и события, появление данных.
4. Воркфлоу прописываются в xml.
5. Launcher — это map-only джоба, которая может запускать другие MR-джобы.

Есть 2 режима работы:

- Oozie Workflow джобы — это DAG'и, в которых просто прописана последовательность действий,
- Oozie Coordinator джобы — повторяющиеся Workflow джобы с триггерами по времени и появлению данных.



# Oozie: практика

Создадим простенький shell-скрипт и добавим его в oozie.



```
$ nano ~/sample.sh
```

```
#!/bin/bash
```

```
echo "`date` hi" >> /tmp/output
```

```
$ hdfs dfs -put sample.sh
```

```
$ nano job.properties
```

```
nameNode=hdfs://<namenode-hostname>:8020
```

```
jobTracker=<resource-manager-hostname>:8050
```

```
queueName=default
```

```
examplesRoot=examples
```

```
oozie.wf.application.path=${nameNode}/user/${user.name}
```



```
$ nano workflow.xml
```

```
<workflow-app xmlns="uri:oozie:workflow:0.3" name="shell-wf">
  <start to="shell-node"/>
  <action name="shell-node">
    <shell xmlns="uri:oozie:shell-action:0.1">
      <job-tracker>${jobTracker}</job-tracker>
      <name-node>${nameNode}</name-node>
      <configuration>
        <property><name>mapred.job.queue.name</name>
          <value>${queueName}</value></property>
      </configuration>
      <exec>sample.sh</exec>
      <file>/user/ubuntu/sample.sh</file>
    </shell>
    <ok to="end"/>
    <error to="fail"/>
  </action>

  <kill name="fail">
    <message>Shell action failed, error message[${wf:errorMessage(wf:lastErrorNode())}]</message>
  </kill>
  <end name="end"/>
</workflow-app>
```

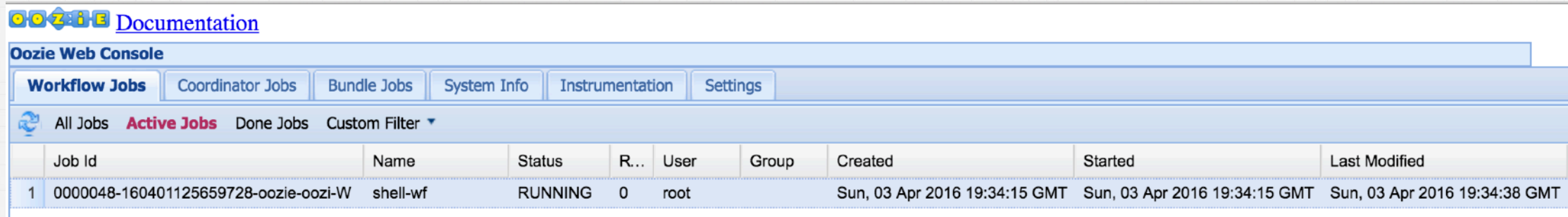
```
$ hdfs dfs -put workflow.xml
```

```
$ scp job.properties ubuntu@<server with oozie>
```

```
$ ssh <server with oozie>
```

```
$ oozie job -oozie http://localhost:11000/oozie -config job.properties -run
```

```
http://<oozie-host>:11000/oozie
```



The screenshot shows the Oozie Web Console interface. At the top, there's a navigation bar with tabs for 'Workflow Jobs', 'Coordinator Jobs', 'Bundle Jobs', 'System Info', 'Instrumentation', and 'Settings'. Below this, there's a section for 'Active Jobs' with a table listing job details. The table has columns for Job Id, Name, Status, R..., User, Group, Created, Started, and Last Modified. One job is listed with Job Id '0000048-160401125659728-oozie-oozi-W', Name 'shell-wf', Status 'RUNNING', R... '0', User 'root', Group, Created 'Sun, 03 Apr 2016 19:34:15 GMT', Started 'Sun, 03 Apr 2016 19:34:15 GMT', and Last Modified 'Sun, 03 Apr 2016 19:34:38 GMT'.

Job Id	Name	Status	R...	User	Group	Created	Started	Last Modified
1 0000048-160401125659728-oozie-oozi-W	shell-wf	RUNNING	0	root		Sun, 03 Apr 2016 19:34:15 GMT	Sun, 03 Apr 2016 19:34:15 GMT	Sun, 03 Apr 2016 19:34:38 GMT

```
$ cat /tmp/output
```

```
Sun Apr 21 13:44:52 UTC 2018 hi
```

The screenshot displays the Hue Oozie Dashboard interface. The top navigation bar includes links for Query Editors, Data Browsers, Workflows, Search, Security, File Browser, Job Browser, and a user profile for 'romain'. The main navigation bar shows 'Oozie Dashboard' with sub-tabs for Workflows, Coordinators, Bundles, SLA, and Oozie. The left sidebar contains sections for WORKFLOW (My Workflow), SUBMITTER (romain), STATUS (SUCCEEDED), PROGRESS (100%), ID (0000008-150319151313008-oozie-oozi-W), VARIABLES (oozie.wf.application.path), and MANAGE (Rerun button).

The main content area shows the 'Workflow My\_Workflow' with tabs for Graph, Actions, Details, Configuration, Log, and Definition. The 'Definition' tab is active, displaying the XML definition of the workflow:

```
1 <workflow-app name="My_Workflow" xmlns="uri:oozie:workflow:0.5">
2   <start to="hive2-099c"/>
3   <kill name="Kill">
4     <message>Action failed, error message[${wf:errorMessage(wf:lastErrorNode())}]</message>
5   </kill>
6   <action name="hive2-099c">
7     <hive2 xmlns="uri:oozie:hive2-action:0.1">
8       <job-tracker>${jobTracker}</job-tracker>
9       <name-node>${nameNode}</name-node>
10      <jdbc-url>jdbc:hive2://localhost:10000/default</jdbc-url>
11      <script>/user/romain/demo/oozie-hive2/select.sql</script>
12      <param>fields=*</param>
13      <param>tablename=sample_07</param>
14      <param>n=10</param>
15    </hive2>
16    <ok to="End"/>
17    <error to="Kill"/>
18  </action>
19  <end name="End"/>
20 </workflow-app>
21
```

# Luigi



1. Придуман компанией Spotify.
2. Написан на Python.
3. Может запускать и Hadoop-джобы, и CLI-тулзы.
4. Каждый task — это класс, внутри которого надо прописать определенные методы.

1. **requires()** — возвращает от каких тасков зависит этот таск.
2. **output()** — возвращает объекты или файлы, которые возникают в итоге.
3. **run()** — вся логика исполнения таска.

```
$ pip install tornado
```

```
$ pip install luigi
```



# Hello, World

```
import luigi

class HelloWorld(luigi.Task):
    def requires(self):
        return None
    def output(self):
        return luigi.LocalTarget('helloworld.txt')
    def run(self):
        with self.output().open('w') as outfile:
            outfile.write('Hello World!\n')

if __name__ == '__main__':
    luigi.run()
```

```
$ python luigitutorial.py --local-scheduler HelloWorld
```



Добавим еще один таск.

```
class NameSubstituter(luigi.Task):  
    name = luigi.Parameter()  
  
    def requires(self):  
        return HelloWorld()  
  
    def output(self):  
        return luigi.LocalTarget(self.input().path + '.name_' + self.name)  
  
    def run(self):  
        with self.input().open() as infile, self.output().open('w') as outfile:  
            text = infile.read()  
            text = text.replace('World', self.name)  
            outfile.write(text)
```

```
$ python luigitutorial.py --local-scheduler NameSubstituter --name artem
```



# Помониторим



В отдельном окне:

```
$ luigid
```

```
http://<master>:8082
```

В обычном окне:

```
$ python luigitutorial.py --scheduler-host 0.0.0.0 NameSubstituter --name YourName
```

## Luigi Task Status



Task List

Dependency Graph

Workers

Resources

Running

K FAMILIES

HelloWorld

NameSubstituter



PENDING TASKS

0



RUNNING TASKS

0



BATCH RUNNING TASKS

0



DONE TASKS

2



FAILED TASKS

0



UPSTREAM FAILURE

0



DISABLED TASKS

0



UPSTREAM DISABLED

0

Displaying **RUNNING, DONE**, tasks .

Show 10 entries

Filter table:

Filter on Server ☐

	Name	Details	Priority	Time	Actions
✓ DONE	NameSubstituter	name=YourName	0	4/21/2018, 11:46:47 PM	
✓ DONE	HelloWorld		0	4/21/2018, 11:44:35 PM	

Showing 1 to 2 of 2 entries

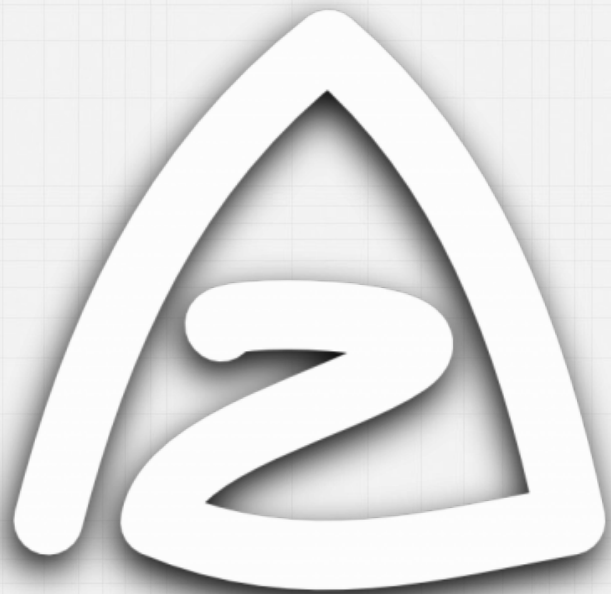
Previous

1

Next

# Azkaban

# Azkaban



1. Придуман компанией LinkedIn.
2. Написан на Java.
3. Реагирует только по времени.
4. Вроде бы уже не развивается.
5. Хорошо работает с Voldemort и Hadoop.



```
$ wget https://github.com/azkaban/azkaban/archive/3.46.0.tar.gz
$ tar -xvf 3.46.0.tar.gz
$ cd azkaban-3.46.0
$ ./gradlew build
$ ./gradlew clean
$ ./gradlew installDist
$ ./gradlew test
$ cd azkaban-solo-server/build/install/azkaban-solo-server
$ ./bin/start-solo.sh
```

`http://<master>:8081/index/`

Нужно сделать zip из следующего содержимого:

```
testflow/  
  testflow.job  
  myflow/  
    test1.job  
    test2.job
```

```
# testflow.job  
type=flow  
flow.name=test2
```

```
# test1.job  
type=command  
command=echo "!!! Hello World !!!"
```

```
# test2.job  
type=command  
dependencies=test1  
command=echo "Welcome to Azkaban !"
```

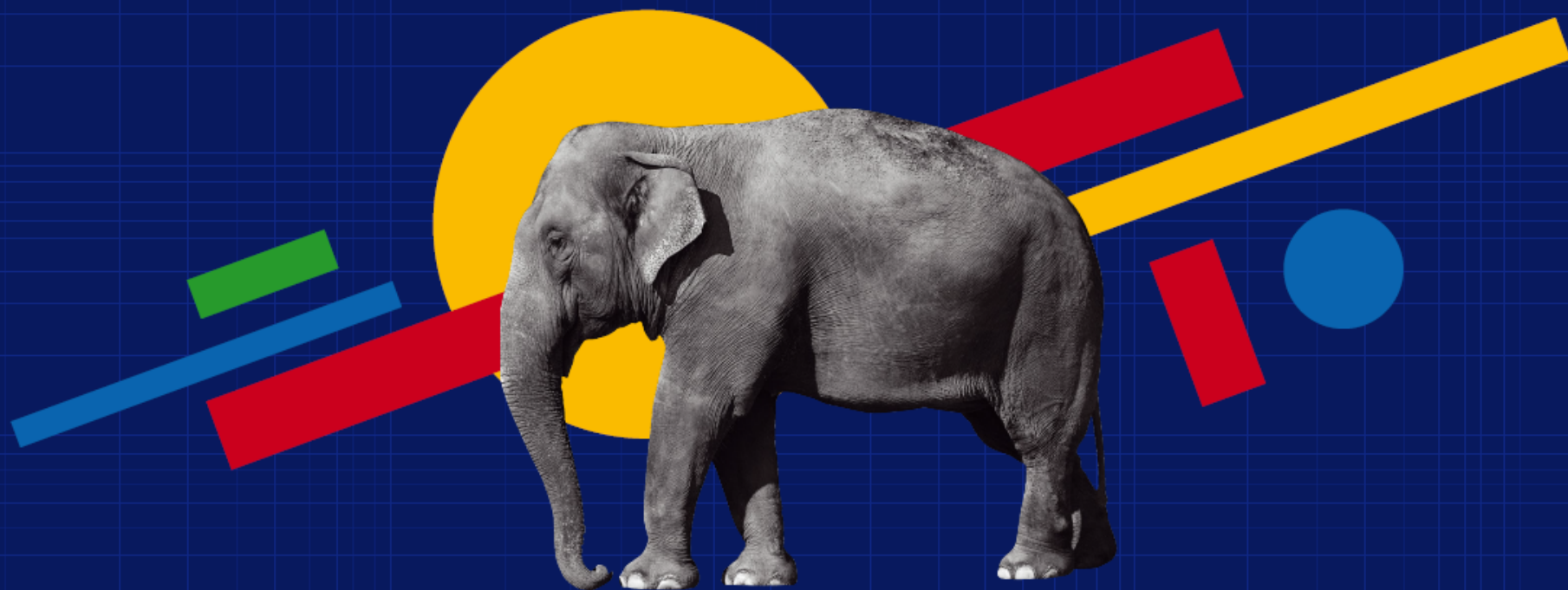


# Ссылки на обзоры и сравнения



- [Workflow Engines for Hadoop](#)
- [Workflow Engine Comparison \(First Impressions\)](#)
- [Open Source Data Pipeline – Luigi vs Azkaban vs Oozie vs Airflow](#)
- [Data Pipelines - Airflow vs Pinball vs Luigi](#)

Oozie seems like software written for you, by someone who does not like you.  
Airflow seems like software written by a friend, who would like you to be happy.



# BIG DATA IS LOVE

NEWPROLAB.COM