

# HandsOn – **Apache Airflow**

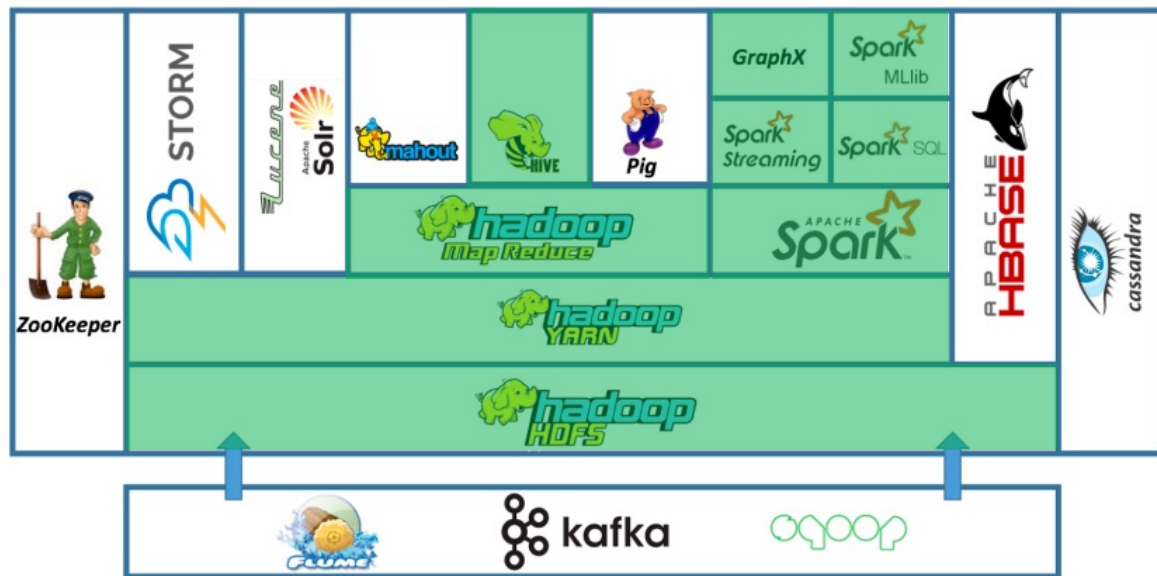
A quick Introduction to ETL Workflow with  
Apache Airflow



# The Hadoop Ecosystem



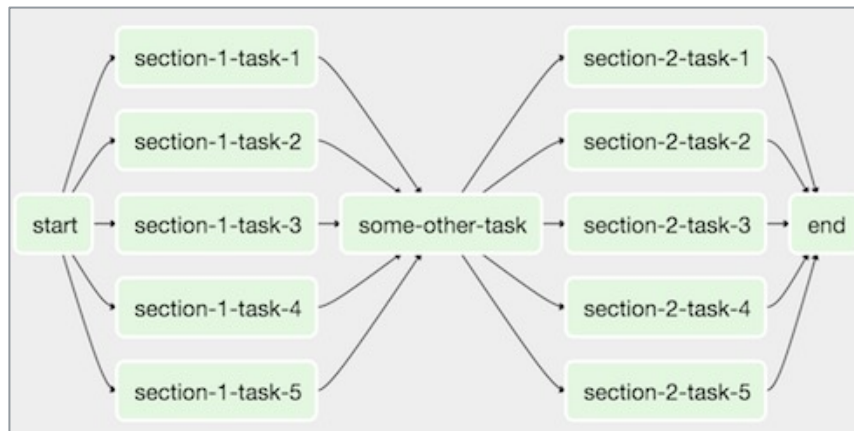
Today's  
(exercise) focus



# Airflow

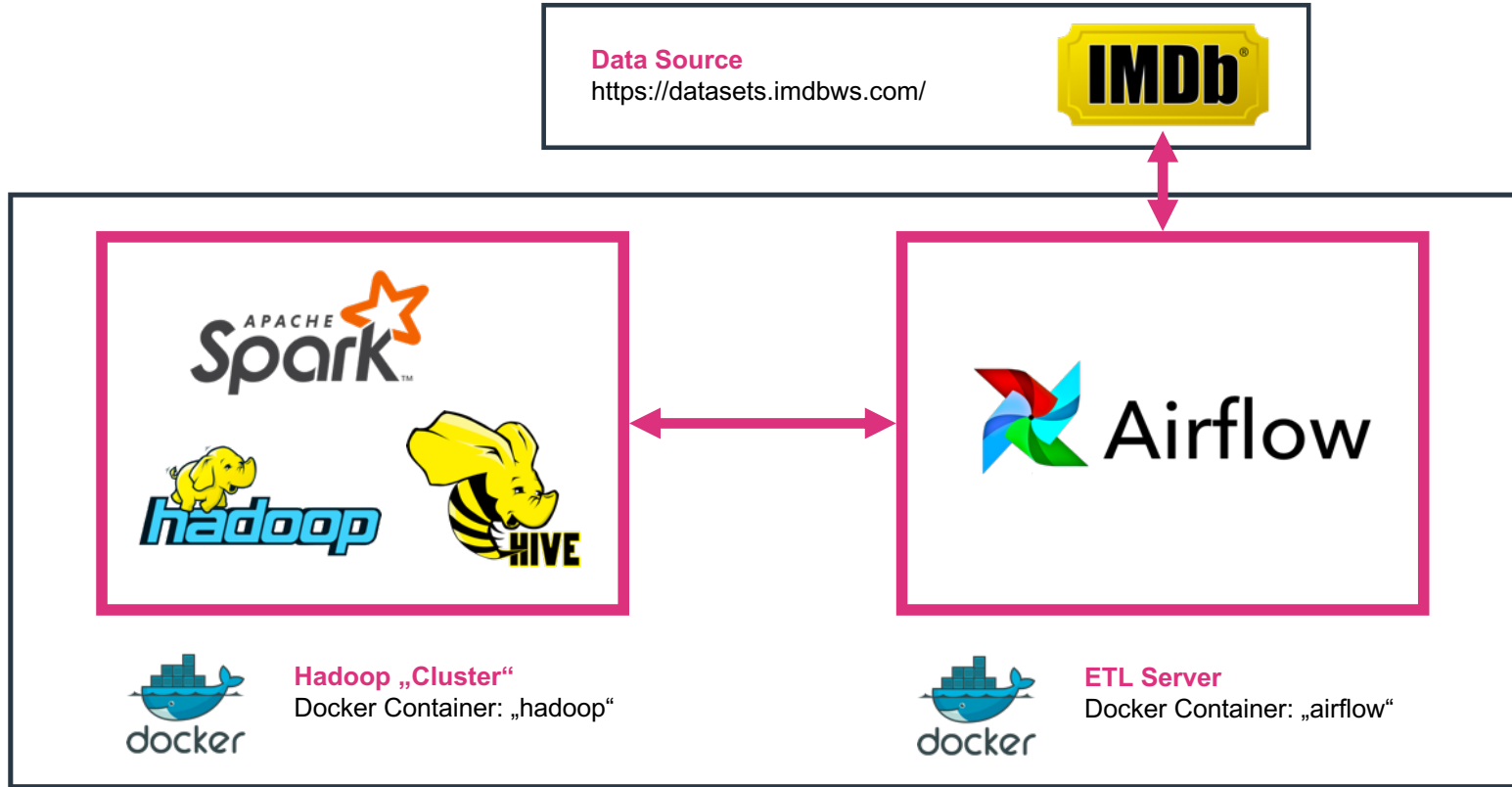


- Apache Open Source Project
- Model Task (e.g. ETL) Workflows
- Python Code Base
- Scheduling, Queues and Pools
- Cluster Ready
- Web UI
- **DAG = Directed Acyclic Graph**



→ a collection of all the tasks you want to run, organized in a way that reflects their relationships and dependencies.

# What do we want to do?



**Gcloud Server** (Connected to via SSH)



# Remove Previously created Docker Container

## 1. Stop and Remove Images:

(This will delete all files you created within previous exercise.  
Save them somewhere outside the docker container, if you haven't done yet.)

```
docker stop hadoop  
docker rm hadoop
```

```
docker stop pentaho  
docker rm pentaho
```

# Start Hadoop/Hive/Spark Docker Container

## 1. Pull Docker Image:

```
docker pull marcelmittelstaedt/spark_base:latest
```

## 2. Start Docker Image:

```
docker run -dit --name hadoop \  
    -p 8088:8088 -p 9870:9870 -p 9864:9864 -p 10000:10000 \  
    -p 8032:8032 -p 8030:8030 -p 8031:8031 -p 9000:9000 \  
    -p 8888:8888 --net bigdatanet \  
    marcelmittelstaedt/spark_base:latest
```

## 3. Wait till first Container Initialization finished:

```
docker logs hadoop  
  
[...]  
Stopping nodemanagers  
Stopping resourcemanager  
Container Startup finished.
```

# Start Hadoop/Hive/Spark Docker Container

## 4. Get into Docker container:

```
docker exec -it hadoop bash
```

## 5. Switch to hadoop user:

```
sudo su hadoop
```

```
cd
```

## 6. Start Hadoop Cluster:

```
start-all.sh
```

## 7. Start HiveServer2:

```
hiveserver2
```

# Start Hadoop/Hive/Spark Docker Container

## 8. Start HiveServer2:

```
hive/bin/hiveserver2
```

```
2018-10-02 16:19:08: Starting HiveServer2
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/hadoop/hive/lib/log4j-slf4j-impl-2.10.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hadoop/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Hive Session ID = b8d1efb3-fc8c-4ec8-bdf0-6a9a41e2ddaa
Hive Session ID = 32503981-a5fd-497e-b887-faf3ec1e686e
Hive Session ID = 00f7eab4-5a29-4ce4-ad97-e90904d9206f
Hive Session ID = 100e54c5-14c6-4acc-b398-040152b08ebf
[...]
```



# Start ETL (Airflow) Docker Container

## 1. Pull Docker Image:

```
docker pull marcelmittelstaedt/airflow:latest
```

## 2. Start Docker Image:

```
docker run -dit --name airflow \  
    -p 8080:8080 \  
    --net bigdatanet \  
    marcelmittelstaedt/airflow:latest
```

## 3. Wait till first Container Initialization finished:

```
docker logs airflow  
  
[...]  
Successfully added `conn_id`=spark : spark://:@yarn:  
  
Container Startup finished.
```

# Start ETL (Airflow) Docker Container

## 4. Get into Docker container:

```
docker exec -it airflow bash
```

## 5. Switch to airflow user:

```
sudo su airflow
```

```
cd
```

# Exercises Preparation II

Airflow First Steps/Dag



# Spoon Interface

Airflow Landing Page <http://xxx.xxx.xxx.xxx:8080/admin/>

Quick Links to e.g.:

- Trigger Dag
- View Dag
- View Execution Logs
- View Code
- ...

DAGs

The screenshot shows the Airflow web interface. At the top is a navigation bar with links for DAGs, Data Profiling, Browse, Admin, Docs, and About. The main heading is 'DAGs'. Below it is a search bar. A table lists the DAGs. The first DAG is 'IMDb' with a status of 'On', a schedule of '56 18 \* \* \*', owner 'airflow', and a last run time of '2019-11-02 18:56'. The table also shows 'Recent Tasks' and 'DAG Runs'. To the right of the table is a 'Links' section with various icons. At the bottom right, it says 'Showing 1 to 1 of 1 entries'. There are five callout boxes: 'DAGs' points to the table header; 'DAG scheduled? (on/off)' points to the 'On' status; 'Schedule Time (Cron)' points to the '56 18 \* \* \*' schedule; 'Recent Executions' points to the 'Recent Tasks' column; and 'Last Execution of DAG' points to the 'Last Run' column.

	DAG	Schedule	Owner	Recent Tasks	Last Run	DAG Runs	Links
	IMDb	56 18 * * *	airflow		2019-11-02 18:56		

Showing 1 to 1 of 1 entries

DAG scheduled?  
(on/off)

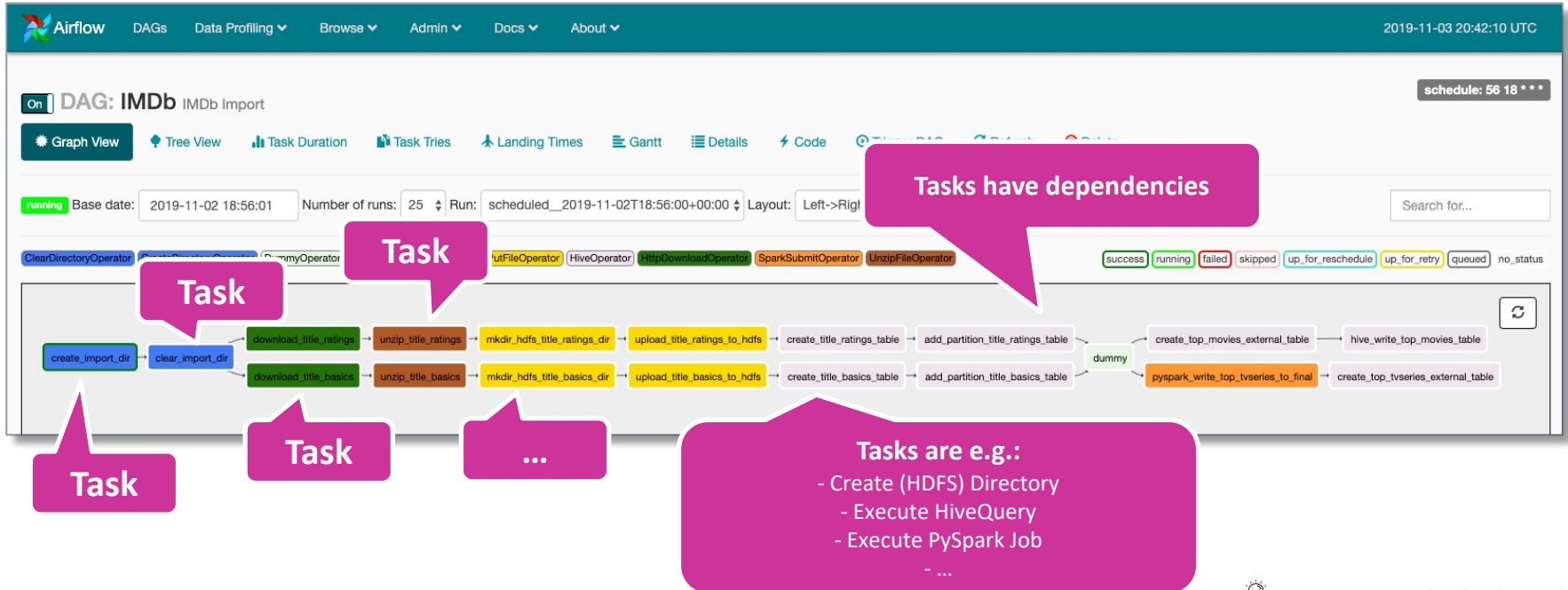
Schedule Time  
(Cron)

Recent Executions

Last Execution of DAG

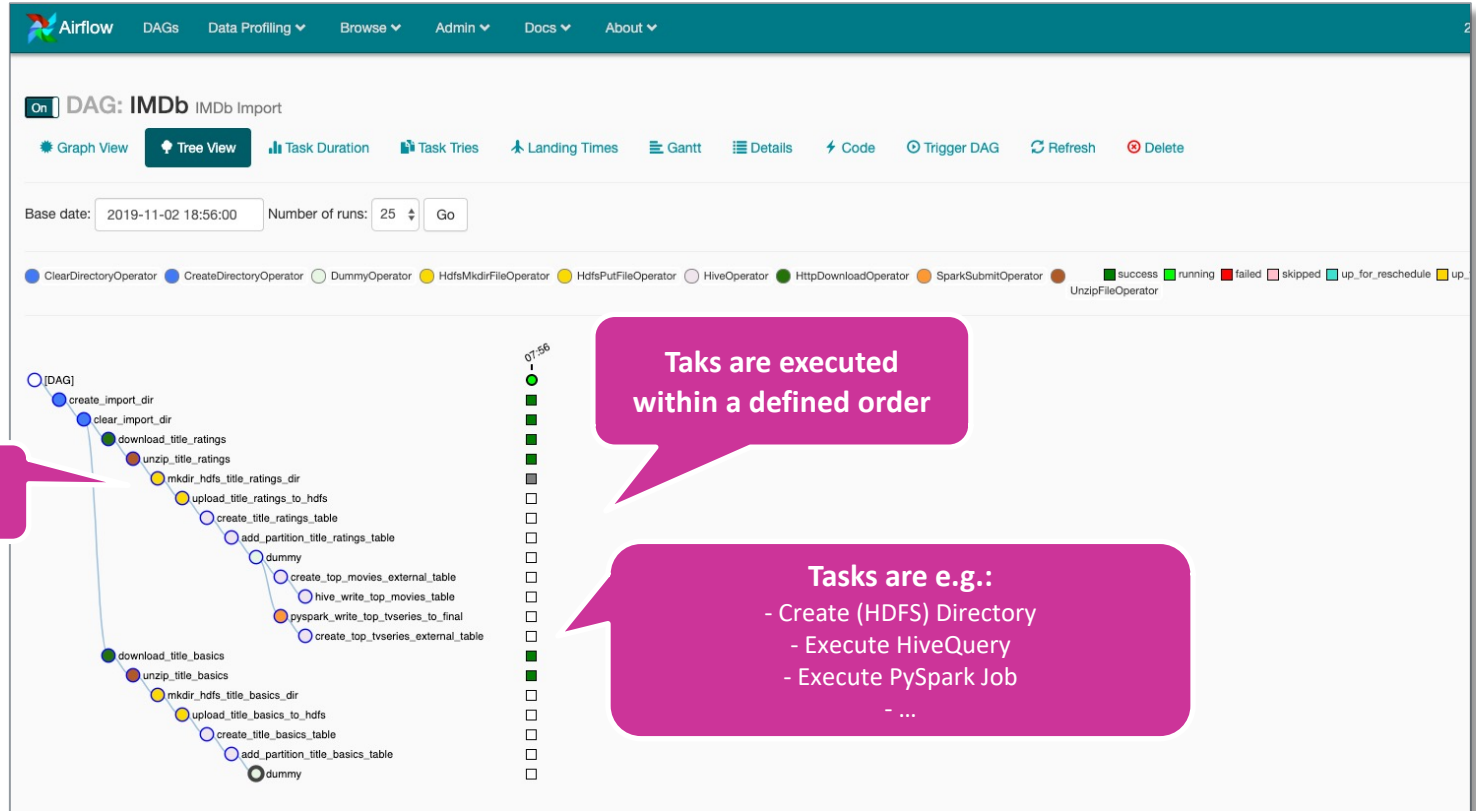
# Spoon Interface

## Graph View:



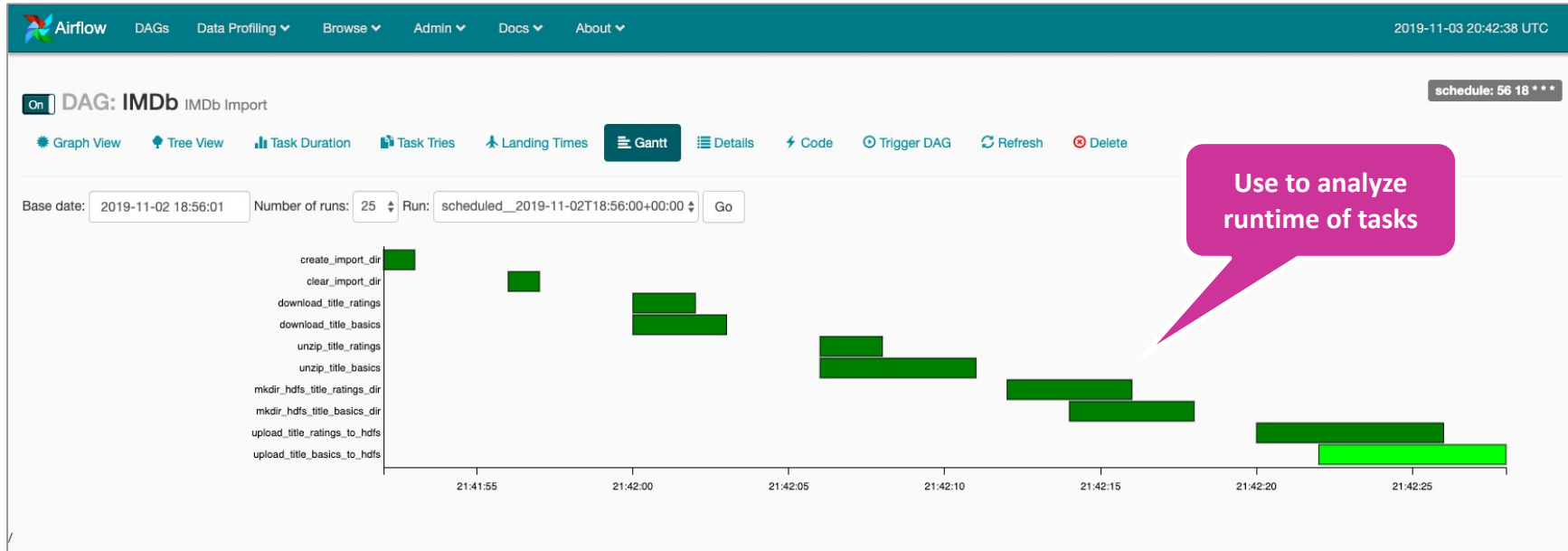
# Spoon Interface

## Tree View:



# Spoon Interface

## Gantt View:





# Spoon Interface

## Logs View:

The screenshot displays the Apache Airflow web interface. At the top, there's a navigation bar with links for DAGs, Data Profiling, Browse, Admin, Docs, and About. The current view is for a DAG named 'IMDb IMDb Import'. Below the navigation bar, there are tabs for Graph View, Tree View, Task Duration, Task Tries, Landing Times, Gantt, Details, Code, Trigger DAG, Refresh, and Delete. The 'Task Instance: unzip\_title\_ratings' is selected, with a date range of '2019-11-02 18:56:00'. Below this, there are buttons for Task Instance Details, Rendered Template, Log, and XCom. The 'Log by attempts' section shows a table with one attempt. The log content is displayed below the table, showing various log messages from the task execution, including dependencies, task execution, and file operations.

On DAG: IMDb IMDb Import schedule: 56 18 \*\*\*

Graph View Tree View Task Duration Task Tries Landing Times Gantt Details Code Trigger DAG Refresh Delete

Task Instance: unzip\_title\_ratings 2019-11-02 18:56:00

Task Instance Details Rendered Template Log XCom

Log by attempts

1 Toggle wrap Jump to end

```
*** Reading local file: /home/airflow/airflow/logs/IMDb/unzip_title_ratings/2019-11-02T18:56:00+00:00/1.log
[2019-11-03 20:42:06,214] {taskinstance.py:630} INFO - Dependencies all met for <TaskInstance: IMDb.unzip_title_ratings 2019-11-02T18:56:00+00:00 [queued]>
[2019-11-03 20:42:06,249] {taskinstance.py:630} INFO - Dependencies all met for <TaskInstance: IMDb.unzip_title_ratings 2019-11-02T18:56:00+00:00 [queued]>
[2019-11-03 20:42:06,249] {taskinstance.py:841} INFO -

[2019-11-03 20:42:06,249] {taskinstance.py:842} INFO - Starting attempt 1 of 1
[2019-11-03 20:42:06,249] {taskinstance.py:843} INFO -

-----
[2019-11-03 20:42:06,270] {taskinstance.py:862} INFO - Executing <Task(UnzipFileOperator): unzip_title_ratings> on 2019-11-02T18:56:00+00:00
[2019-11-03 20:42:06,271] {base_task_runner.py:133} INFO - Running: ['airflow', 'run', 'IMDb', 'unzip_title_ratings', '2019-11-02T18:56:00+00:00', '--job_id', '6', '--pool', 'default_pool', '--raw', '-sd', 'DAGS_FOLDER']
[2019-11-03 20:42:07,151] {base_task_runner.py:115} INFO - Job 6: Subtask unzip_title_ratings [2019-11-03 20:42:07,151] {settings.py:252} INFO - settings.configure_orm(): Using pool settings. pool_size=5, max_overflow=5
[2019-11-03 20:42:07,185] {base_task_runner.py:115} INFO - Job 6: Subtask unzip_title_ratings /home/airflow/.local/lib/python3.6/site-packages/psycpg2/__init__.py:144: UserWarning: The psycpg2 wheel package will
[2019-11-03 20:42:07,185] {base_task_runner.py:115} INFO - Job 6: Subtask unzip_title_ratings
[2019-11-03 20:42:07,944] {base_task_runner.py:115} INFO - Job 6: Subtask unzip_title_ratings [2019-11-03 20:42:07,942] {__init__.py:51} INFO - Using executor LocalExecutor
[2019-11-03 20:42:07,944] {base_task_runner.py:115} INFO - Job 6: Subtask unzip_title_ratings [2019-11-03 20:42:07,944] {dagbag.py:92} INFO - Filling up the DagBag from /home/airflow/airflow/dags/imdb.py
[2019-11-03 20:42:07,994] {base_task_runner.py:115} INFO - Job 6: Subtask unzip_title_ratings [2019-11-03 20:42:07,994] {cli.py:545} INFO - Running <TaskInstance: IMDb.unzip_title_ratings 2019-11-02T18:56:00+00:00 [queued]>
[2019-11-03 20:42:08,029] {zip_file_operator.py:33} INFO - UnzipFileOperator execution started.
[2019-11-03 20:42:08,029] {zip_file_operator.py:35} INFO - Unzipping 'home/airflow/imdb/title.ratings_2019-11-02.tsv.gz' to '/home/airflow/imdb/title.ratings_2019-11-02.tsv'.
[2019-11-03 20:42:08,200] {zip_file_operator.py:40} INFO - UnzipFileOperator done.
[2019-11-03 20:42:11,163] {logging_mixin.py:112} INFO - [2019-11-03 20:42:11,163] {local_task_job.py:124} WARNING - Time since last heartbeat(0.02 s) < hearttrate(5.0 s), sleeping for 4.979647 s
[2019-11-03 20:42:16,150] {logging_mixin.py:112} INFO - [2019-11-03 20:42:16,148] {local_task_job.py:103} INFO - Task exited with return code 0
```

Use for  
Debugging

Each Task has it's  
own Log





# Create Simple Example DAG

## 1. Open Dag File:

```
vi /home/airflow/airflow/dags/example_dag.py
```

```
from airflow import DAG
from airflow.operators.bash_operator import BashOperator
from datetime import datetime, timedelta

args = {
    'owner': 'airflow'
}

dag = DAG('ExampleDAG', default_args=args, description='Simple Example DAG',
          schedule_interval='56 18 * * *',
          start_date=datetime(2019, 10, 16), catchup=False, max_active_runs=1)

task_1 = BashOperator(
    task_id='print_date',
    bash_command='date',
    dag=dag)

task_2 = BashOperator(
    task_id='sleep',
    bash_command='sleep 5',
    retries=3,
    dag=dag)

task_1 >> task_2
```

Task 1

Task 2

**Task Execution Order**  
*task\_2* is dependent on *task\_1*

**DAG Definition, e.g.**

- Name
- Schedule Interval (Cron)
- Description
- Start date
- ...

# Spoon Interface

Execute DAG:

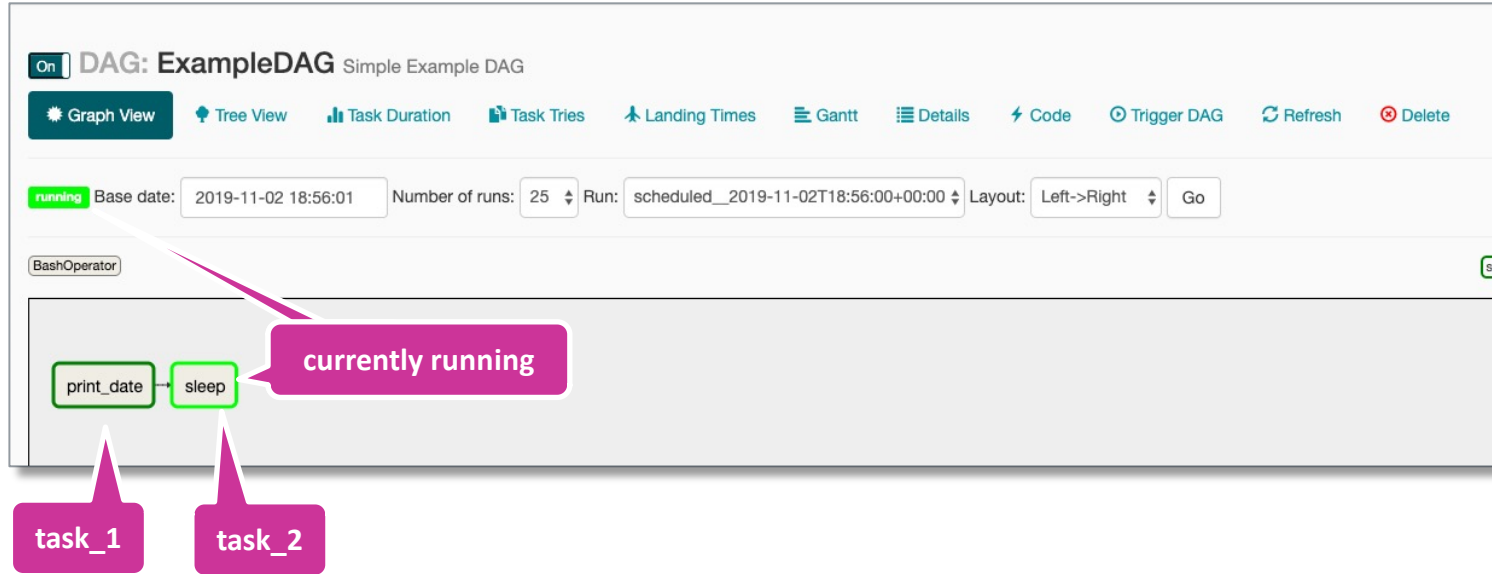
The screenshot shows the Apache Airflow web interface. The top navigation bar includes the Airflow logo and links for DAGs, Data Profiling, Browse, Admin, Docs, and About. The current page is 'DAGs'. A search bar is present. Below the search bar is a table of DAGs. The table has columns: Name, Schedule, Owner, Recent Tasks, Last Run, DAG Runs, and Links. Two DAGs are listed: 'ExampleDAG' and 'IMDb'. The 'IMDb' DAG is highlighted with a green circle around its 'On' button. A callout points to this button with the text 'Enable DAG For First Run'. Another callout points to the 'Trigger' icon in the 'Links' column of the 'IMDb' row with the text 'Or trigger DAG manually'.

		Schedule	Owner	Recent Tasks	Last Run	DAG Runs	Links
	<input type="checkbox"/> Off ExampleDAG	56 18 ***	airflow				
	<input checked="" type="checkbox"/> On IMDb	56 18 ***	airflow		2019-11-02 18:56		

Showing 1 to 2 of 2 entries

# Spoon Interface

See executing DAG:



# Spoon Interface

View Log of task *print\_date*:

print\_date on 2019-11-02T18:56:00+00:00

Task Instance Details Rendered Task Instances **View Log**

Download Log (by attempts):

1

Run Ignore All Deps Ignore Task Sta

Clear Past Future Upstream Downstream Recursive Failed

Mark Failed Past Future Upstream Downstream

Mark Success Past Future Upstream Downstream

Close

Press View Log

Instance: print\_date 2019-11-02 18:56:00

Log by attempts

Toggle wrap Jump to end

```
***
[2019-11-03 21:10:11,930] (taskinstance.py:630) INFO - Dependencies all met for <TaskInstance: ExampleDAG.print_date 2019-11-02
[2019-11-03 21:10:11,933] (taskinstance.py:630) INFO - Dependencies all met for <TaskInstance: ExampleDAG.print_date 2019-11-02
[2019-11-03 21:10:11,954] (taskinstance.py:841) INFO -
[2019-11-03 21:10:11,954] (taskinstance.py:842) INFO - Starting attempt 1 of 1
[2019-11-03 21:10:11,954] (taskinstance.py:843) INFO -
[2019-11-03 21:10:11,969] (taskinstance.py:862) INFO - Executing <Task(BashOperator): print_date> on 2019-11-02T18:56:00+00:00
[2019-11-03 21:10:11,970] (base_task_runner.py:133) INFO - Running: ['airflow', 'run', 'ExampleDAG', 'print_date', '2019-11-02T18:56:00+00:00']
[2019-11-03 21:10:12,035] (base_task_runner.py:115) INFO - Job 21: Subtask print_date [2019-11-03 21:10:12,035] (settings.py:25
[2019-11-03 21:10:12,060] (base_task_runner.py:115) INFO - Job 21: Subtask print_date /home/airflow/.local/lib/python3.6/site-p
[2019-11-03 21:10:12,060] (base_task_runner.py:115) INFO - Job 21: Subtask print_date ""
[2019-11-03 21:10:13,590] (base_task_runner.py:115) INFO - Job 21: Subtask print_date [2019-11-03 21:10:13,590] (__init__.py:51
[2019-11-03 21:10:13,590] (base_task_runner.py:115) INFO - Job 21: Subtask print_date [2019-11-03 21:10:13,597] (dagbag.py:92)
[2019-11-03 21:10:13,637] (base_task_runner.py:115) INFO - Job 21: Subtask print_date [2019-11-03 21:10:13,636] (cli.py:545) IN
[2019-11-03 21:10:13,660] (bash_operator.py:813) INFO - tmp dir root location:
/tmp
[2019-11-03 21:10:13,660] (bash_operator.py:91) INFO - Exporting the following env vars:
AIRFLOW_CTX_DAG_ID=ExampleDAG
AIRFLOW_CTX_TASK_ID=print_date
AIRFLOW_CTX_EXECUTION_DATE=2019-11-02T18:56:00+00:00
AIRFLOW_CTX_RUN_ID=scheduled_2019-11-02T18:56:00+00:00
[2019-11-03 21:10:13,660] (bash_operator.py:105) INFO - Temporary script location: /tmp/airflowtmpass3x9j/print_datev2ze8bz
[2019-11-03 21:10:13,660] (bash_operator.py:115) INFO - Running command: date
[2019-11-03 21:10:13,676] (bash_operator.py:124) INFO - Output:
Sun Nov 3 21:10:13 UTC 2019
[2019-11-03 21:10:13,681] (bash_operator.py:128) INFO - Command exited with return code 0
[2019-11-03 21:10:16,923] (local_task_job.py:124) WARNING - Time since last heartbeat: 3.24s
[2019-11-03 21:10:21,915] (local_task_job.py:103) INFO - Task exited successfully
```

date bash command of task\_1



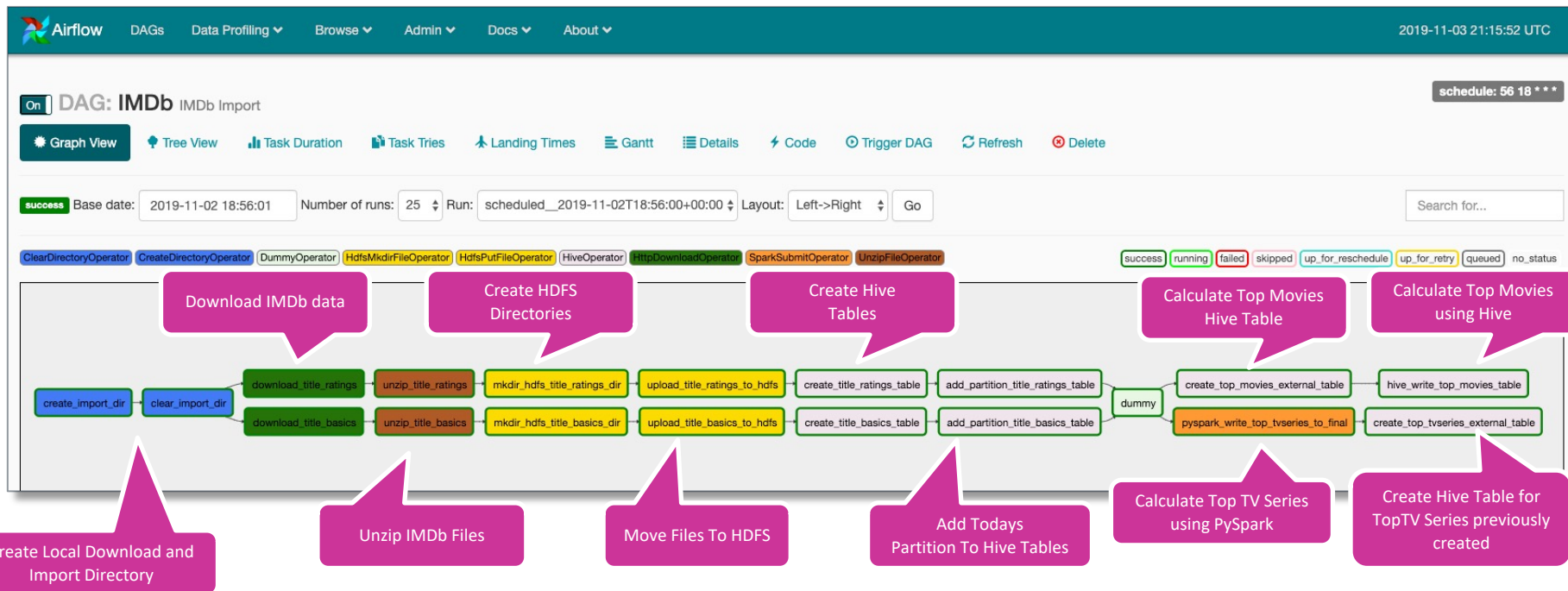
# Exercises II

Use Apache Airflow to solve exercises based on IMDb data



# Spoon Interface

See executing DAG:





# Pentaho Data Integration Exercises – IMDB

1. Execute IMDb DAG

2. Use **Airflow** and previous IMDb **DAG** to do following changes (`vi /home/airflow/airflow/dags/imdb.py`):

a) **Extend Airflow IMDb DAG** to also download *name.basics.tsv.gz*

b) **Extend Airflow IMDb DAG** to also import *name.basics.tsv* to HDFS raw layer.

c) **Create Hive table *name\_basics*** for *name.basics.tsv* in raw layer within DAG. Table should be partitioned by year, month and day of load date like the other tables.

d) **Create table *actors*** and **extend IMDb Airflow DAG** to fill table using Hive or PySpark:

- make use of all columns within table *name\_basics*
- add column ***alive*** which contains alive if actor is alive or dead if actor is dead
- add column ***age*** which contains current age of actor (calculated by using birth and death year)

e) Run DAG

Well Done

WE'RE DONE  
FOR  
...TODAY





# Stop Your VM Instances

DON'T FORGET TO  
STOP YOUR VM  
INSTANCE!



```
gcloud compute instances stop big-data
```

