



Apache Airflow workshop:

Throw away your cron jobs

Julia Volkova
Sevak Avetisyan
Vladimir Baev
Vladimir Gavrilenko



Agenda

1. Intro to Apache Airflow
2. Build your own first Airflow pipeline
 - Note: please, ensure that you have Python 3.6, Docker and Docker Compose installed

Our team



Julia Volkova

Python Specialization Lead
jvolkova@griddynamics.com



Sevak Avetisyan

Senior Data Engineer
savetisyan@griddynamics.com



Vladimir Baev

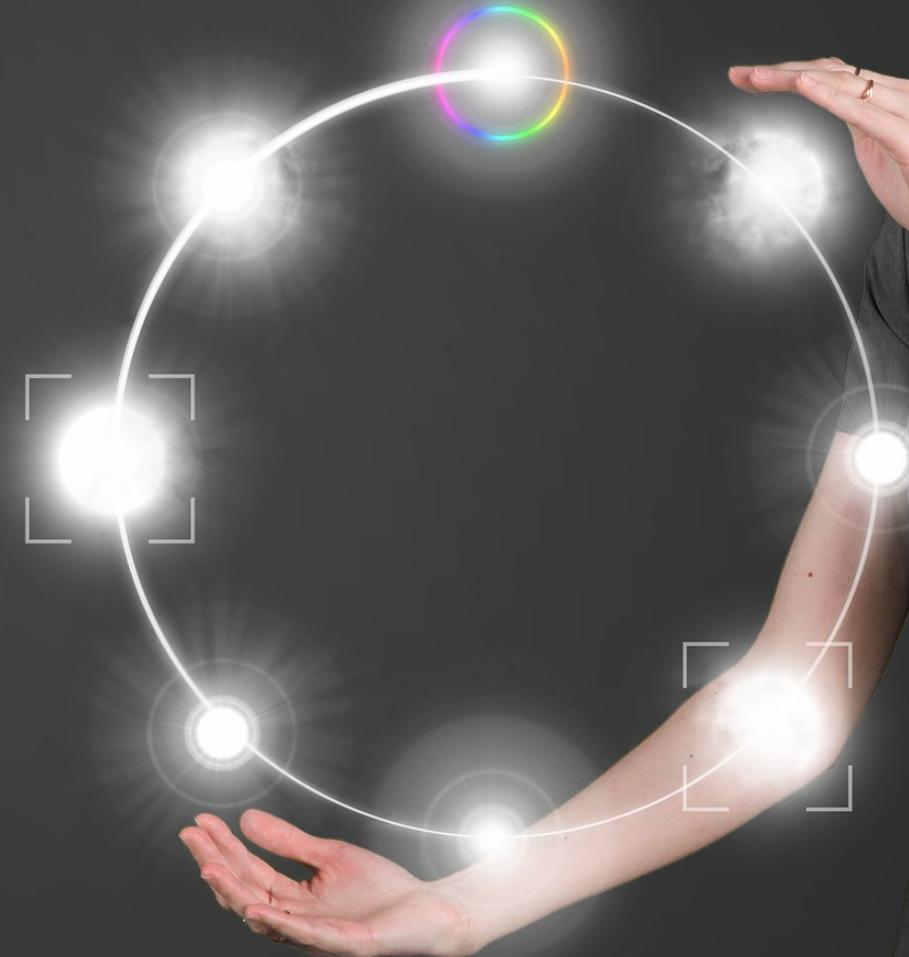
Senior Data Engineer
vbaev@griddynamics.com



Vladimir Gavrilenko

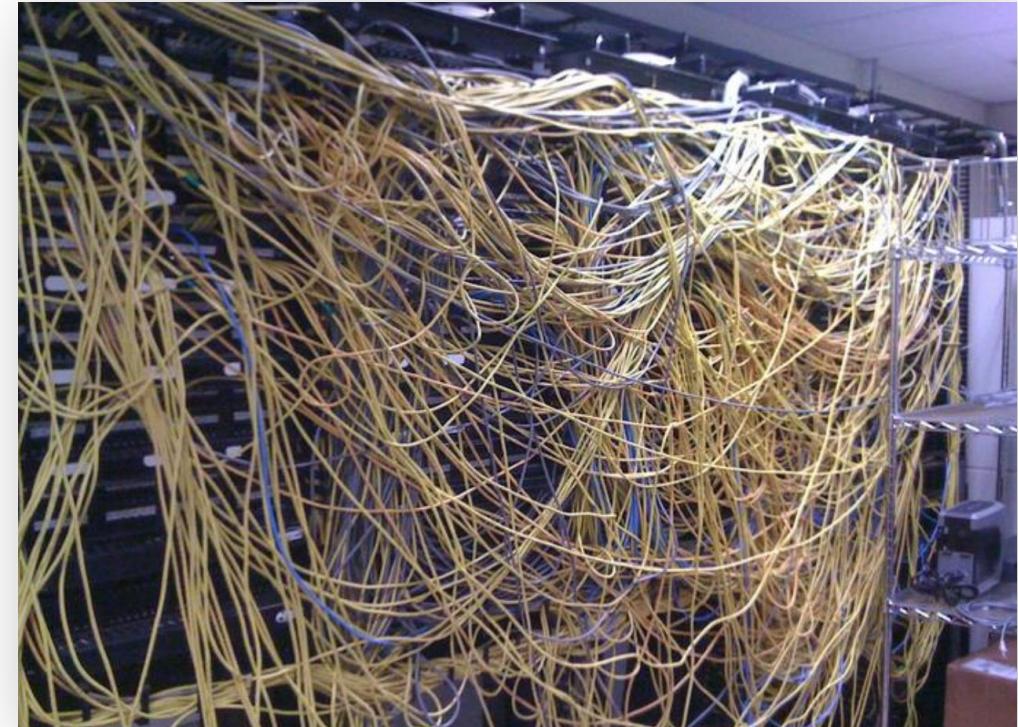
Data Engineer
vgavrilenko@griddynamics.com

Introduction to Apache Airflow

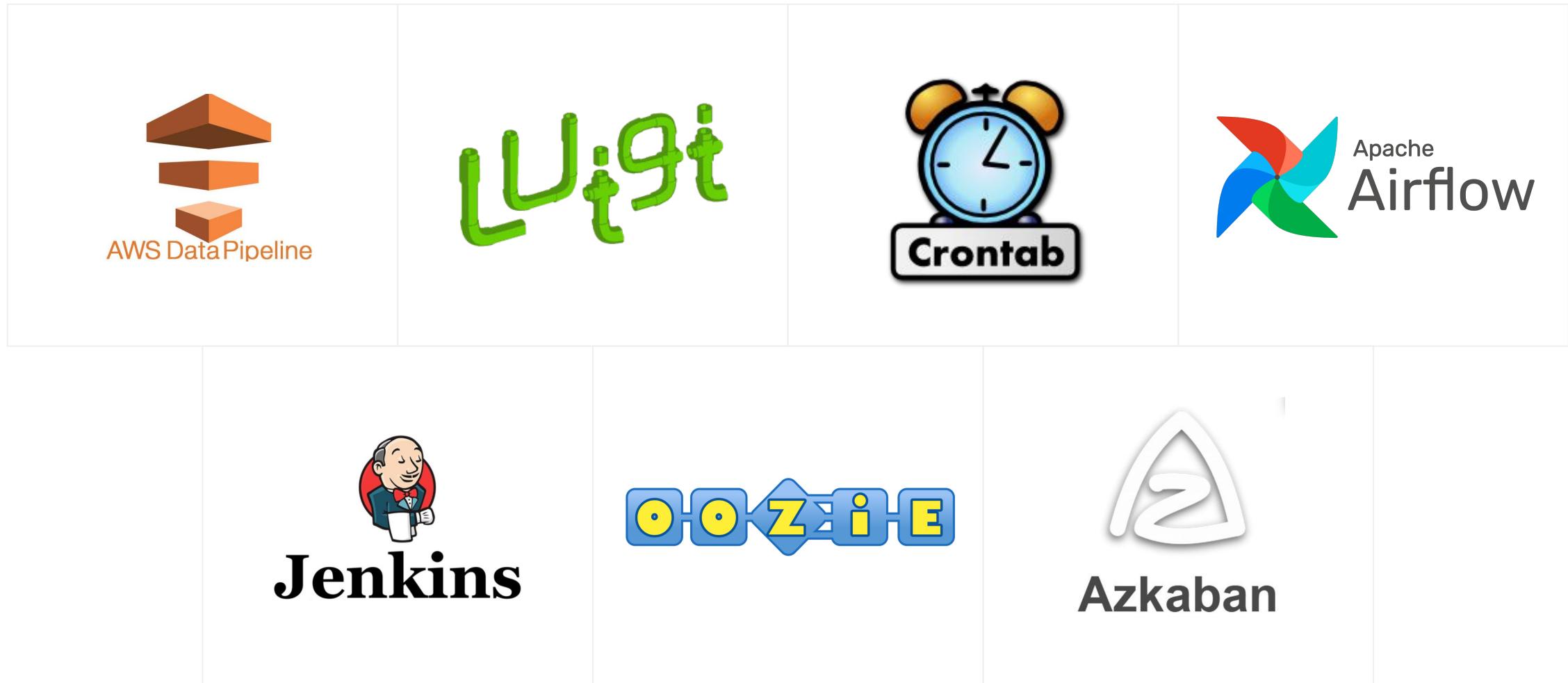


Orchestration tools

- Way of connecting application's components to provide appropriate scheduling and interaction
- Examples
 - Sequential processing of image by multiple separate logical units (resize, apply filters, export)
 - Sequence of ETL jobs, connected by inputs/outputs



Orchestration tools: major players



Apache Airflow: Overview

“Platform to programmatically author, schedule, and monitor workflows”

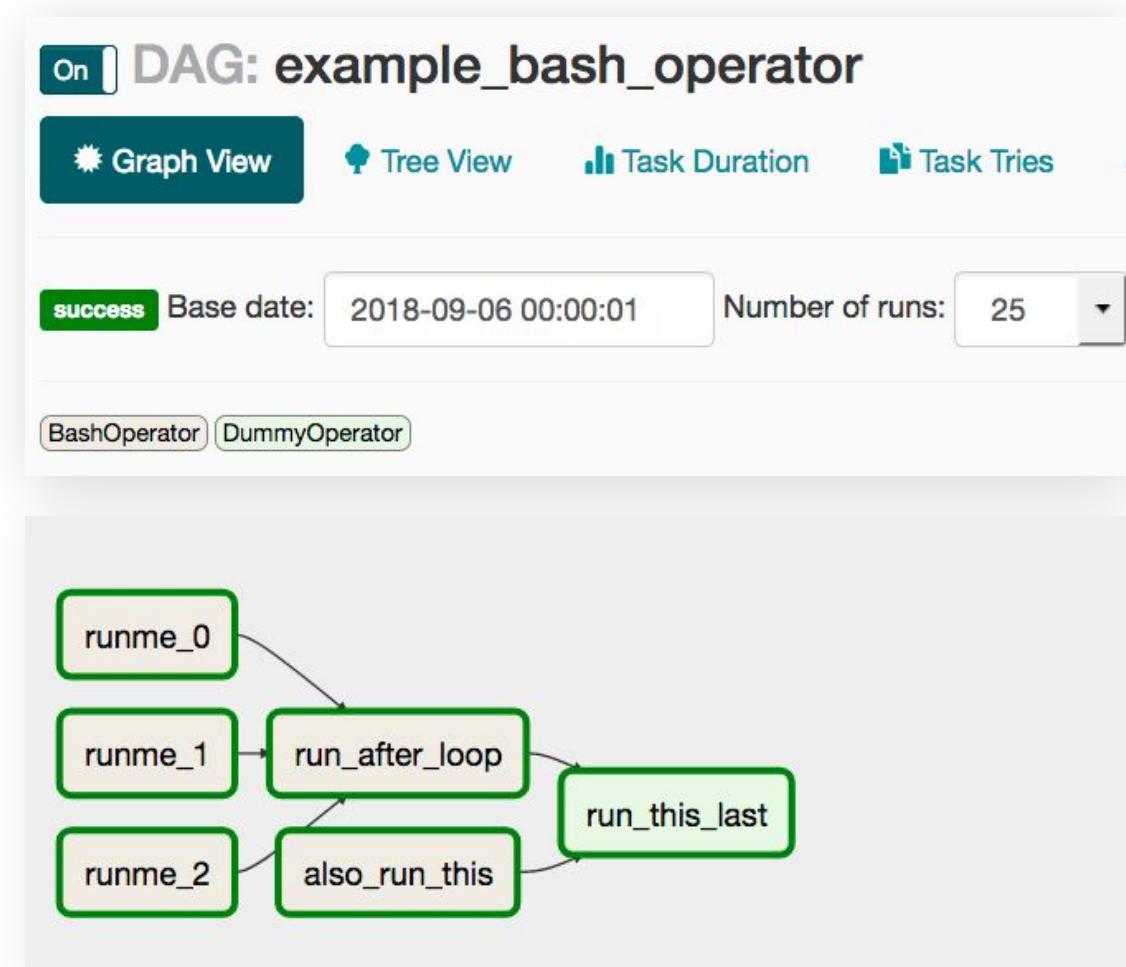


- Vocabulary

- DAGs (Directed acyclic graphs) = pipelines
- Tasks, Task instances
- Operators, Sensors
- DAG run
- master, worker nodes

- Features

- Batch-oriented
- Define pipelines (DAGs) as Python code
- Dynamic DAGs support
- Extensible
- Parameterizing with Jinja templates
- Scalable
- Rich UI



Apache Airflow: Overview

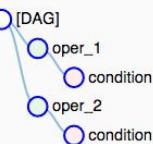
DAGs

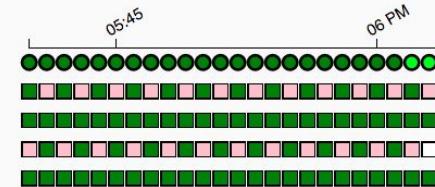
	i	DAG	Schedule	Owner	Recent Tasks i	Last Run i	DAG Runs i	Links
i	On	example_bash_operator	00***	airflow		2018-09-06 00:00 i		Graph Tree Duration Tries Landing Times Gantt Details Code
i	On	example_branch_dop_operator_v3	*/* ***	airflow		2018-09-05 00:56 i		Graph Tree Duration Tries Landing Times Gantt Details Code
i	On	example_branch_operator	@daily	airflow		2018-09-06 00:00 i		Graph Tree Duration Tries Landing Times Gantt Details Code
i	On	example_xcom	@once	airflow		2018-09-05 00:00 i		Graph Tree Duration Tries Landing Times Gantt Details Code
i	On	latest_only	4:00:00	Airflow		2018-09-07 16:00 i		Graph Tree Duration Tries Landing Times Gantt Details Code

[On](#) DAG: example_branch_dop_operator_v3

[Graph View](#) [Tree View](#) [Task Duration](#) [Task Tries](#) [Landing Times](#) [Gantt](#) [Details](#) [Code](#)

Base date: 2018-09-05 01:04:00 Number of runs: 25 [Go](#)





run_after_loop [▼](#) on 2018-09-08T00:00:00+00:00

[Task Instance Details](#) [Rendered](#) [Task Instances](#) [View Log](#)

[Run](#) [Ignore All Deps](#) [Ignore Task State](#) [Ignore Task Deps](#)

[Clear](#) [Past](#) [Future](#) [Upstream](#) [Downstream](#) [Recursive](#)

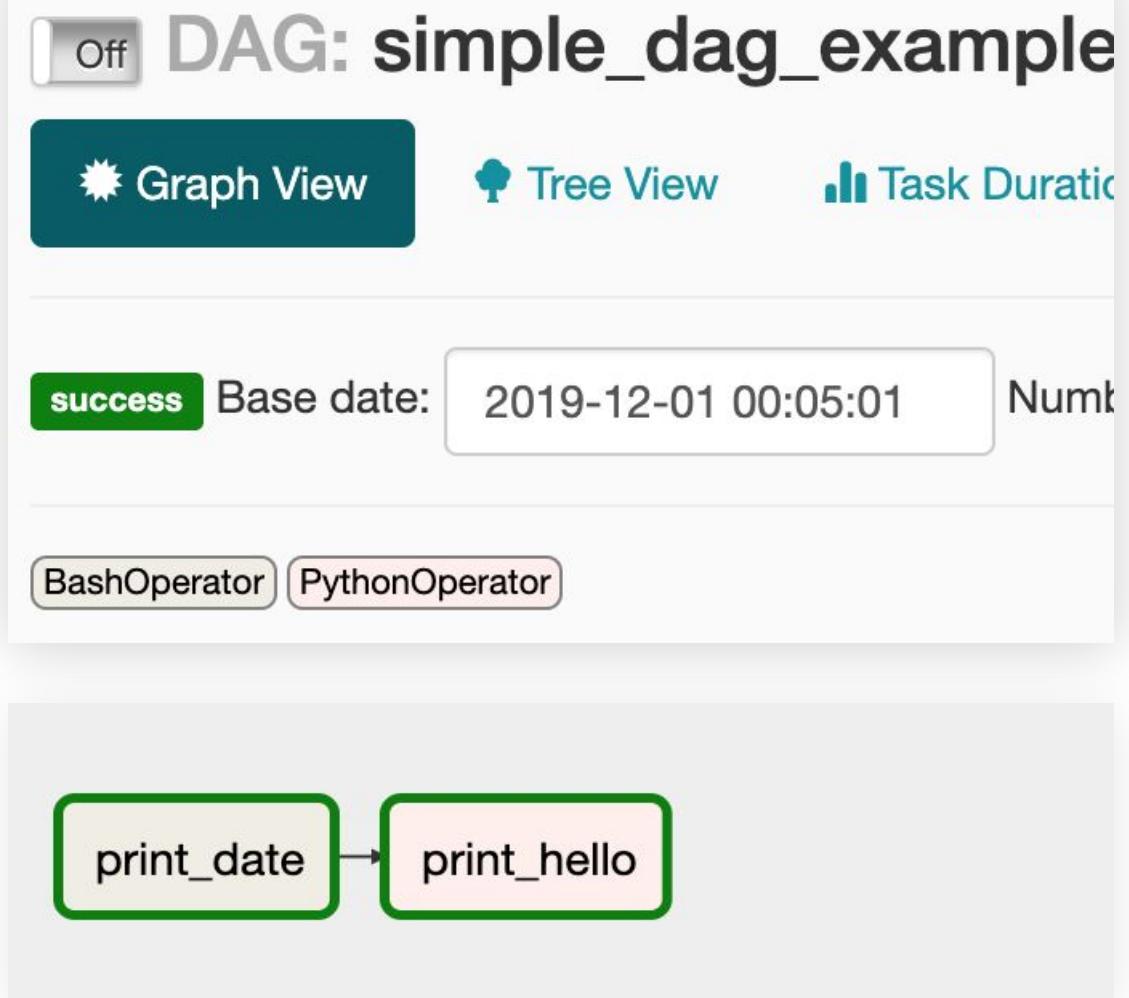
[Mark Failed](#) [Past](#) [Future](#) [Upstream](#) [Downstream](#)

[Mark Success](#) [Past](#) [Future](#) [Upstream](#) [Downstream](#)

[Close](#)

Simple DAG

```
1  dag = DAG(
2      'simple_dag_example',
3      default_args=default_args,
4      description='A simple tutorial DAG',
5      schedule_interval='*/5 * * * *',
6  )
7
8  t1 = BashOperator(
9      task_id='print_date',
10     bash_command='date',
11     dag=dag
12 )
13
14 def print_hello():
15     print('Hello!')
16
17 t2 = PythonOperator(
18     task_id='print_hello',
19     python_callable=print_hello,
20     dag=dag
21 )
22
23 t1 >> t2
```



Simple DAG

On DAG: **simple_dag_example** A simple tutorial DAG schedule: */5 * * * *

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

Base date: 2019-12-01 00:05:00 Number of runs: 25 Go

BashOperator PythonOperator success running failed skipped up_for_reschedule up_for_retry queued no_status

[DAG] print_hello print_date

Log by attempts

1 Toggle wrap Jump to end

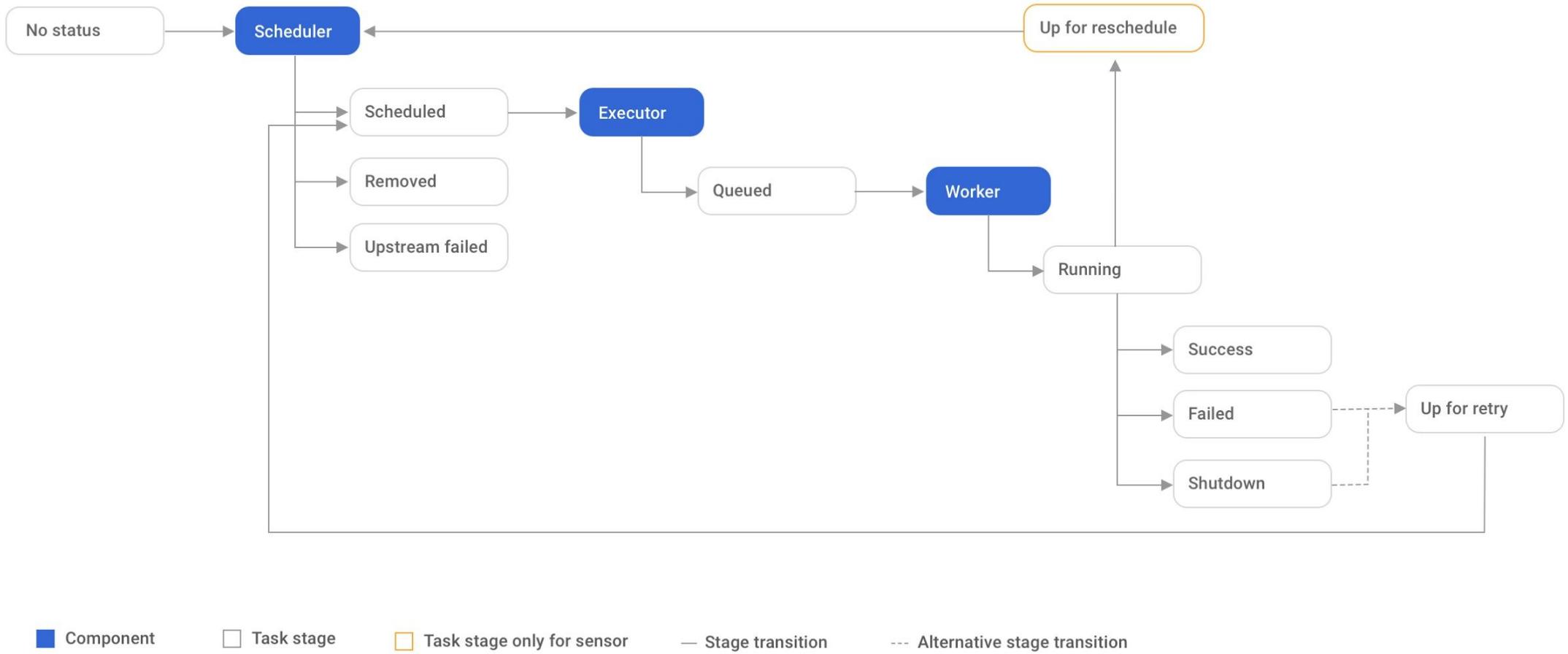
```
*** Reading local file: /usr/local/airflow/logs/simple_dag_example/print_hello/2019-12-01T00:00:00+00:00/1.log
[2019-12-01 00:05:10,229] {{taskinstance.py:620}} INFO - Dependencies all met for <TaskInstance: simple_dag_example.print_hello 2019-12-01T00:00:00+00:00 [queued]
[2019-12-01 00:05:10,259] {{taskinstance.py:620}} INFO - Dependencies all met for <TaskInstance: simple_dag_example.print_hello 2019-12-01T00:00:00+00:00 [queued]
[2019-12-01 00:05:10,260] {{taskinstance.py:838}} INFO -
[2019-12-01 00:05:10,260] {{taskinstance.py:839}} INFO - Starting attempt 1 of 2
[2019-12-01 00:05:10,260] {{taskinstance.py:840}} INFO -

[2019-12-01 00:05:10,281] {{taskinstance.py:859}} INFO - Executing <Task(PythonOperator): print_hello> on 2019-12-01T00:00:00+00:00
[2019-12-01 00:05:10,281] {{base_task_runner.py:133}} INFO - Running: ['airflow', 'run', 'simple_dag_example', 'print_hello', '2019-12-01T00:00:00+00:00', '-c', '/tmp/airflow/2019-12-01/1/1/1']
[2019-12-01 00:05:11,840] {{base_task_runner.py:115}} INFO - Job 26: Subtask print_hello [2019-12-01 00:05:11,839] {{settings.py:213}} INFO - settings.configure()
[2019-12-01 00:05:11,911] {{base_task_runner.py:115}} INFO - Job 26: Subtask print_hello /usr/local/lib/python3.7/site-packages/psycopg2/__init__.py:144: UserWarning: The psycopg2 package was built against PostgreSQL 11, but a different version of PostgreSQL is currently in use.
[2019-12-01 00:05:11,911] {{base_task_runner.py:115}} INFO - Job 26: Subtask print_hello ""
[2019-12-01 00:05:12,570] {{base_task_runner.py:115}} INFO - Job 26: Subtask print_hello [2019-12-01 00:05:12,569] {{__init__.py:51}} INFO - Using executor LocalExecutor
[2019-12-01 00:05:14,202] {{base_task_runner.py:115}} INFO - Job 26: Subtask print_hello [2019-12-01 00:05:14,202] {{dagbag.py:90}} INFO - Filling up the DagBag with 1 DAGs
[2019-12-01 00:05:14,276] {{base_task_runner.py:115}} INFO - Job 26: Subtask print_hello [2019-12-01 00:05:14,275] {{cli.py:516}} INFO - Running <TaskInstance: simple_dag_example.print_hello 2019-12-01T00:00:00+00:00>
[2019-12-01 00:05:14,312] {{python_operator.py:105}} INFO - Exporting the following env vars:
AIRFLOW_CTX_DAG_ID=simple_dag_example
AIRFLOW_CTX_TASK_ID=print_hello
AIRFLOW_CTX_EXECUTION_DATE=2019-12-01T00:00:00+00:00
AIRFLOW_CTX_DAG_RUN_ID=scheduled_2019-12-01T00:00+00:00
[2019-12-01 00:05:14,313] {{logging_mixin.py:95}} INFO - Hello!
[2019-12-01 00:05:14,313] {{python_operator.py:114}} INFO - Done. Returned value was: None
[2019-12-01 00:05:15,197] {{logging_mixin.py:95}} INFO - [ [34m2019-12-01 00:05:15,196 [0m] {{ [34mlocal_task_job.py: [0m105}} INFO [0m - Task exited with return code None
```

Apache Airflow: Concepts

- Operators and Sensors
 - BashOperator
 - PythonOperator
 - PostgresOperator
 - SparkSubmitOperator
 - BaseBranchOperator, TriggerDagRunOperator, SubDagOperator
 - S3PrefixSensor
- Pools, Queues
- Hooks (HDFSHook, SlackHook), Connections, Variables, XComs, etc

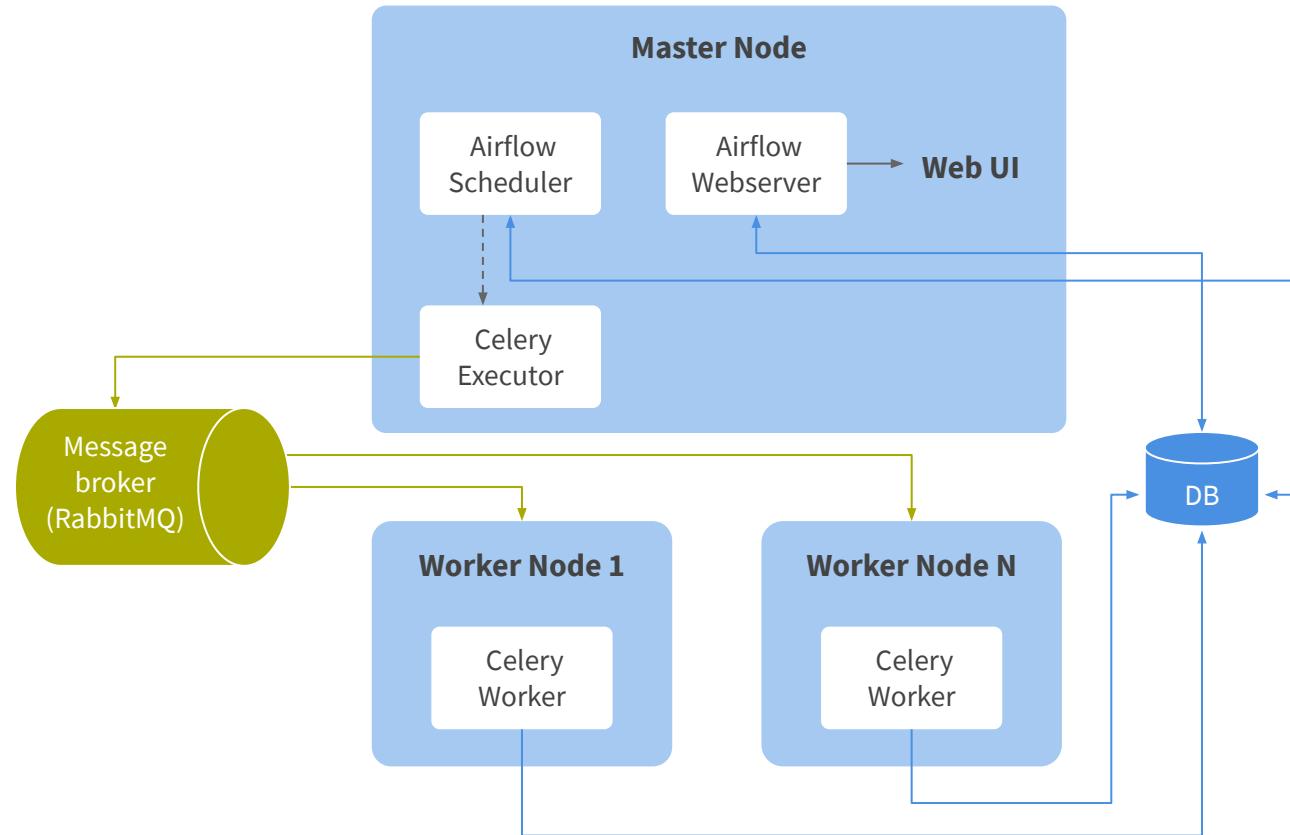
Infrastructure Task Lifecycle



Infrastructure

Airflow setup (Celery Executors)

- Master node have:
 - Airflow scheduler
 - Webserver
- Several Airflow workers (Celery Executors)
- Python virtualenvs
 - Airflow runs scheduler and executes tasks under virtualenvs
 - scheduler env
 - workers env
 - PythonVirtualenvOperator
- Migrations
 - Airflow versions
 - Python versions



Infrastructure Airflow Integration

- Executors
 - SequentialExecutor
 - LocalExecutor
 - CeleryExecutor
 - Kubernetes Executor
- Cloud integrations (Operators, Hooks)
 - Azure
 - AWS
 - GCP
- Command Line Interface
- REST API (experimental)
- Airflow DataBase

Practical part



Let's launch a rocket



- Our goals for today
 - Learn something cool about Airflow
 - Build your first DAG
 - Have fun

Let's launch a rocket



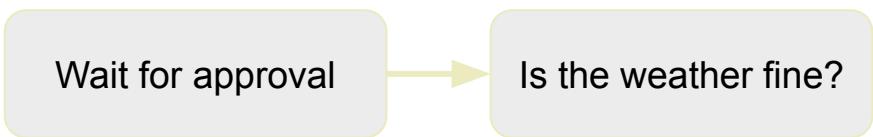
- Our goals for today
 - Learn something cool about Airflow
 - Build your first DAG
 - Have fun
- Implementation:

Wait for approval

Let's launch a rocket



- Our goals for today
 - Learn something cool about Airflow
 - Build your first DAG
 - Have fun
- Implementation:



Let's launch a rocket



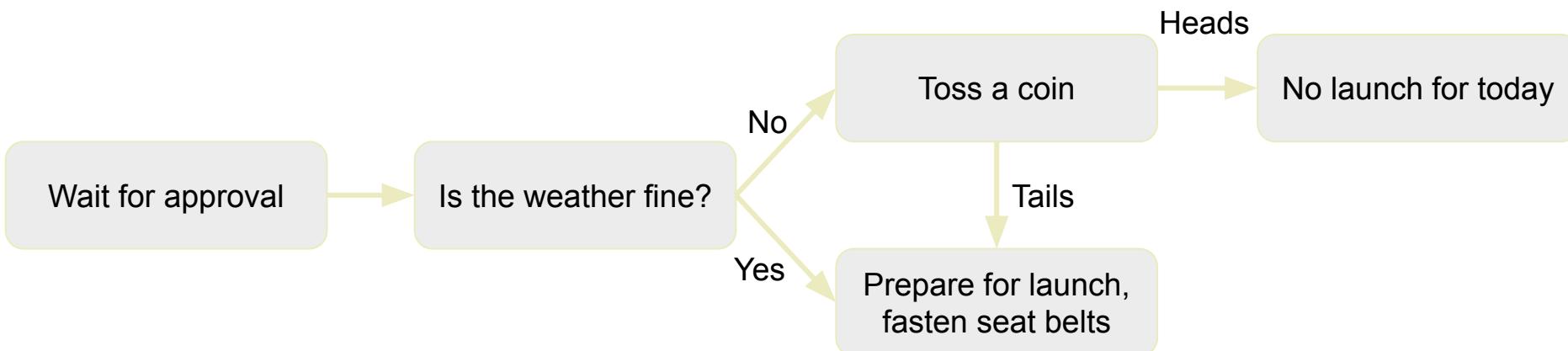
- Our goals for today
 - Learn something cool about Airflow
 - Build your first DAG
 - Have fun
- Implementation:



Let's launch a rocket



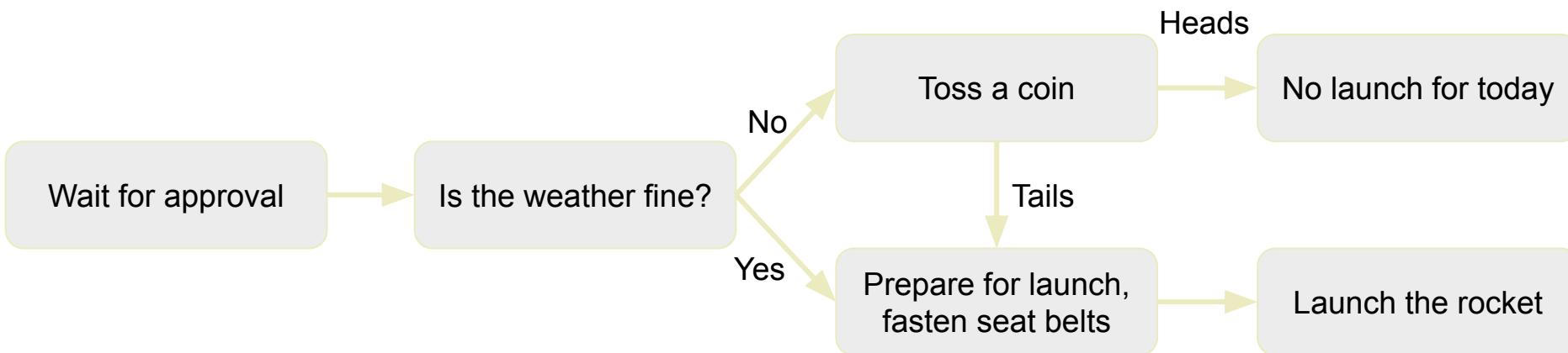
- Our goals for today
 - Learn something cool about Airflow
 - Build your first DAG
 - Have fun
- Implementation:



Let's launch a rocket



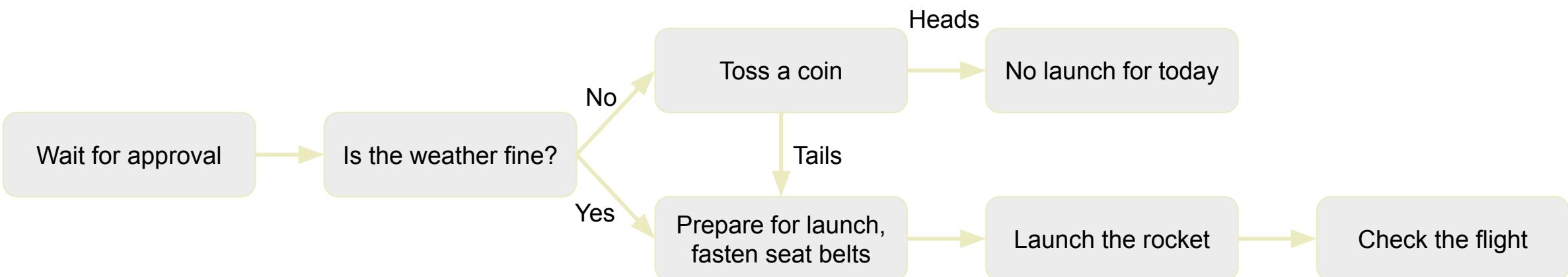
- Our goals for today
 - Learn something cool about Airflow
 - Build your first DAG
 - Have fun
- Implementation:



Let's launch a rocket



- Our goals for today
 - Learn something cool about Airflow
 - Build your first DAG
 - Have fun
- Implementation:



Environment setup

- Prerequisites
 - Python 3.6
 - Docker
 - Docker Compose
- Setup Airflow
 - Download workshop [repo](#)
 - Execute steps from README.md
 - Check that example DAG works

It's coding time



Resources

- https://github.com/xnuinside/piter_py_2020_apache_airflow
- <https://airflow.apache.org/>
- <https://github.com/apache/airflow>
- <https://stackoverflow.com/questions/tagged/airflow/>
- Common Pitfalls: <https://cwiki.apache.org/confluence/display/AIRFLOW/Common+Pitfalls>
- Official Slack workspace: <https://apache-airflow.slack.com/>
- Russian Telegram community (870 members): <https://t.me/ruairflow>



Thank you!

www.griddynamics.com

