

# Apache Airflow workshop:

Throw away your cron jobs

Andrey Elistratov  
Sevak Avetisyan  
Vladimir Baev

Privileged and confidential



# 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



**Andrey Elistratov**

Data Engineer  
aelistratov@griddynamics.com



**Sevak Avetisyan**

Senior Data Engineer  
savetisyan@griddynamics.com

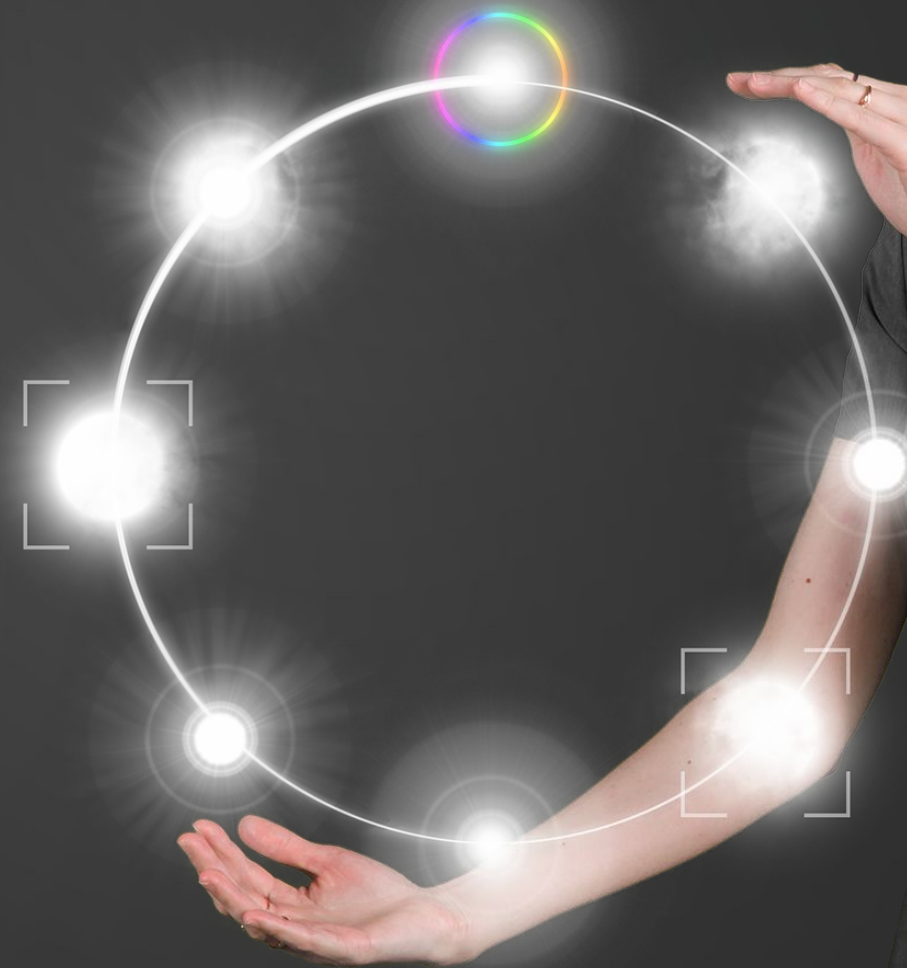


**Vladimir Baev**

Delivery Manager  
vbaev@griddynamics.com

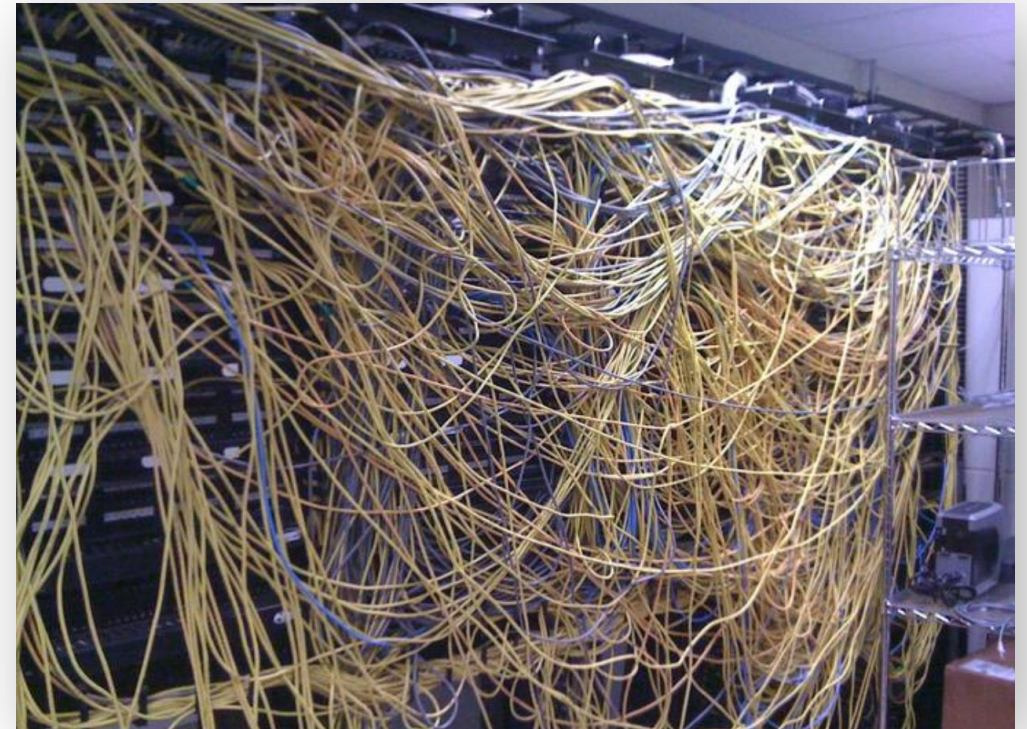
# Introduction to Apache Airflow

---



# 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

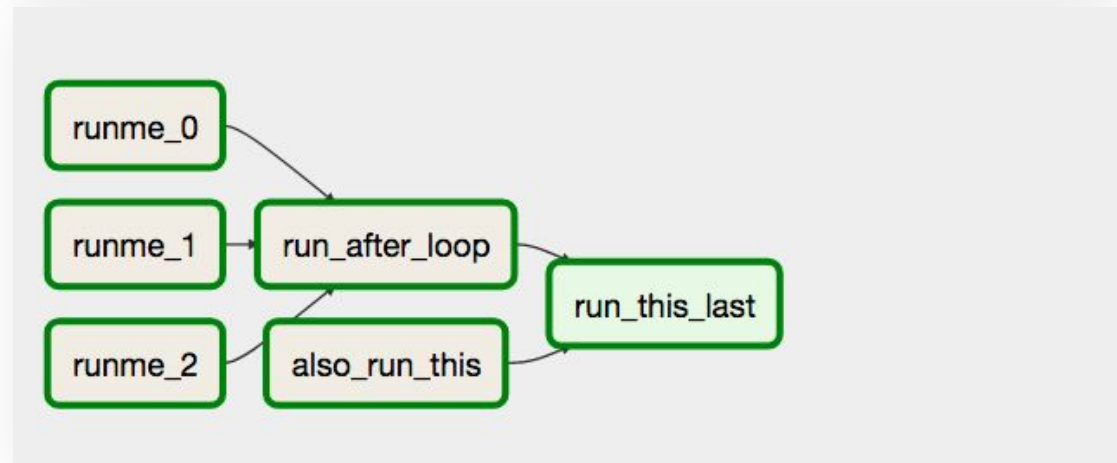
- Batch-oriented
- Define pipelines (DAGs) as Python code
- Dynamic DAGs support
- Extensible
- Parameterizing with Jinja templates
- Scalable
- Rich UI

On **DAG: example\_bash\_operator**

**Graph View** Tree View Task Duration Task Tries

**success** Base date: 2018-09-06 00:00:01 Number of runs: 25

BashOperator DummyOperator



# Apache Airflow: Overview

## DAGs

Search:

		DAG	Schedule	Owner	Recent Tasks ⓘ	Last Run ⓘ	DAG Runs ⓘ	Links
	<input checked="" type="checkbox"/>	example_bash_operator	00***	airflow	<div><div>6</div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>	2018-09-06 00:00 ⓘ	<div><div>5</div><div></div><div></div></div>	
	<input checked="" type="checkbox"/>	example_branch_dop_operator_v3	*/* ****	airflow	<div><div>3</div><div>1</div><div></div><div></div><div></div><div></div><div>1</div><div>5</div></div>	2018-09-05 00:56 ⓘ	<div><div>54</div><div>3</div><div></div></div>	
	<input checked="" type="checkbox"/>	example_branch_operator	@daily	airflow	<div><div>5</div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>	2018-09-06 00:00 ⓘ	<div><div>2</div><div></div><div></div></div>	
	<input checked="" type="checkbox"/>	example_xcom	@once	airflow	<div><div>3</div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>	2018-09-05 00:00 ⓘ	<div><div>1</div><div></div><div></div></div>	
	<input checked="" type="checkbox"/>	latest_only	4:00:00	Airflow	<div><div>2</div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>	2018-09-07 16:00 ⓘ	<div><div>35</div><div></div><div></div></div>	

### ☒ 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

BranchPythonOperator DummyOperator






run\_after\_loop  on 2018-09-08T00:00:00+00:00



# 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
```

☐ Off **DAG: simple\_dag\_example**

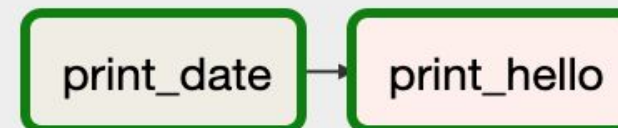
 **Graph View**  **Tree View**  **Task Duration**

---

**success** Base date: 2019-12-01 00:05:01 Number of tasks: 2

---

**BashOperator** **PythonOperator**



# 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: 
 Number of runs:

☒ BashOperator
 ☐ PythonOperator
 

■ success 
 ■ running 
 ■ failed 
 ■ skipped 
 ■ up\_for\_rescheduled 
 ■ up\_for\_retry 
  queued 
  no\_status

### Log by attempts

1

```

*** 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 [q
[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 [q
[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 - 

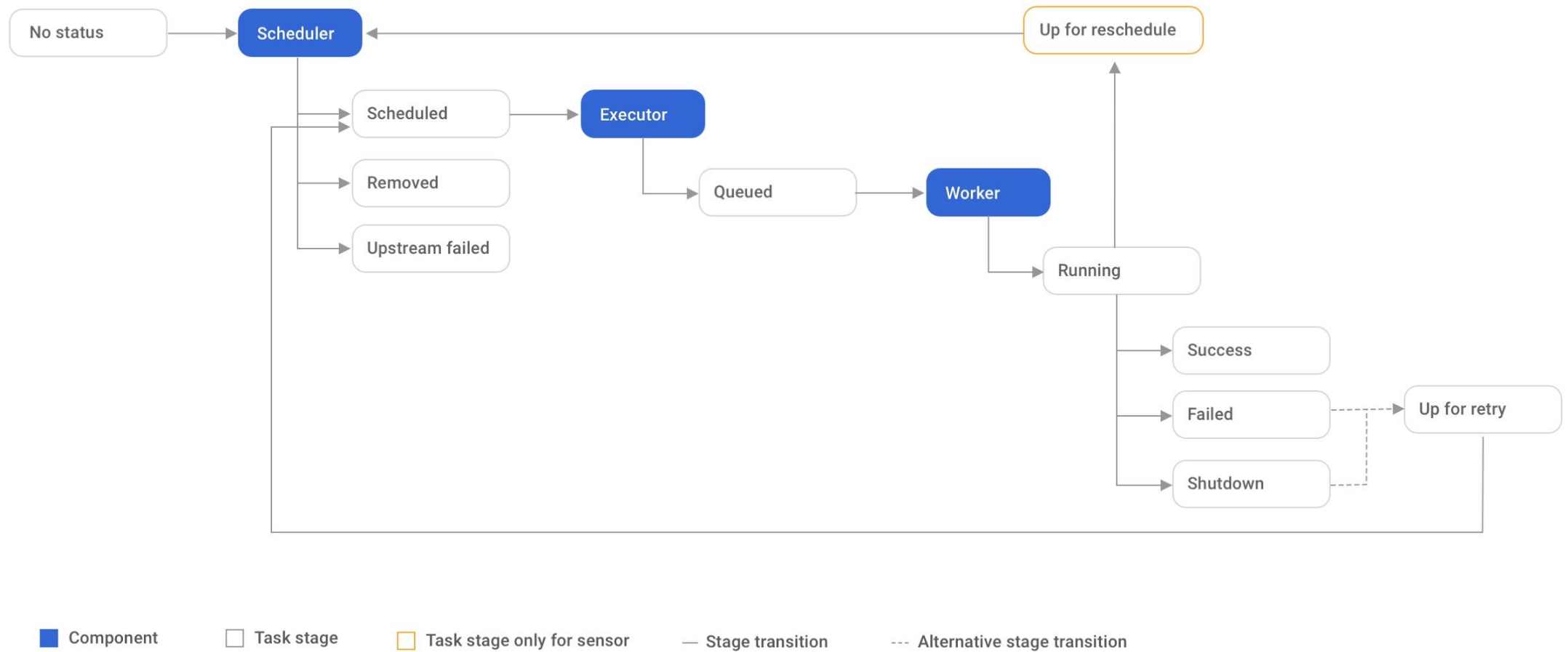
-----
[2019-12-01 00:05:10,281] {{taskinstance.py:859}} INFO - Executing <Task(PythonOperator): print_hello> on 2019-12-01T00:00:00+00:00
[2019-12-01 00:05:10,281] {{base_task_runner.py:133}} INFO - Running: ['airflow', 'run', 'simple_dag_example', 'print_hello', '2019-12-01T00:00:00+00:00', '
[2019-12-01 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.confir
[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: Us
[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
[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 Da
[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 <TaskInstan
[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+00:00
[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 retr

```

# Apache Airflow: Concepts

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

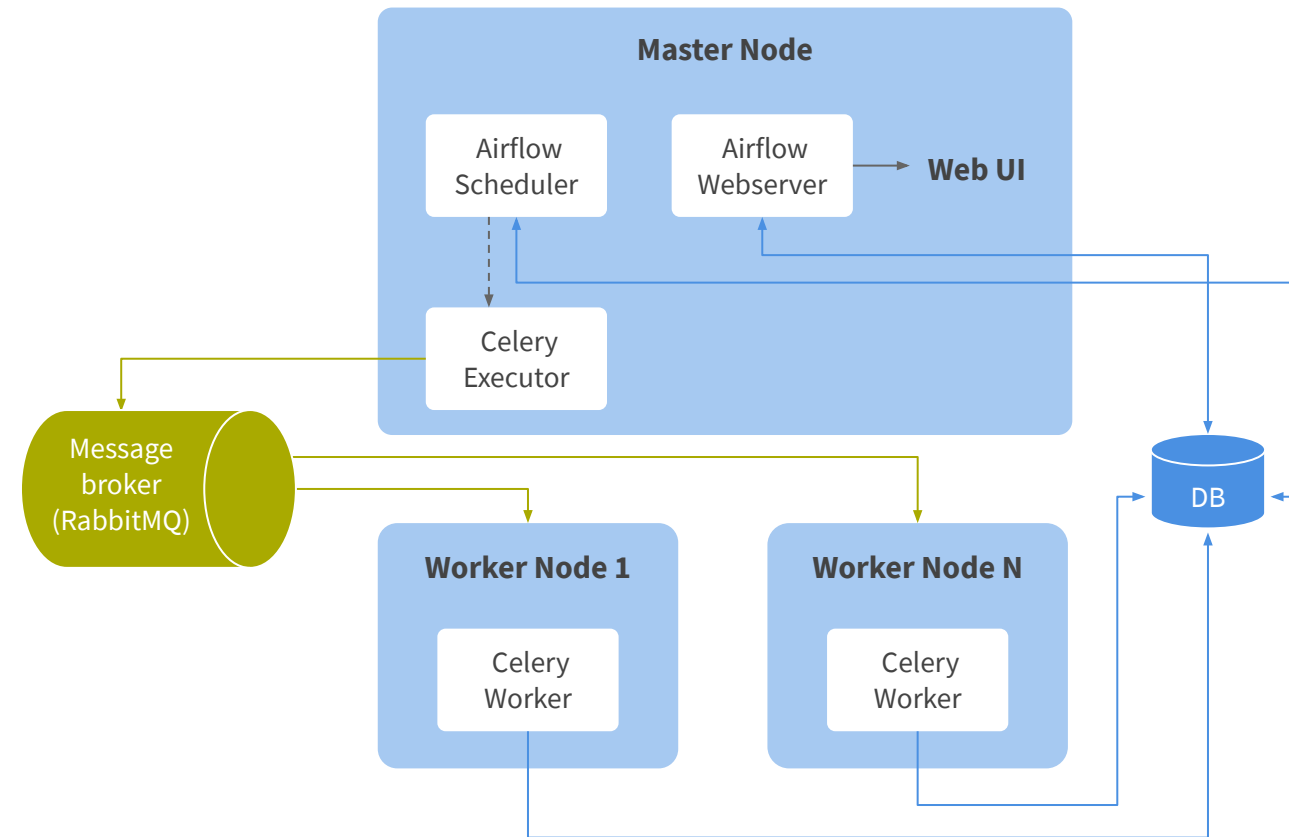
# Infrastructure Task Lifecycle



# Infrastructure

## Airflow setup (Celery Executors)

- Master node have:
  - Airflow scheduler
  - Webserver
- Several Airflow workers (CeleryExecutors)
- 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:

Wait for approval

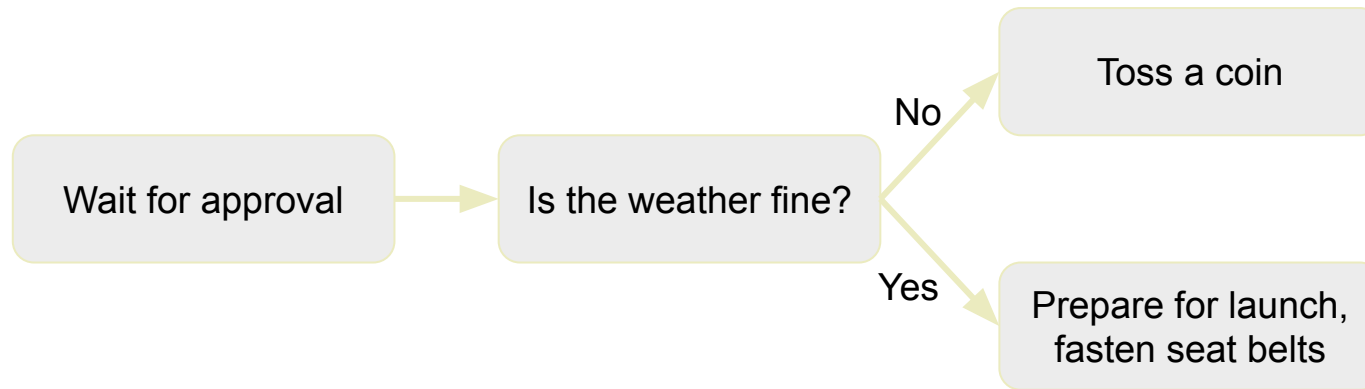


Is the weather fine?



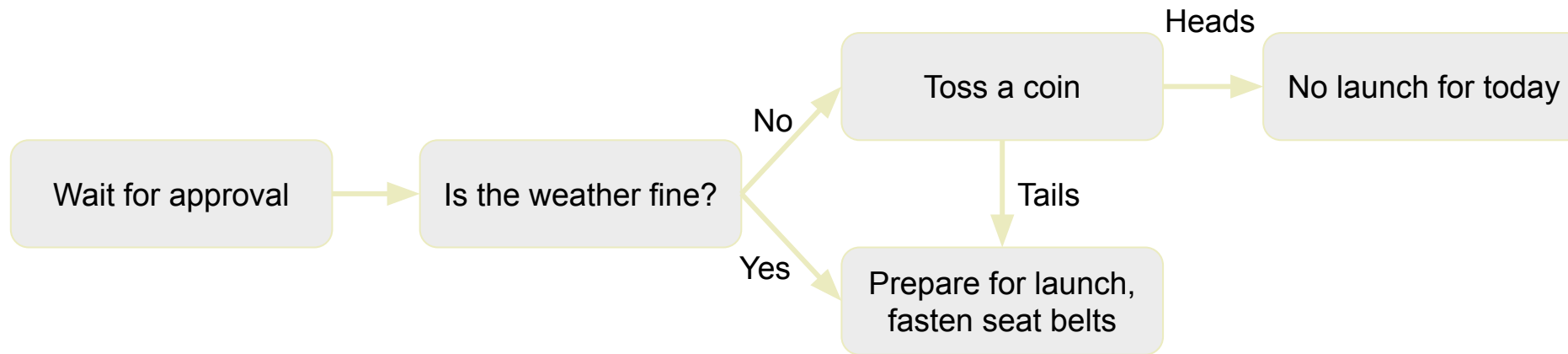
# Let's launch a rocket 🚀

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



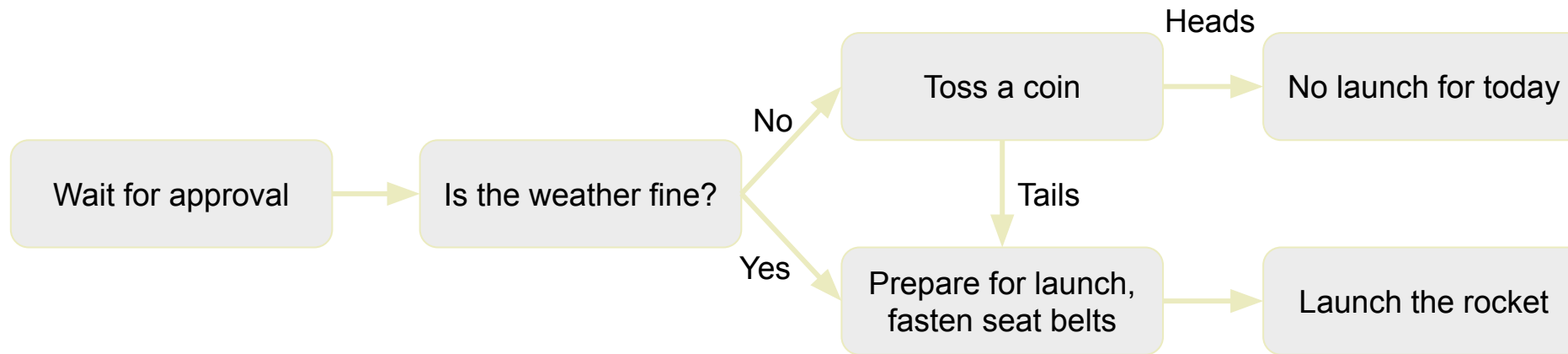
# Let's launch a rocket 🚀

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



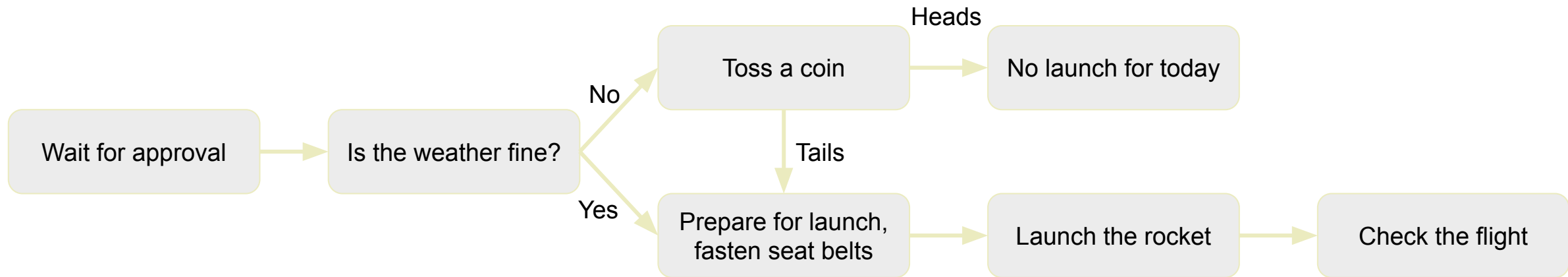
# Let's launch a rocket 🚀

- 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

---

11011101111  
0101111000011111101  
10101011101111010111101  
11010000010110111110  
1010101110101110010  
10101010111110101100101  
0101011101011001011110  
0111110101



0111010101  
010101111  
0010101  
010100101  
010101010  
011011110  
10101010111  
1100101  
01000011110

# Resources

- [https://github.com/vladimirbaev/grid\\_dynamics\\_apache\\_airflow\\_workshop\\_2021](https://github.com/vladimirbaev/grid_dynamics_apache_airflow_workshop_2021)
- <https://airflow.apache.org/>
- <https://github.com/apache/airflow>
- <https://stackoverflow.com/questions/tagged/airflow/>
- Common Pitfalls: <https://cwiki.apache.org/confluence/pages/viewpage.action?pageId=62694614>
- Official Slack workspace: <https://apache-airflow.slack.com/>
- Russian Telegram community (1200 members): <https://t.me/ruairflow>



# Thank you!

[www.griddynamics.com](http://www.griddynamics.com)

