Grid Dynamics

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

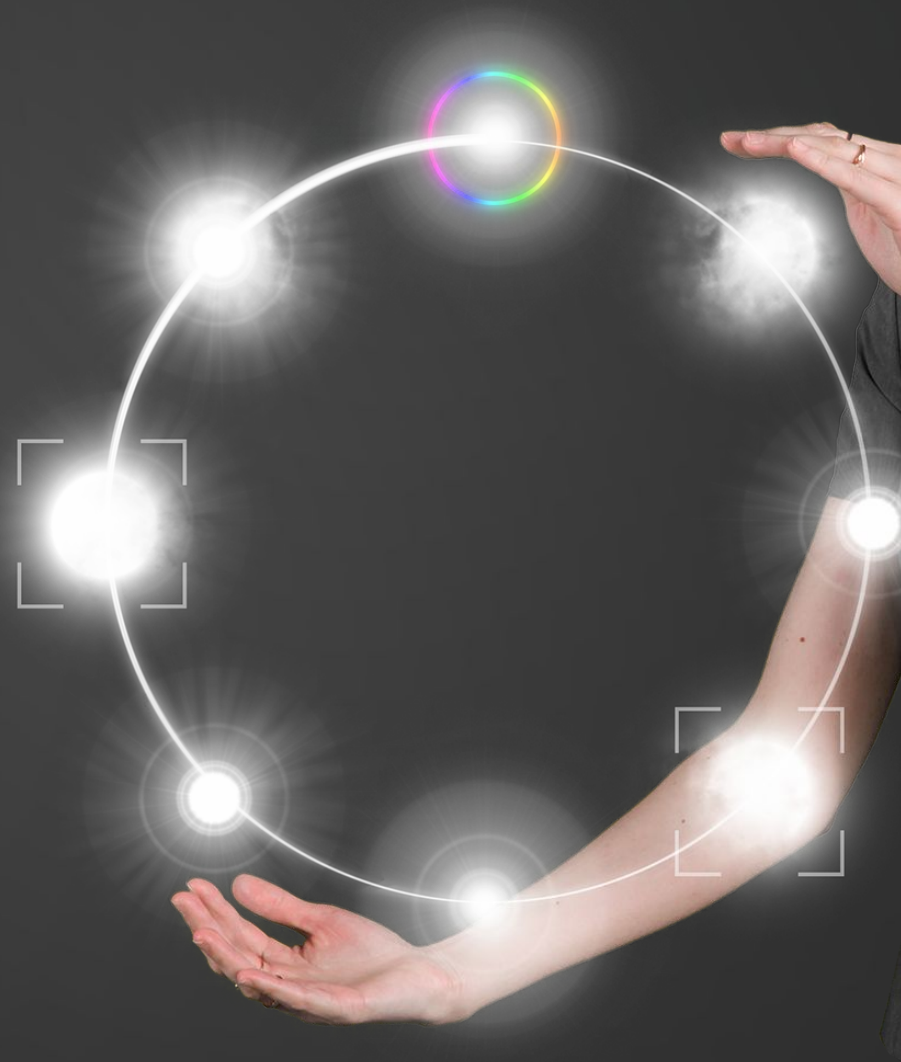
Senior Data Engineer
vbaev@griddynamics.com

Vladimir Gavrilenko

Data Engineer
vgavrilenko@griddynamics.com

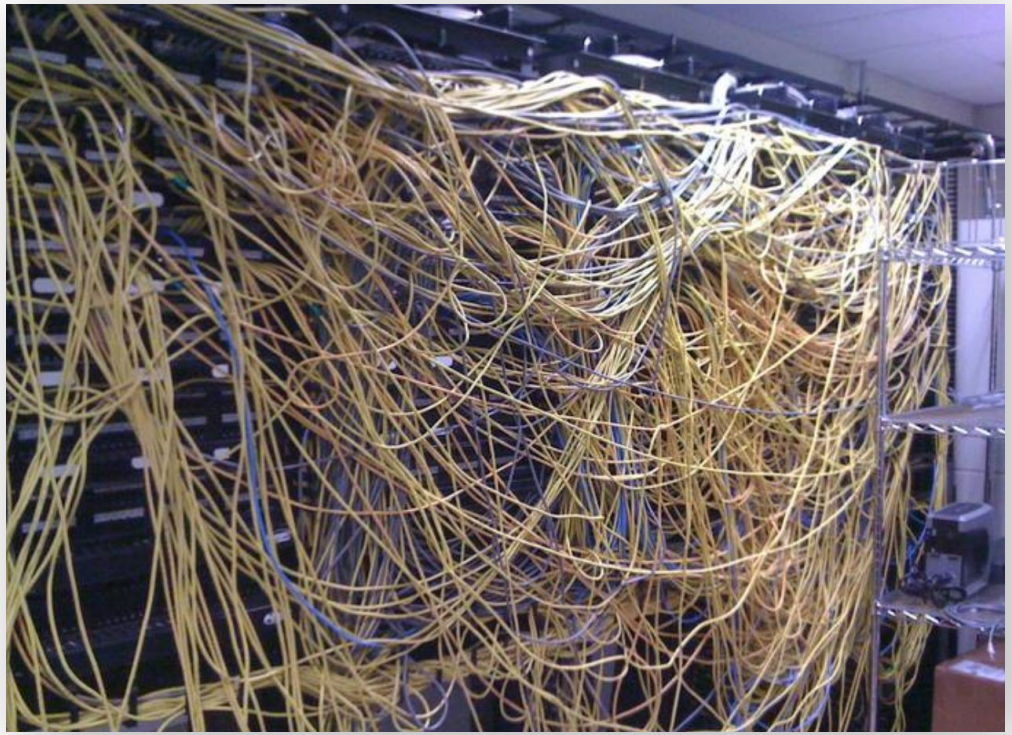
Grid Dynamics

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

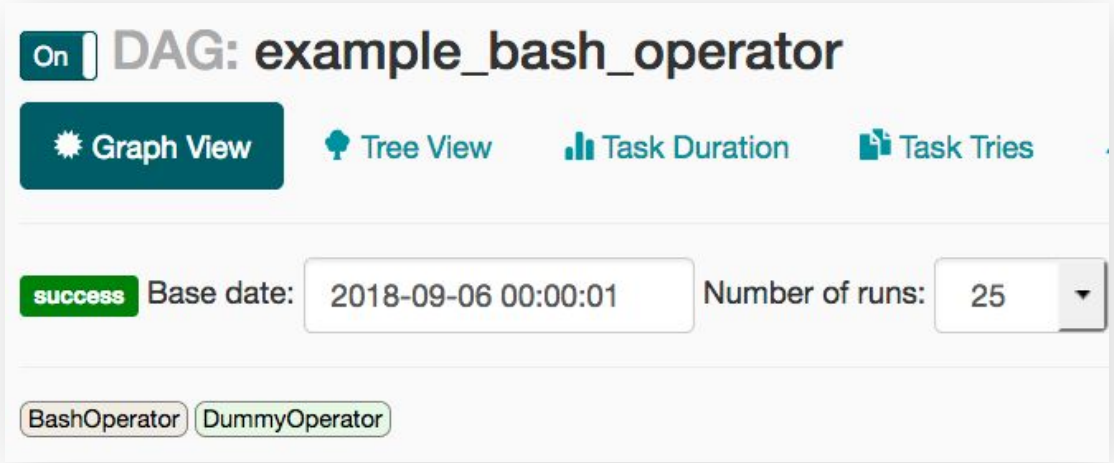


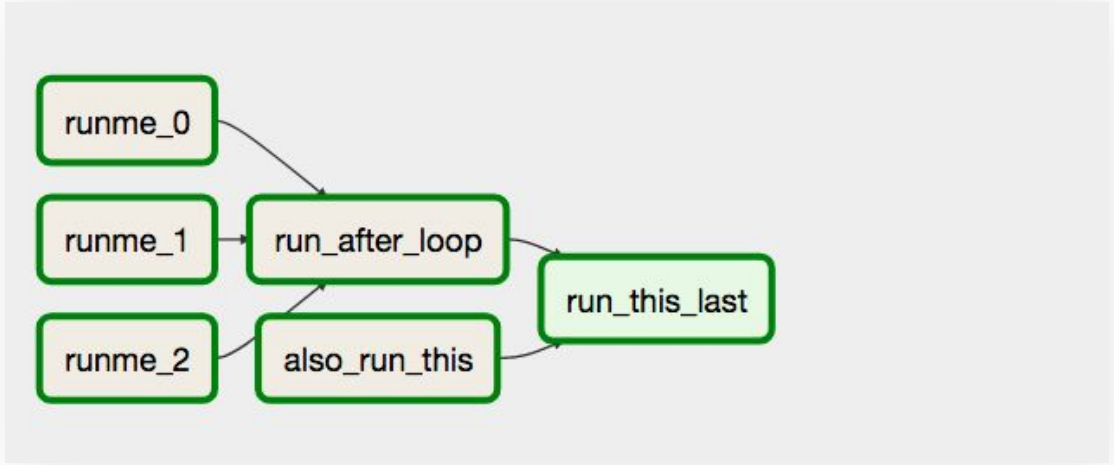
Orchestration tools: major players

Apache Airflow: Overview

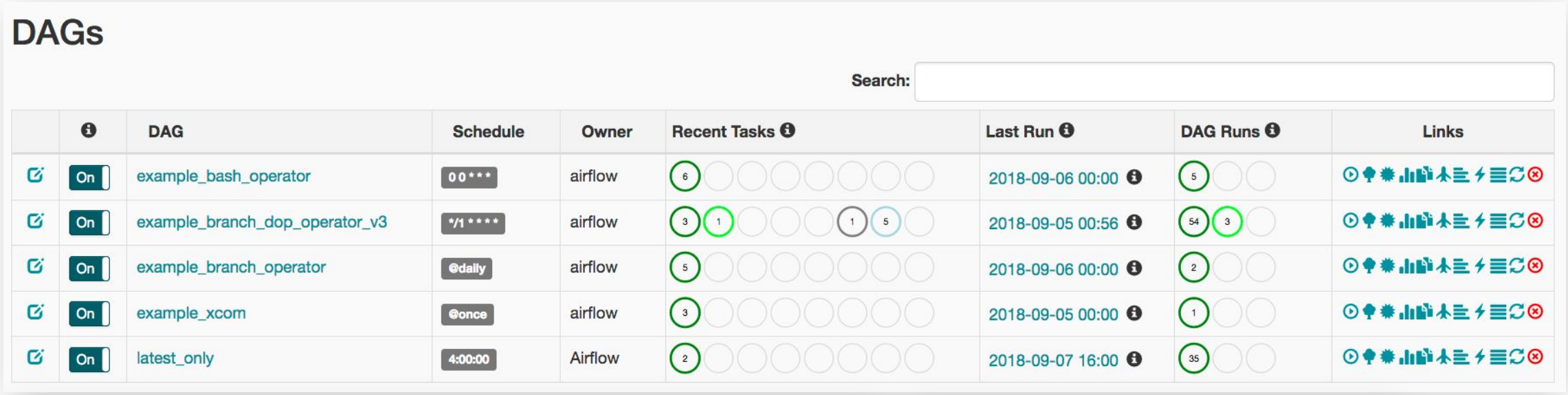
"Platform to programmatically author, schedule, and monitor workflows"

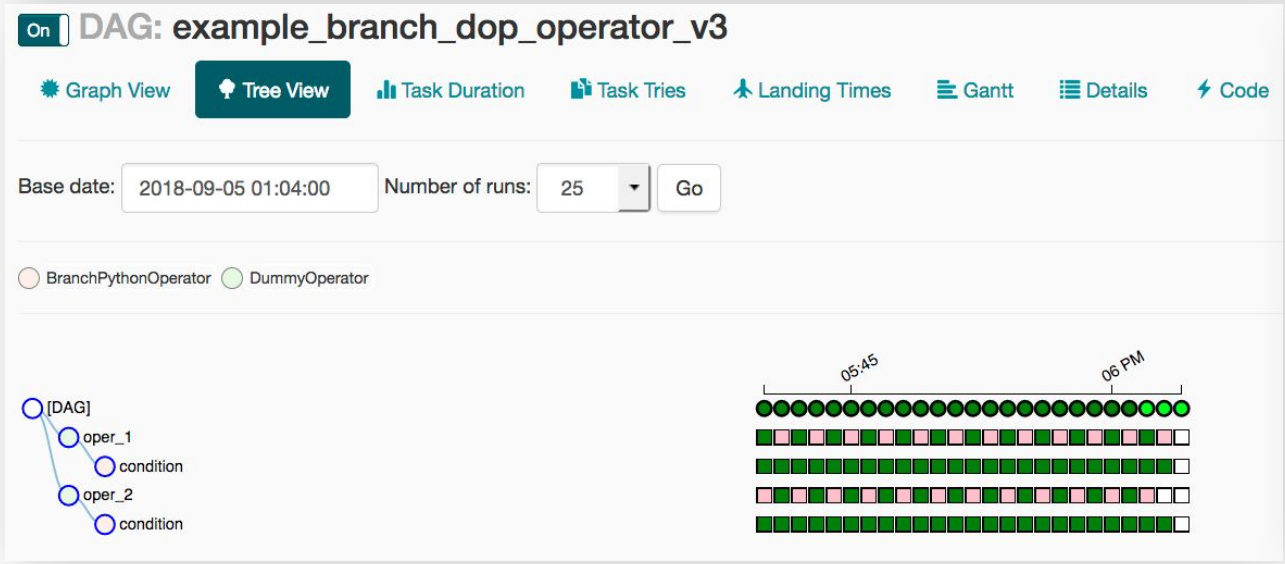
- Vocabulary
  - DAGs (Directed acyclic graphs) = pipelines
  - Tasks, Task instances
  - Operators, Sensors
  - DAG run
  - master, worker nodes
- Features
  - Batch-oriented
  - Define pipelines (DAGs) as Python code
  - Dynamic DAGs support
  - Extensible
  - Parameterizing with Jinja templates
  - Scalable
  - Rich UI

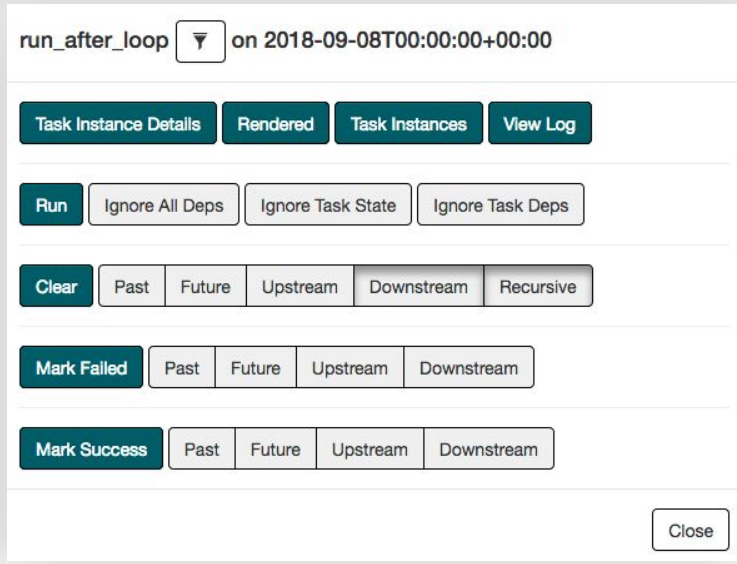




Apache Airflow: Overview

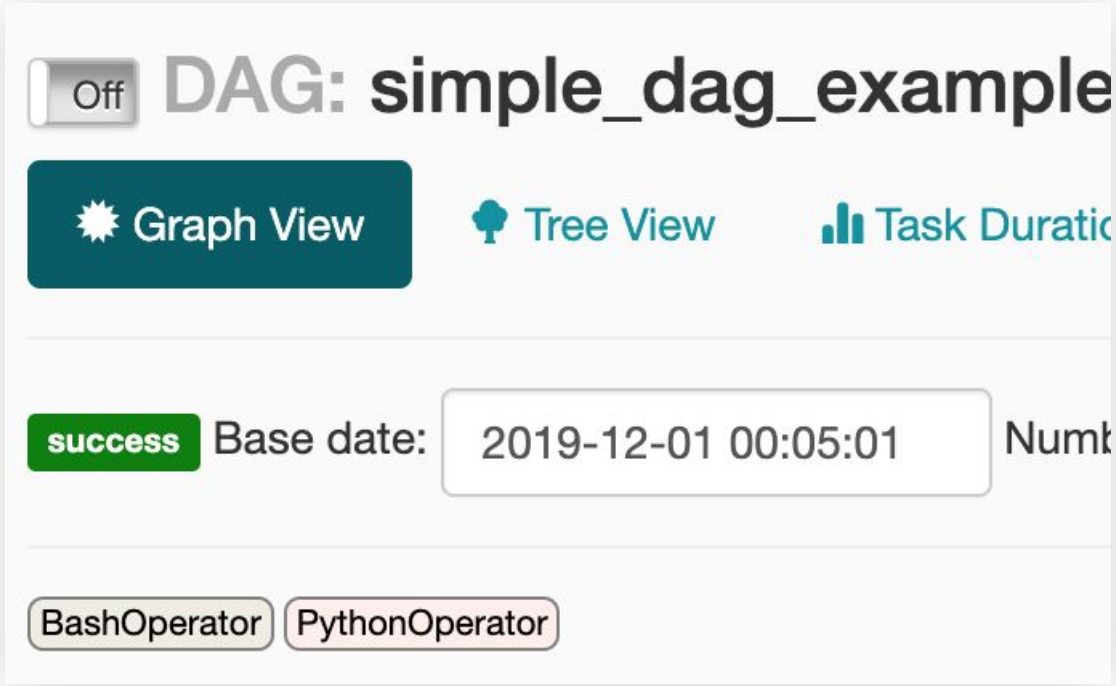




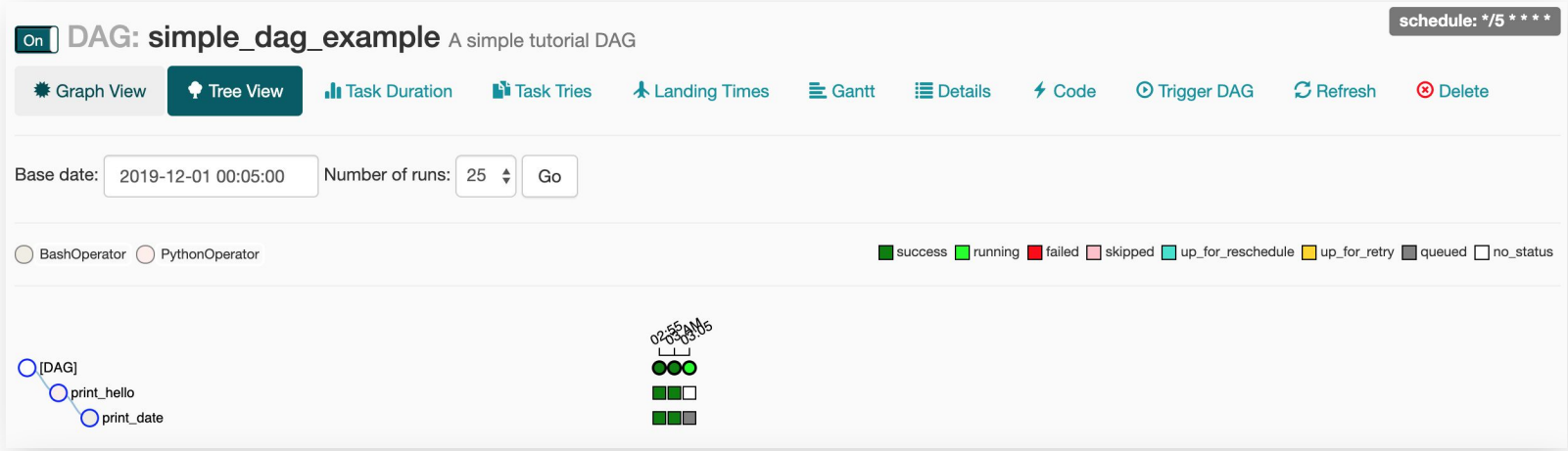


Simple DAG

```python
dag = DAG(
    'simple_dag_example',
    default_args=default_args,
    description='A simple tutorial DAG',
    schedule_interval='*/5 * * * *',
)

t1 = BashOperator(
    task_id='print_date',
    bash_command='date',
    dag=dag
)

def print_hello():
    print('Hello!')

t2 = PythonOperator(
    task_id='print_hello',
    python_callable=print_hello,
    dag=dag
)

t1 >> t2
```



Simple DAG



Log by attempts

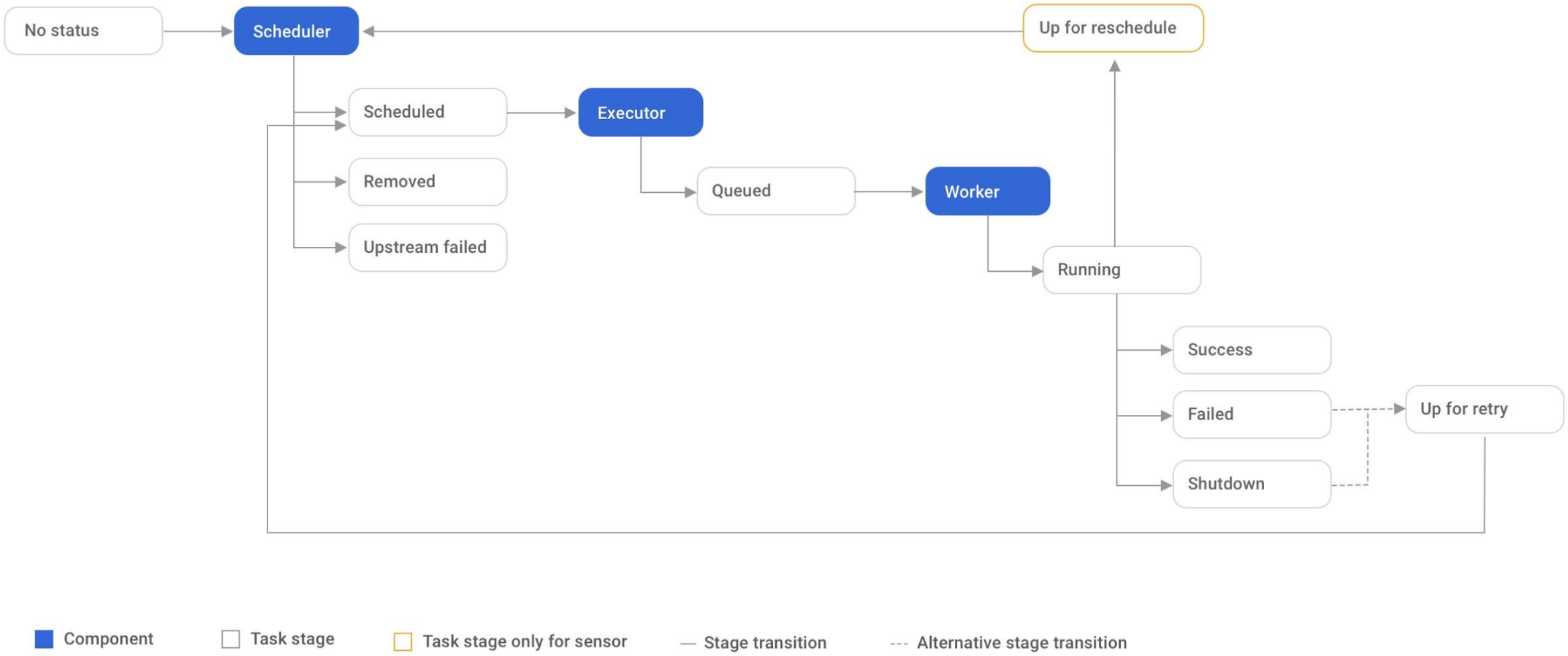
1 | Toggle wrap | Jump to end

```
*** 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 [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.config
[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: 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 l
[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 Dag
[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 retu
```

Apache Airflow: Concepts

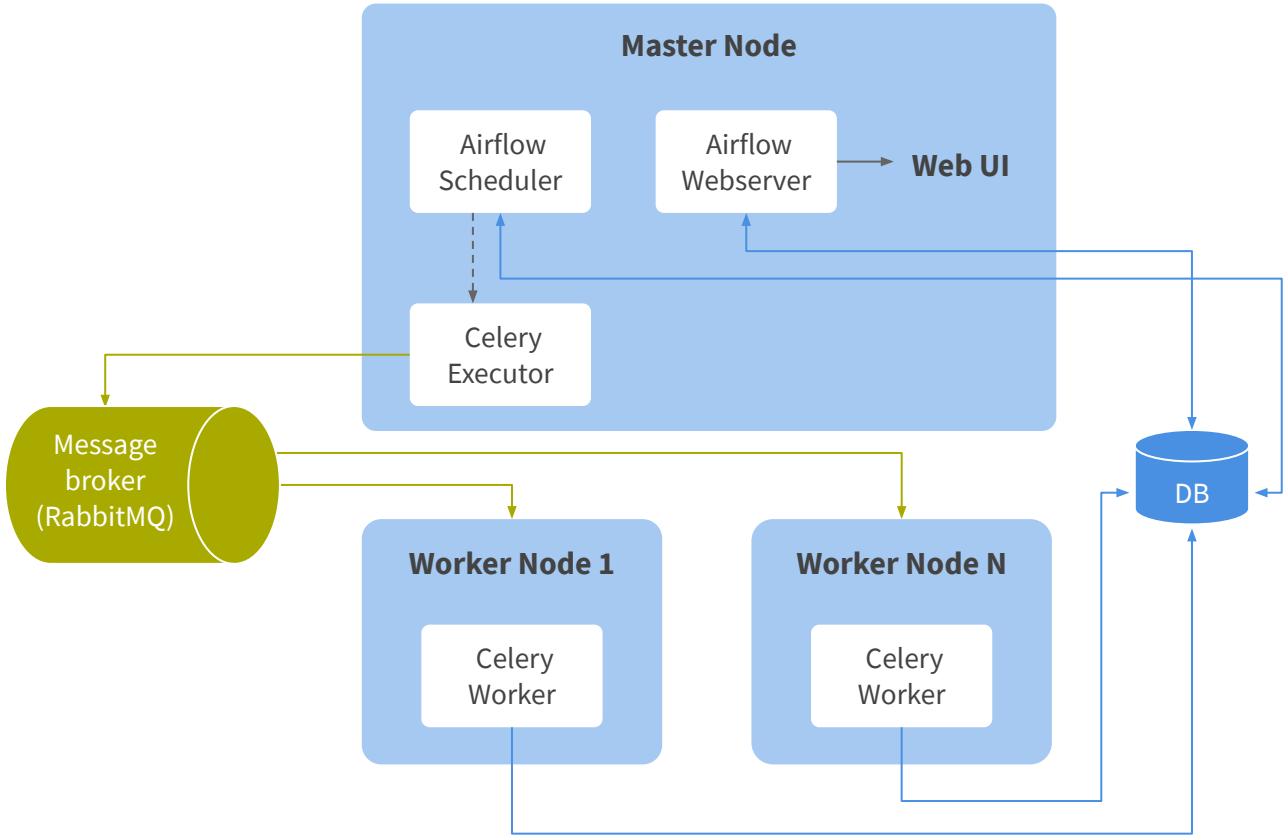
- Operators and Sensors
  - BashOperator
  - PythonOperator
  - PostgresOperator
  - SparkSubmitOperator
  - BaseBranchOperator, TriggerDagRunOperator, SubDagOperator
  - S3PrefixSensor
- Pools, Queues
- Hooks (HDFSHook, SlackHook), Connections, Variables, XComs, etc

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

Grid Dynamics

Practical part

Let's launch a rocket 🚀

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

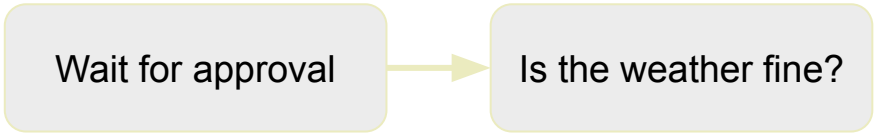
Let's launch a rocket 🚀

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

Wait for approval

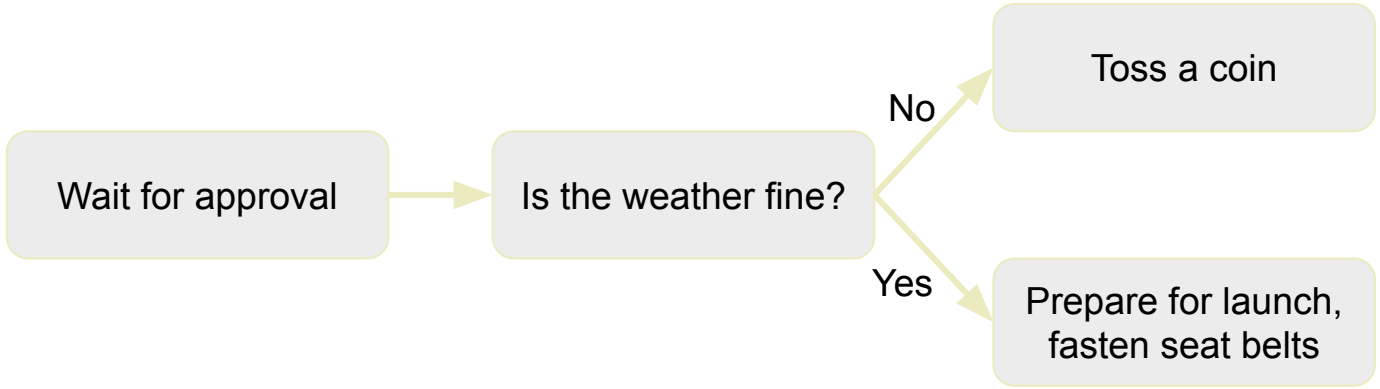
Let's launch a rocket 🚀

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



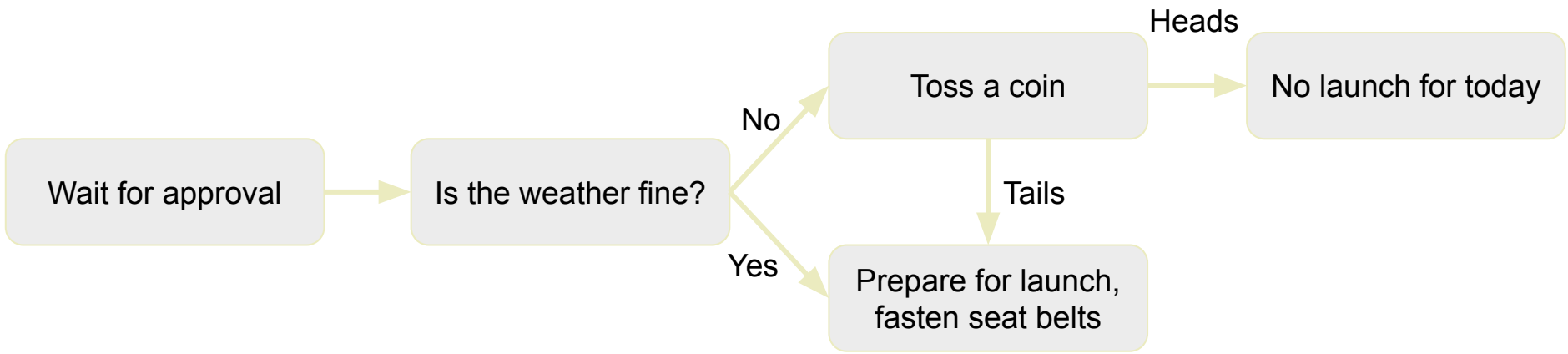
Let's launch a rocket 🚀

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



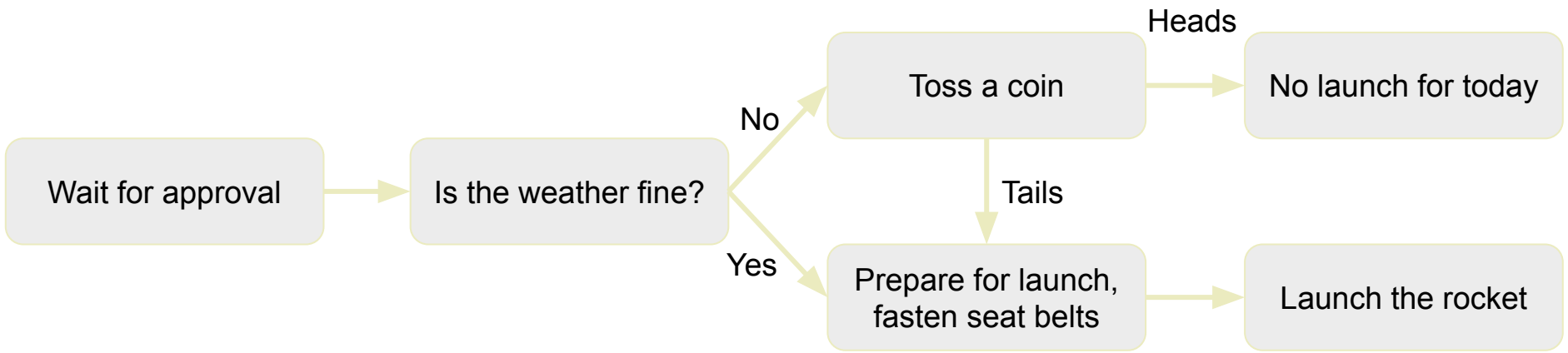
Let's launch a rocket 🚀

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



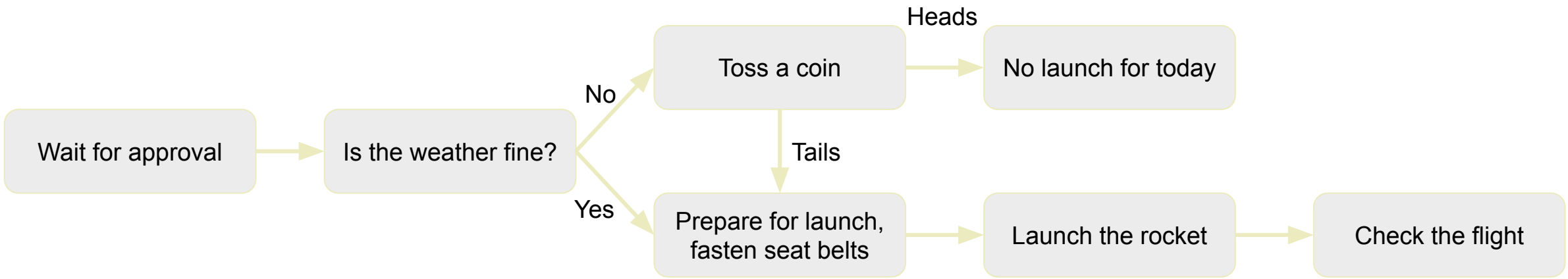
Let's launch a rocket 🚀

- 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

Grid Dynamics

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

Grid Dynamics

Thank you!

www.griddynamics.com