Airflow

Agenda

- Introduction to Airflow
- Architectural Overview
- Deployment Models
- Setting up Airflow for High Availability
- Common issues
- Demo of important features
- Airflow is production - a few use cases
- QA

Introduction

Data engineering workflow management - requirements/challenges

Connect to Data Sources	Hooks in Airflow
Orchestration of various stages in the pipeline dynamically with no downtime	Pipelines are configured via code making the pipelines dynamic
Managing dependencies between stages	XCom
Scheduling jobs	Has support for Calendar schedule and Cron tab scheduling
Monitoring the health of the pipelines	A graphical representation of the DAG instances and Task Instances along with the metrics. Email notifications are natively supported

Introduction

Data engineering workflow management - requirements/challenges

Fault Tolerant pipelines	Retry Mechanism and Backfill utilities in Airflow
Controlling the Pipeline execution externally	PlugIn architecture that helps to add new functionality (REST plugin), TriggerDags
Scaling the Workflow Management System	Scalability: Distribution of Workers and Queues for Task execution
More control to the Management/ Analysts to control the pipelines	Variables for making the changes to the Dags/ Tasks quick and easy
Security	kerberos Support

Getting Started

- DAGs —> Collection/Orchestrated Tasks
- Executors
  - derives from BaseExecutor
  - Three types: Sequential, Local and Celery
- Schedule properties
    - start_date
    - end_date
    - schedule_interval
    - depends_on_past

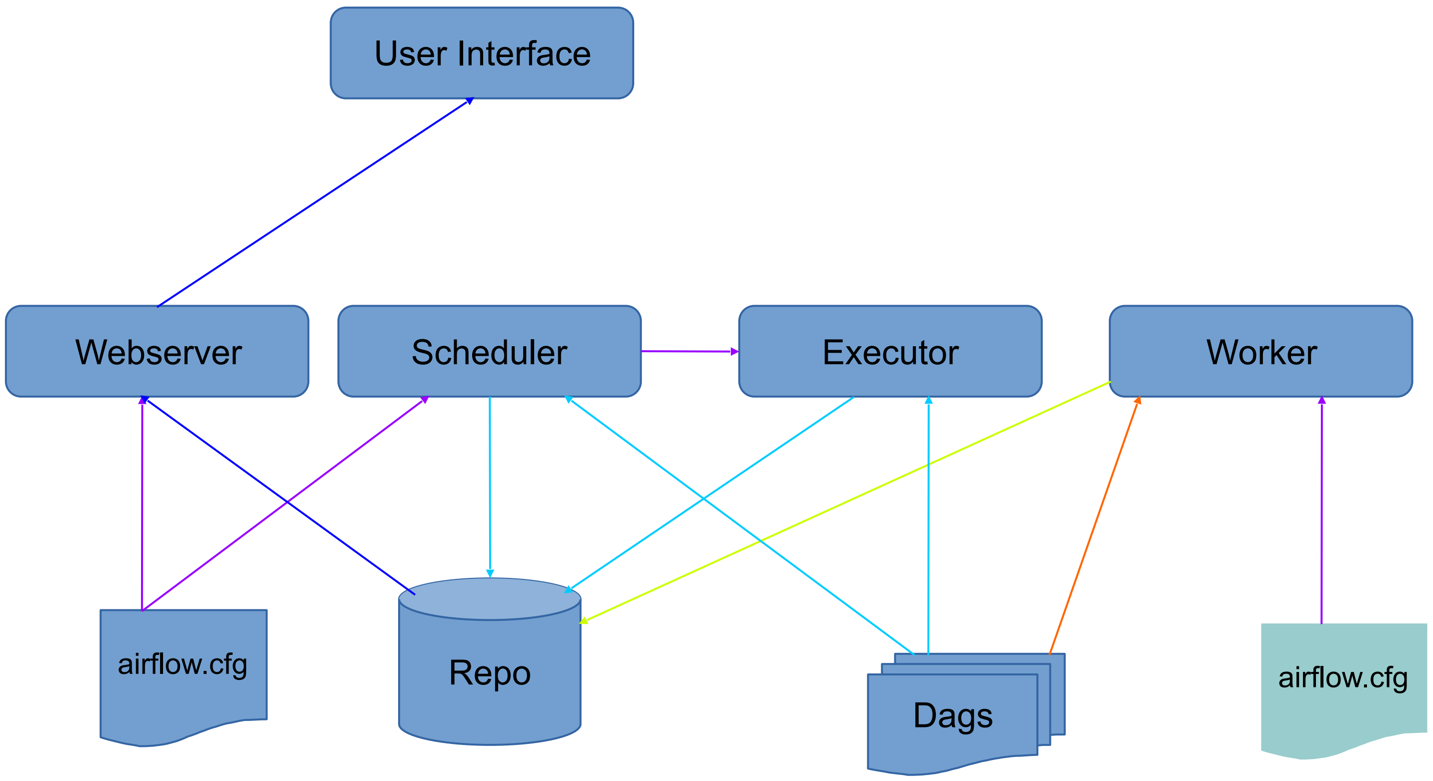
Getting Started

- Operators
  - derives from BaseOperator
  - Three types: Sensors, RemoteExecution, Data Tranfer
    - ExternalTaskSensor
  - trigger_rule --> defines when the task has to be triggered based on the status of upstream tasks.
- Hooks
  - derives from BaseHook.
  - Operators use hooks that actually has API calls to perform an operation
- Task
  - a parameterized Operator
  - Limiting Parallelism: Supports LIMIT on parallelism and PRIORITY_WEIGHT for tasks.
  - Dependencies – upstream()/downstream()

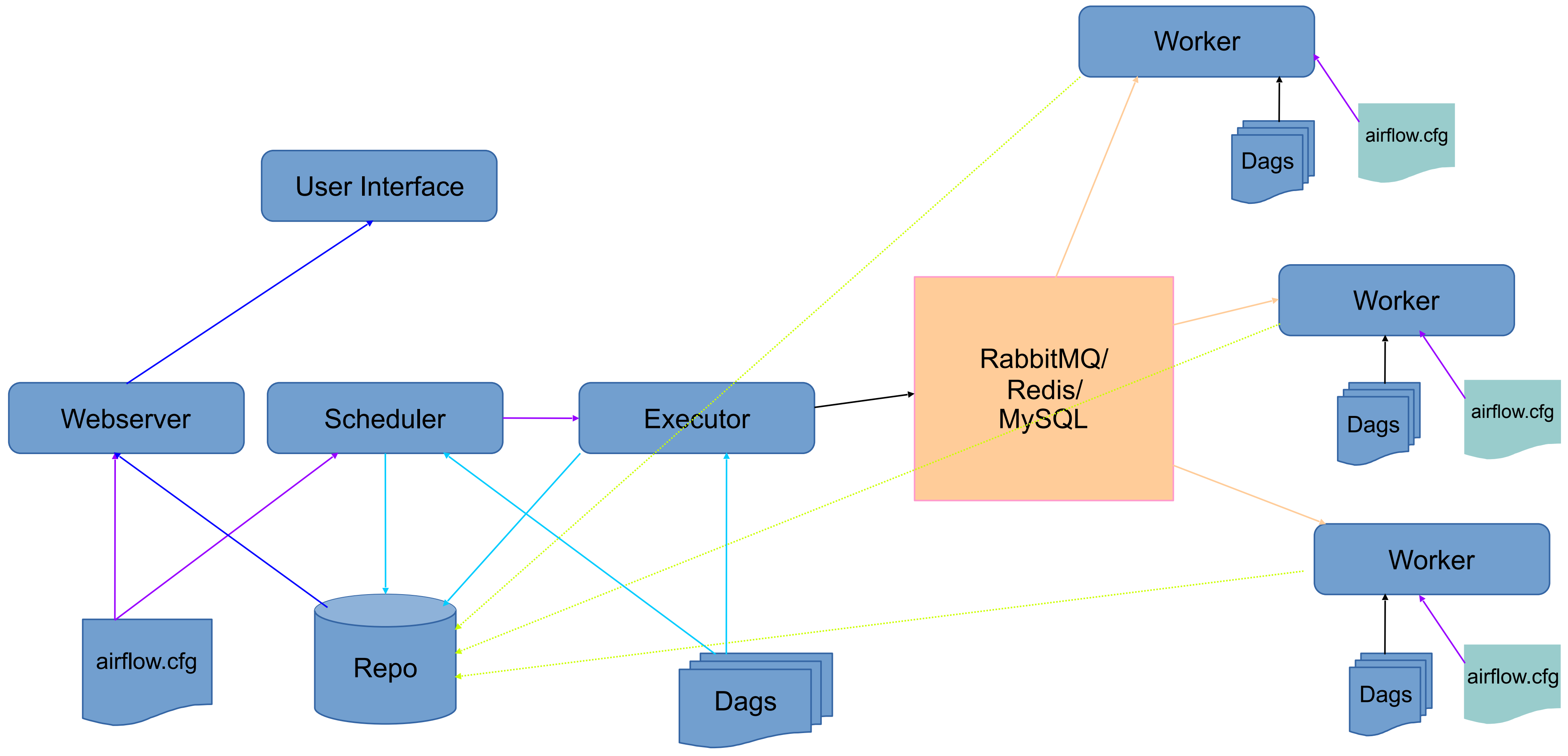
Components of Airflow

- ➔ Configuration file
- ➔ a metadata database (mysql or postgres)
- ➔ the Airflow scheduler
- ➔ a broker (redis or rabbitmq)
- ➔ a set of Airflow worker nodes
- ➔ the Airflow web server

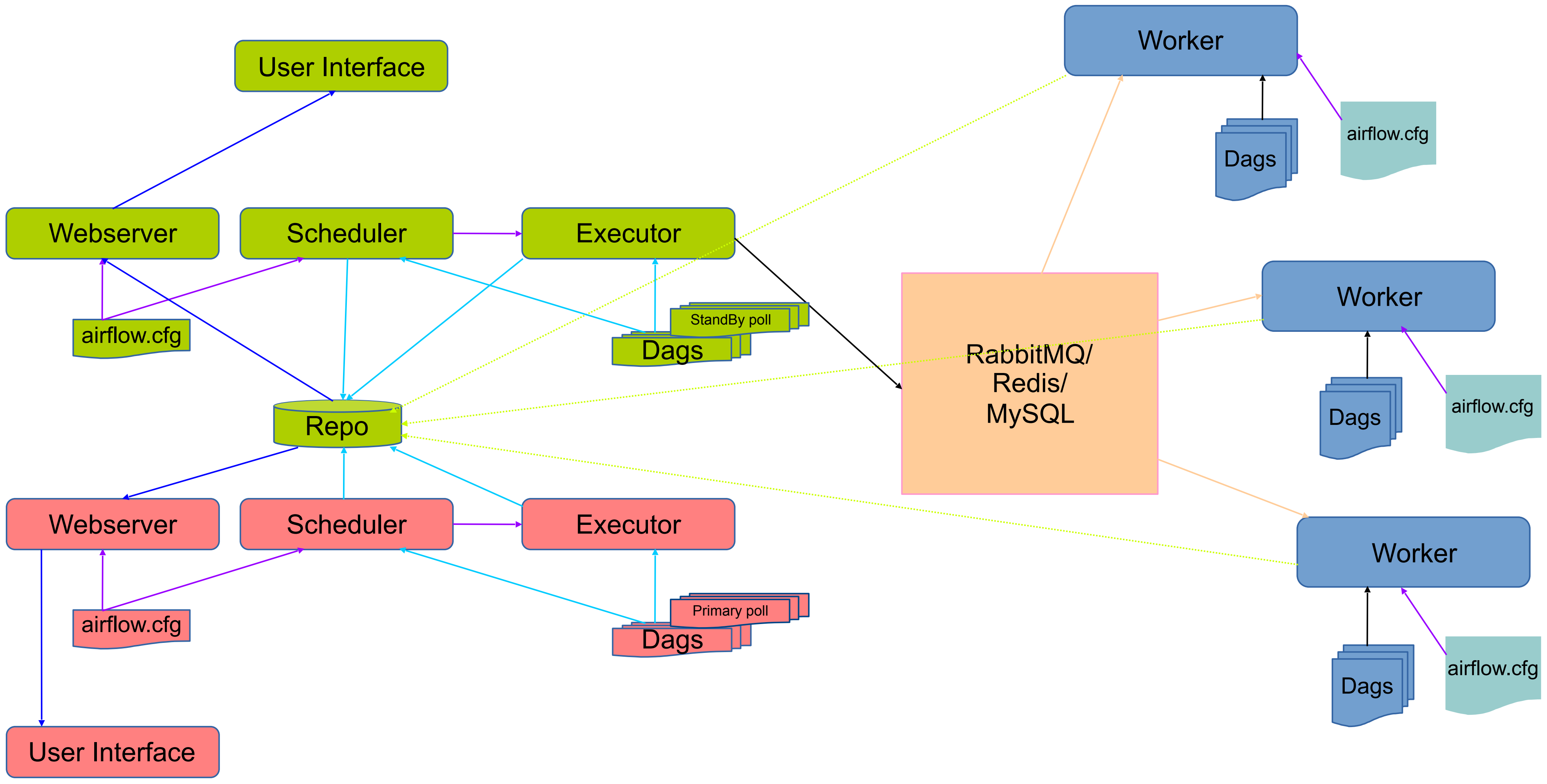
Components of Airflow



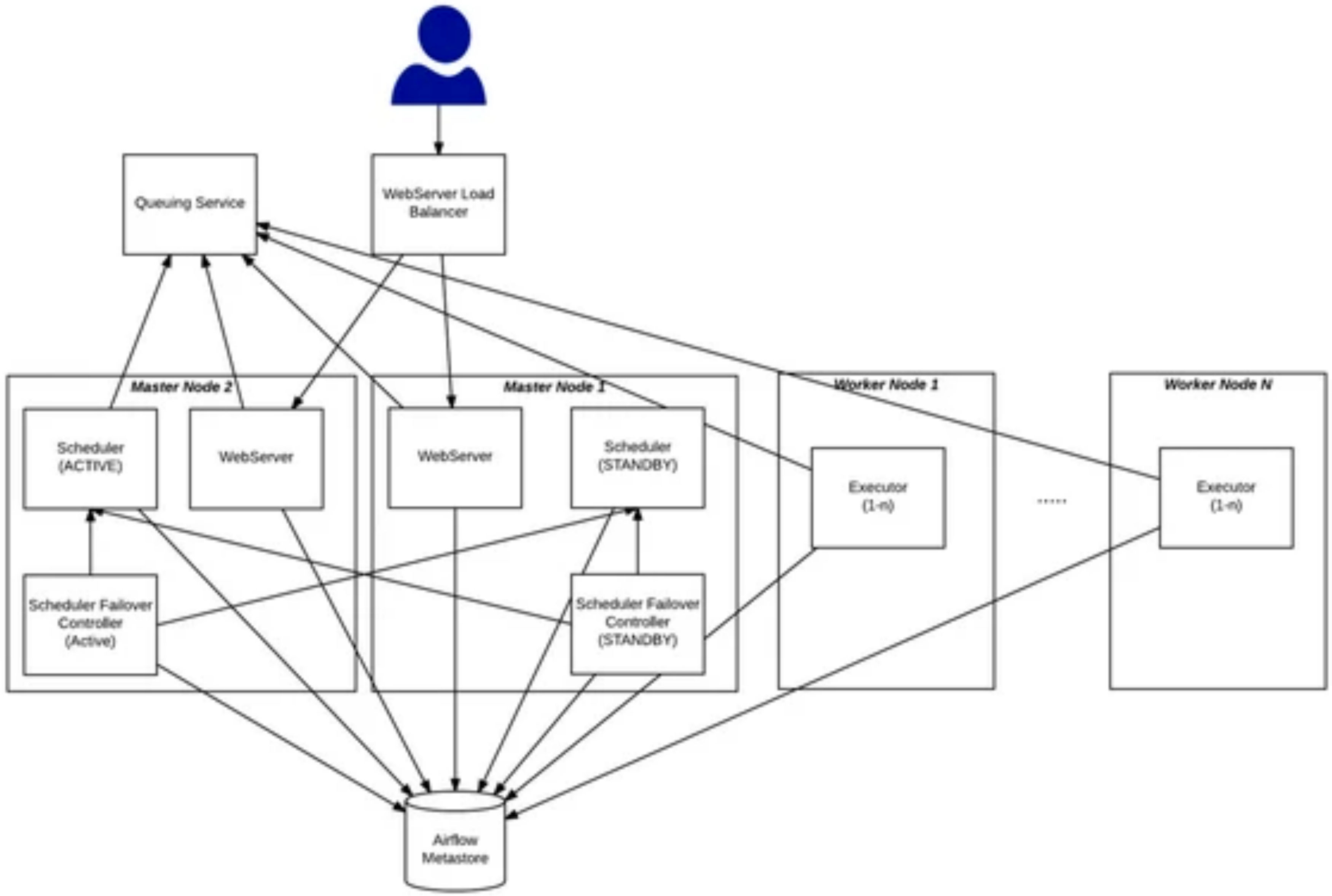
Components of Airflow



Components of Airflow



Components of Airflow



Common Issues

1. Unintended instances when the DAG is not executed for a couple of days.
2. Limit on number of Tasks in a DAG - UI issues
3. If DB goes down, no clue to the end user.
4. Historic instances refer to latest DAG definition

Important points

- RabbitMQ is to be configured for Celery workers to connect to the RabbitMQ server.
- Airflow/Celery needs to be installed on every server.
- The codebase for the tasks and the scripts that are being called from Airflow needs to be present on all the workers.
    - The code based should be in sync to avoid any un-intended results.
- RabbitMQ can be run in standalone mode or cluster mode. Cluster mode would ensure that the queues are distributed.

Demo

Some links

Steps to install RabbitMQ on Centos

http://geek-kb.com/rabbitmq-server-centos-6x/

HA for Rabbit MQ

https://www.rabbitmq.com/blog/2011/10/25/high-availability-in-rabbitmq-solving-part-of-the-puzzle/

nstalling Mysql in Centos

https://www.linode.com/docs/databases/mysql/how-to-install-mysql-on-centos-7

Replicating MySql Database

https://www.digitalocean.com/community/tutorials/how-to-set-up-master-slave-replication-in-mysql