1. Ставится на каждой ноде? Кому в хадупе отправляет задачу?

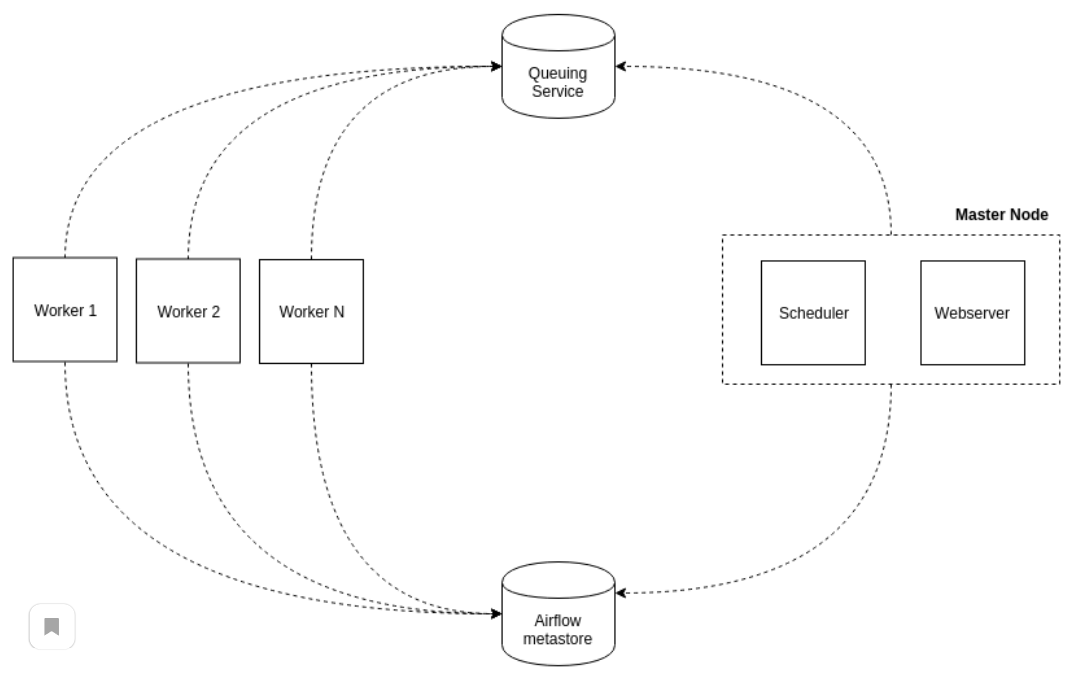
Worker – это место, где запускается наш код и выполняются задачи. Airflow поддерживает несколько экзекьюторов:

* Первый, самый простенький – это SequentialExecutor. Он последовательно запускает прилетающие задачи, а на время их выполнения приостанавливает шедулер.
* LocalExecutor на каждую прилетающую задачу стартует новый процесс, с ним появляется возможность запускать несколько задач в параллель, поэтому LocalExecutor чуточку лучше предыдущего экзекьютора. Есть один нюанс: если в качестве базы метаданных используется что-то типа однопоточного SQLite, ваш LocalExecutor превращается в SequentialExecutor.
* CeleryExecutor позволяет иметь несколько воркеров, работающих на разных машинах. Celery – это распределенная очередь задач, которая под капотом использует RabbitMQ или Redis. При запуске воркера ему можно указать названия очередей, из которых он будет принимать задачи от шедулера.
* DaskExecutor запускает задачу с помощью Dask – библиотеки для параллельных вычислений.
* KubernetesExecutor на каждую задачу запускает новый pod в Kubernetes.
* DebugExecutor создан для запуска и отладки пайплайнов из IDE.

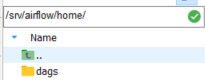
**Мультинодовый кластер Airflow**

демоны распределены по всем рабочим нодам. Поскольку веб-сервер и планировщик будут установлены на главной ноде, а рабочие будут установлены на каждом отдельном рабочей ноде, поэтому он может хорошо масштабироваться как по горизонтали, так и по вертикали. Чтобы использовать этот режим архитектуры, необходимо настроить Airflow с помощью CeleryExecutor.

Серверную часть Celery необходимо настроить для включения режима CeleryExecutor в архитектуре Airflow. Популярными фреймворками / приложениями для бэкэнда Celery являются Redis и RabbitMQ. RabbitMQ — это брокер сообщений. Каждая очередь в RabbitMQ опубликована с событиями / сообщениями в виде команд задач, работники Celery будут извлекать команды задач из каждой очереди и выполнять их как действительно распределенные и параллельные способы.



**Практика**



<https://github.com/vvandriichuk/airflow_docker_start> - запуск эирфлоу

docker-compose up --build –d -?

**Параметры**

schedule\_interval – параметр для обозначения времени запуска дага. Можно указать в cron-time:

“0/5 9 \* \* \*” – каждые 5 минут в течении девятого часа каждый день.

Или “once”, “hourly” и т.д.

Catchup – если true, то при запуске дага с каким-то расписанием то, будет выполнить все не выполненные запуски начиная с start\_date.

**Trigger**

Version: 1.10

С помощью bash команды через BashOpertor можно вызвать другой DAG:

bash\_command="airflow trigger\_dag --conf '{\"conf1\": \"value1\"}' nameAdmin ",

nameAdmin – название DAG

Version: 2

Через TRIGGERDAGRUNOPERATOR