


# 如何部署一个健壮的 apache-airflow 调度系统

 2 2018.09.17 07:09:41 字数 2483 阅读 17482

之前介绍过的 apache-airflow 系列文章

[任务调度神器 airflow 之初体验](#)

[airflow 安装，部署，填坑](#)

[airflow 配置 CeleryExecutor](#)

介绍了如何安装、配置、及使用，接下来介绍如何在跨多个节点来安装部署 apache-airflow：本文主要介绍以下几点：

- airflow 的守护进程
- airflow 单节点部署
- airflow 多节点（集群）部署
- airflow 集群部署的具体步骤

集群部署将为您的 apache-airflow 系统带来更多的计算能力和高可用性。

## airflow 的守护进程

airflow 系统在运行时有许多守护进程，它们提供了 airflow 的全部功能。守护进程包括 Web 服务器-webserver、调度程序-scheduler、执行单元-worker、消息队列监控工具-Flower等。下面是 apache-airflow 集群、高可用部署的主要守护进程。

### webserver

webserver 是一个守护进程，它接受 HTTP 请求，允许你通过 Python Flask Web 应用程序与 airflow 进行交互，webserver 提供以下功能：

- 中止、恢复、触发任务。
- 监控正在运行的任务，断点续跑任务。
- 执行 ad-hoc 命令或 SQL 语句来查询任务的状态，日志等详细信息。
- 配置连接，包括不限于数据库、ssh 的连接等。

webserver 守护进程使用 gunicorn 服务器（相当于 java 中的 tomcat ）处理并发请求，可通过修改{AIRFLOW\_HOME}/airflow.cfg文件中 workers 的值来控制处理并发请求的进程数。例如：

```
1 | workers = 4 #表示开启4个gunicorn worker(进程)处理web请求
```

启动 webserver 守护进程：

```
1 | $ airfow webserver -D
```

### scheduler



somenzz

关注

拥有256钻 (约29.12元)

启动的 scheduler 守护进程：

```
1 | $ airflow scheduler -D
```

worker

worker 是一个守护进程，它启动 1 个或多个 Celery 的任务队列，负责执行具体的 DAG 任务。

当设置 airflow 的 executors 设置为 CeleryExecutor 时才需要开启 worker 守护进程。推荐你在生产环境使用 CeleryExecutor：

```
1 | executor = CeleryExecutor
```

启动一个 worker守护进程，默认的队列名为 default:

```
1 | $ airflow worker -D
```

flower

flower 是一个守护进程，用于监控 celery 消息队列。启动守护进程命令如下：

```
1 | $ airflow flower -D
```

默认的端口为 5555，您可以在浏览器地址栏中输入 "<http://hostip:5555>" 来访问 flower，对 celery 消息队列进行监控。

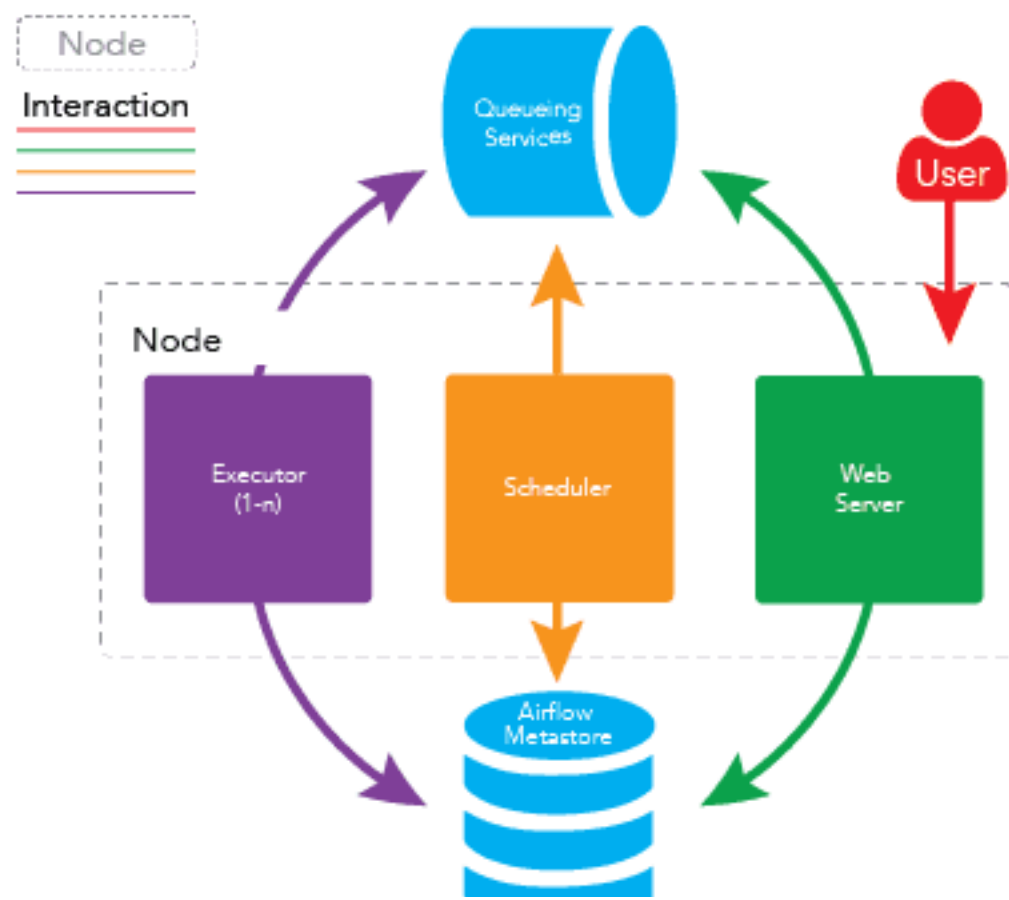
airflow 的守护进程是如何一起工作的？

需要注意的是 airflow 的守护进程彼此之间是独立的，他们并不相互依赖，也不相互感知。每个守护进程在运行时只处理分配到自己身上的任务，他们在一起运行时，提供了 airflow 的全部功能。

1. 调度器 scheduler 会间隔性的去轮询元数据库（Metastore）已注册的 DAG（有向无环图，可理解为作业流）是否需要被执行。如果一个具体的 DAG 根据其调度计划需要被执行，scheduler 守护进程就会先在元数据库创建一个 DagRun 的实例，并触发 DAG 内部的具体 task（任务，可以这样理解：DAG 包含一个或多个task），触发其实并不是真正的去执行任务，而是推送 task 消息至消息队列（即 broker）中，每一个 task 消息都包含此 task 的 DAG ID，task ID，及具体需要被执行的函数。如果 task 是要执行 bash 脚本，那么 task 消息还会包含 bash 脚本的代码。
2. 用户可能在 webserver 上来控制 DAG，比如手动触发一个 DAG 去执行。当用户这样做的时候，一个DagRun 的实例将在元数据库被创建，scheduler 使同 #1 一样的方法去触发 DAG 中具体的 task。
3. worker 守护进程将会监听消息队列，如果有消息就从消息队列中取出消息，当取出任务消息时，它会更新元数据中的 DagRun 实例的状态为正在运行，并尝试执行 DAG 中的 task，如果 DAG 执行成功，则更新任 DagRun 实例的状态为成功，否则更新状态为失败。

airflow 单节点部署

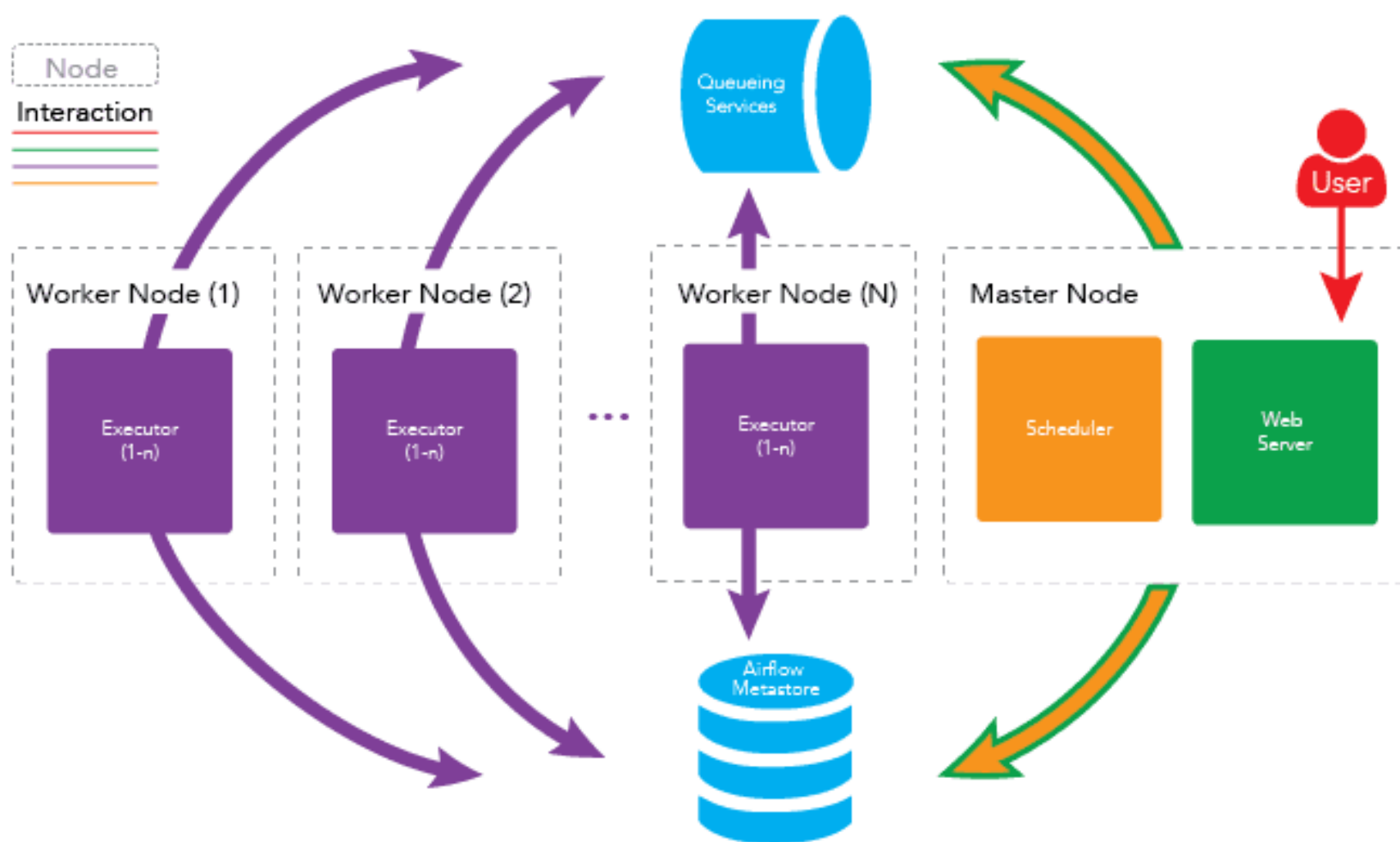
将以所有上守护进程运行在同一台机器上即可完成 airflow 的单结点部署，架构如下图所示



airflow 单节点部署

## airflow 多节点（集群）部署

在稳定性要求较高的场景，如金融交易系统中，一般采用集群、高可用的方式来部署。Apache Airflow 同样支持集群、高可用的部署，airflow 的守护进程可分布在多台机器上运行，架构如下图所示：



airflow 集群部署

### 这样做有以下好处

- 高可用

如果一个 worker 节点崩溃或离线时，集群仍可以被控制的，其他 worker 节点的任务仍会被执行。

- 分布式处理

如果你的工作流中有一些内存密集型的任务，任务最好是分布在多台机器上运行以便得到更快的执行。

### 扩展 worker 节点

• 水平扩展

你可以通过向集群中添加更多 worker 节点来水平地扩展集群，并使这些新节点指向同一个元数据库，从而分发处理过程。由于 worker 不需要在任何守护进程注册即可执行任务，因此所以 worker 节点可以在不停机，不重启服务下的情况进行扩展，也就是说可以随时扩展。

• 垂直扩展

你可以通过增加单个 worker 节点的守护进程数来垂直扩展集群。可以通过修改 airflow 的配置文件-{AIRFLOW\_HOME}/airflow.cfg 中 celeryd\_concurrency 的值来实现，例如：

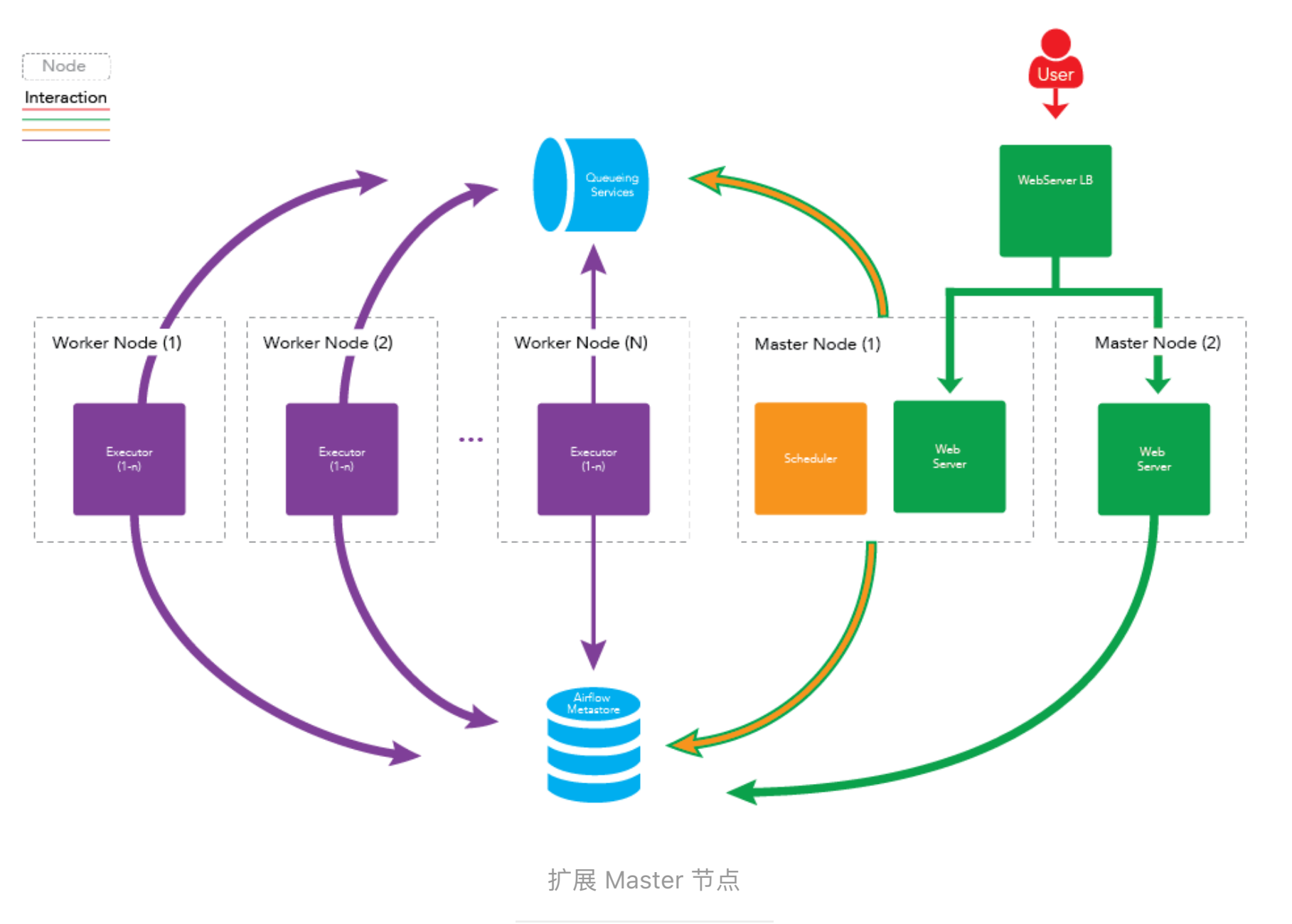
```
1 | celeryd_concurrency = 30
```

您可以根据实际情况，如集群上运行的任务性质，CPU 的内核数量等，增加并发进程的数量以满足实际需求。

扩展 Master 节点

您还可以向集群中添加更多主节点，以扩展主节点上运行的服务。您可以扩展 webserver 守护进程，以防止太多的 HTTP 请求出现在一台机器上，或者您想为 webserver 的服务提供更高的可用性。需要注意的一点是，每次只能运行一个 scheduler 守护进程。如果您有多个 scheduler 运行，那么就有可能一个任务被执行多次。这可能会导致您的工作流因重复运行而出现一些问题。

下图为扩展 Master 节点的架构图：



看到这里，可能有人会问，**scheduler** 不能同时运行两个，那么运行 **scheduler** 的节点一旦出了问题，任务不就完全不运行了吗？

答案：这是个非常好的问题，不过已经有解决方案了，我们可以在两台机器上部署 scheduler，只运行一台机器上的 scheduler 守护进程，一旦运行 scheduler 守护进程的机器出现故障，立刻启动另一台机器上的 scheduler 即可。我们可以借助第三方组件 **airflow-scheduler-failover-controller** 实现 scheduler 的高可用。

具体步骤如下所示：

1. 下载 failover



```
1 | git clone https://github.com/teamclairvoyant/airflow-scheduler-failover-controller
```

## 2. 使用 pip 进行安装

```
1 | cd{AIRFLOW_FAILOVER_CONTROLLER_HOME}  
2 | pip install -e .
```

## 3. 初始化 failover

```
1 | scheduler_failover_controller init
```

注：初始化时，会向airflow.cfg中追加内容，因此需要先安装 airflow 并初始化。

## 4. 更改 failover 配置

```
1 | scheduler_nodes_in_cluster= host1,host2
```

注:host name 可以通过scheduler\_failover\_controller get\_current\_host命令获得

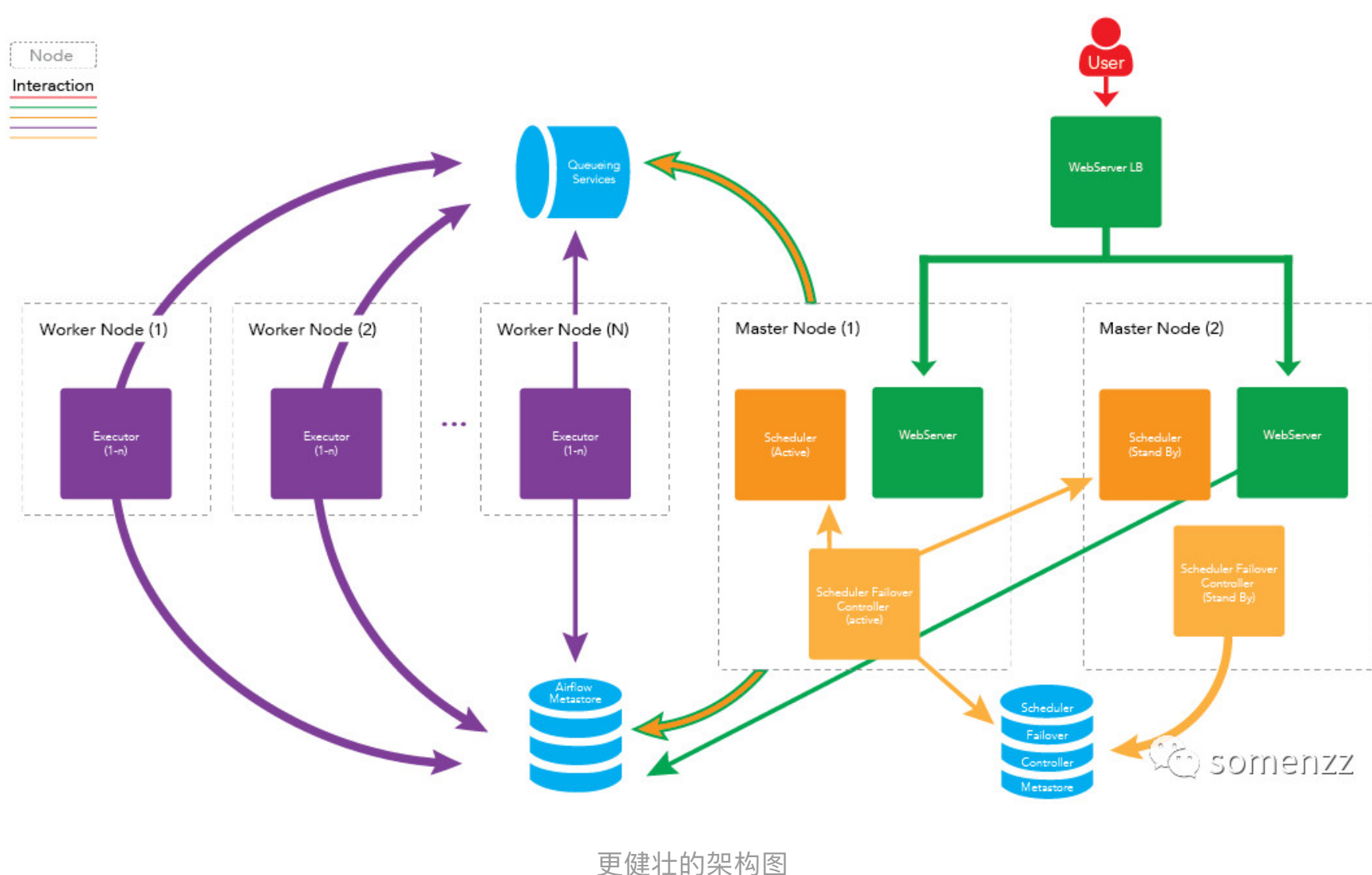
## 5. 配置安装 failover 的机器之间的免密登录，配置完成后，可以使用如下命令进行验证：

```
1 | scheduler_failover_controller test_connection
```

## 6. 启动 failover

```
1 | scheduler_failover_controller start
```

因此更健壮的架构图如下所示：



## 队列服务及元数据库(Metestore)的高可用。

- 队列服务取决于使用的消息队列是否可以高用可部署，如 RabbitMQ 和 Redis。

RabbitMQ 集群并配置Mirrored模式

见：<http://blog.csdn.net/u010353408/article/details/77964190>

- 元数据库(Metestore) 取决于所使用的数据库，如 Mysql 等。  
Mysql 做主从备份见：<http://blog.csdn.net/u010353408/article/details/77964157>

## airflow 集群部署的具体步骤

### 前提条件

- 节点运行的守护进程如下：
  - master1
    - 运行: webserver, scheduler
  - master2
    - 运行：webserver
  - worker1
    - 运行：worker
  - worker2
    - 运行：worker
- 队列服务处于运行中. (RabbitMQ, Redis, etc)
  - 安装 RabbitMQ 方法参见: <http://site.clairvoyantsoft.com/installing-rabbitmq/>
  - 如果正在使用 RabbitMQ, 推荐 RabbitMQ 也做成高可用的集群部署，并为 RabbitMQ 实例配置负载均衡。

### 步骤

- 在所有需要运行守护进程的机器上安装 Apache Airflow。具体安装方法可参考 [airflow 安装，部署，填坑](#)
- 修改 {AIRFLOW\_HOME}/airflow.cfg 文件，确保所有机器使用同一份配置文件。

- 修改 Executor 为 CeleryExecutor

```
1 | executor = CeleryExecutor
```

- 指定元数据库（metestore）

```
1 | sql_alchemy_conn = mysql://{USERNAME}:{PASSWORD}@{MYSQL_HOST}:3306/airflow
```

- 设置中间人（broker）

如果使用 RabbitMQ

```
1 | broker_url = amqp://guest:guest@{RABBITMQ_HOST}:5672/
```

如果使用 Redis

```
1 | broker_url = redis://{REDIS_HOST}:6379/0 #使用数据库 0
```

- 设定结果存储后端 backend

```
1 | celery_result_backend = db+mysql://{USERNAME}:{PASSWORD}@{MYSQL_HOST}:3306/airflow #当然您
```

- 3. 在 master1 和 master2 上部署您的工作流（DAGs）。
- 4. 在 master 1，初始 airflow 的元数据库

```
1 | $ airflow initdb
```

- 5. 在 master1, 启动相应的守护进程

```
1 | $ airflow webserver
```

```
1 | $ airflow scheduler
```

- 6. 在 master2，启动 Web Server

```
1 | $ airflow webserver
```

- 7. 在 worker1 和 worker2 启动 worker

```
1 | $ airflow worker
```

- 8. 使用负载均衡处理 webserver  
可以使用 nginx，AWS 等服务器处理 webserver 的负载均衡，不在此详述

至此，所有均已集群或高可用部署，apache-airflow 系统已坚不可摧。

官方文档如下：  
Documentation: <https://airflow.incubator.apache.org/>  
Install Documentation: <https://airflow.incubator.apache.org/installation.html>  
GitHub Repo: <https://github.com/apache/incubator-airflow>

(完)

如果您对文章感兴趣，请关注微信公众号搜索 somenzz 关注，或扫下方二维码关注



扫码关注



somenzz  
拥有256钻 (约29.12元)

关注

"小礼物走一走，来简书关注我"

赞赏

被以下专题收入，发现更多相似内容



Amazing...



Python学习之道



Apache



airflow

推荐阅读

更多精彩内容 >

### 深度解析 | 基于DAG的分布式任务调度平台：Maat

阿里妹导读：搜索中台建设过程中，单个系统不再能满足复杂业务的需求，更多时候需要多个子系统互相协作，异步地按照指定流...



高级java架构师

### 深度解析 | 基于DAG的分布式任务调度平台：Maat

背景 什么是Maat？ Maat是一个基于开源项目Airflow的流程调度系统，它支持用户自定义地组装流程节点，流...



阿里云云栖社区

### 浅谈调度工具——Airflow

本文将介绍 Airflow 这一款优秀的调度工具。主要包括 Airflow 的服务构成、Airflow 的 Web...



泛金融技术

### yarn应用场景基本架构和资源调度

YarnYarn产生背景：Yarn直接来自于MR1.0MR1.0 问题：采用的是master slave结构，ma...



时待吾

### 冰解的破-spark

Apache Spark 是专为大规模数据处理而设计的快速通用的计算引擎。Spark是UC Berkeley AM...



大佛爱读书