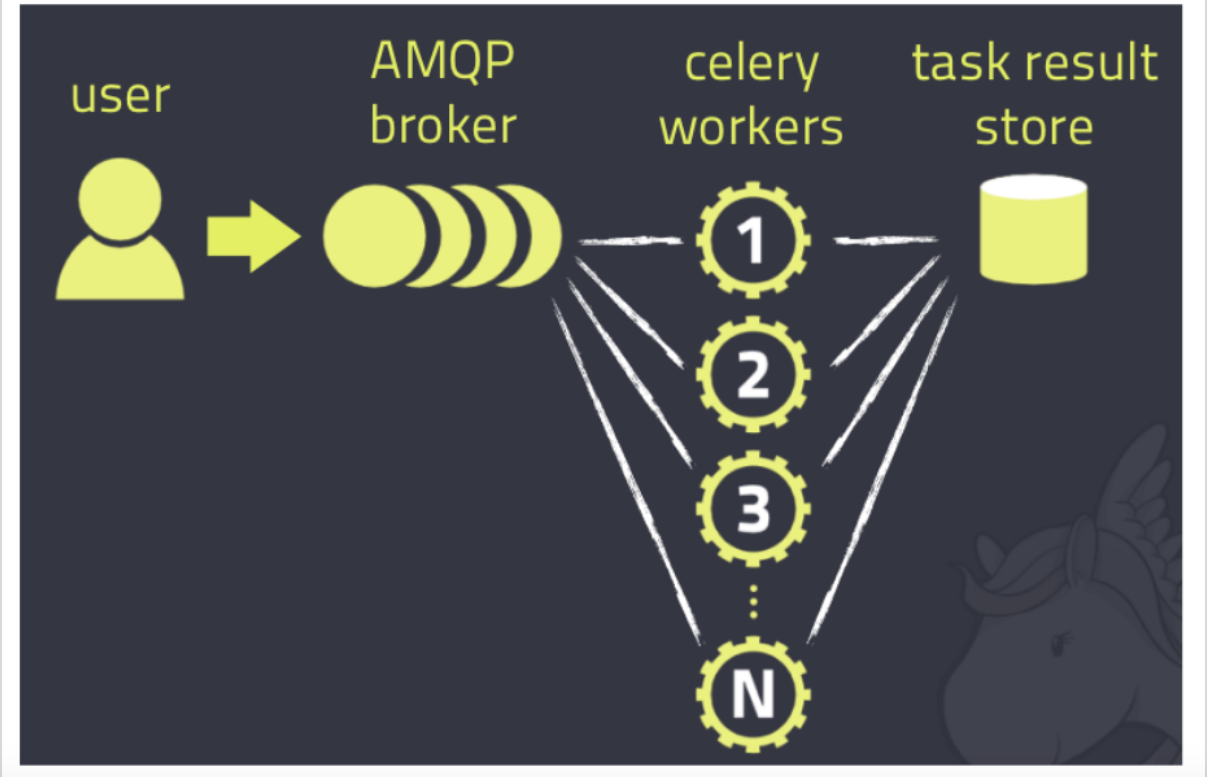
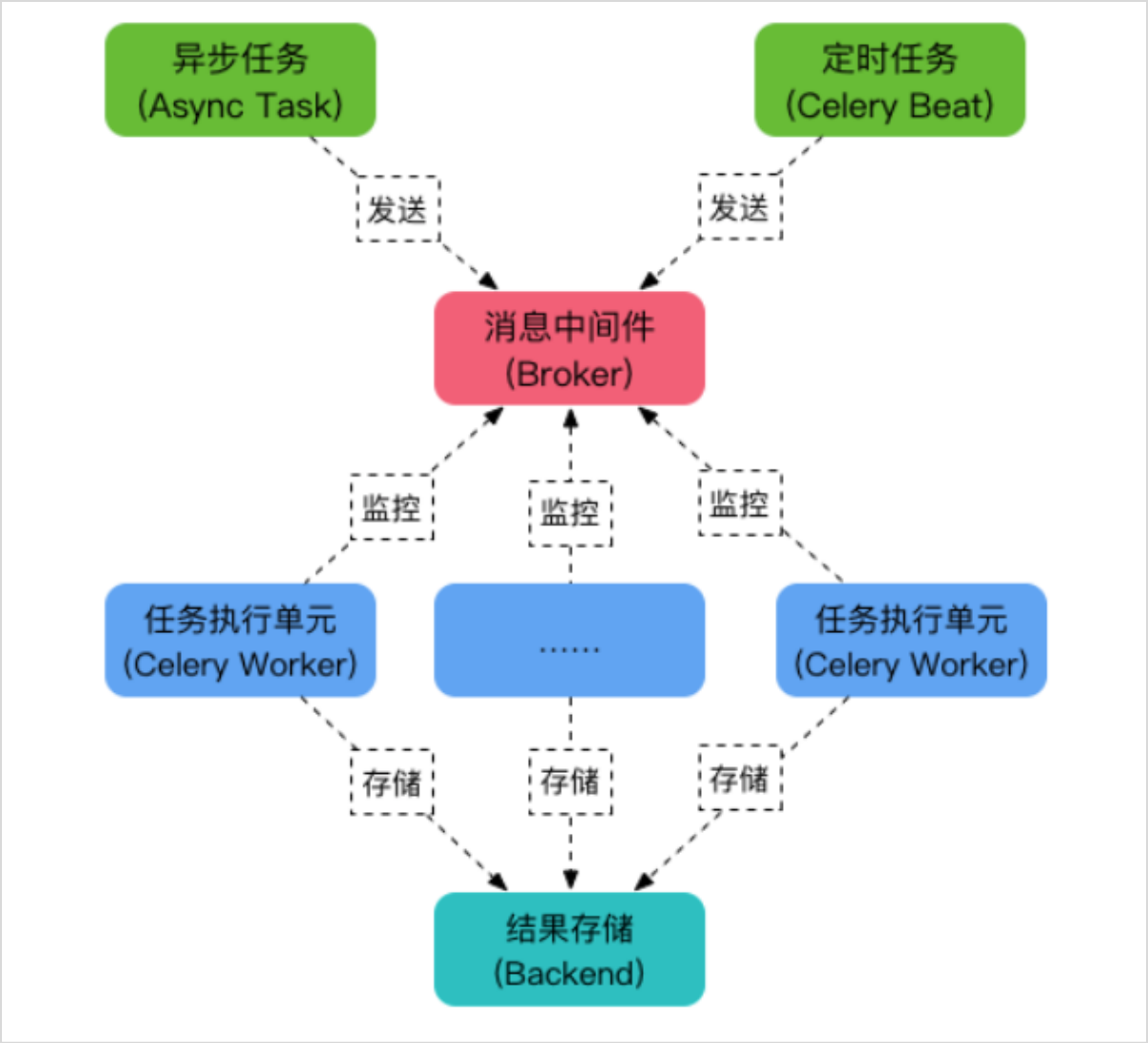
Celery 是 Distributed Task Queue，分布式任务队列，分布式决定了可以有多个 worker 的存在，队列表示其是异步操作，即存在一个产生任务提出需求的工头，和一群等着被分配工作的码农。

在 Python 中定义 Celery 的时候，我们要引入 Broker，中文翻译过来就是“中间人”的意思，在这里 Broker 起到一个中间人的角色。在工头提出任务的时候，把所有的任务放到 Broker 里面，在 Broker 的另外一头，一群码农等着取出一个个任务准备着手做。

什么是backend?
通常程序发送的消息，发完就完了，可能都不知道对方时候接受了。为此，celery实现了一个backend，用于存储这些消息以及celery执行的一些消息和结果。





Celery 主要包含以下几个模块：

任务模块 Task

包含异步任务和定时任务。其中，异步任务通常在业务逻辑中被触发并发往任务队列，而定时任务由 Celery Beat 进程周期性地将任务发往任务队列。

消息中间件 Broker

Broker，即为任务调度队列，接收任务生产者发来的消息（即任务），将任务存入队列。Celery 本身不提供队列服务，官方推荐使用 RabbitMQ 和 Redis 等。

任务执行单元 Worker

Worker 是执行任务的处理单元，它实时监控消息队列，获取队列中调度的任务，并执行它。

任务结果存储 Backend

Backend 用于存储任务的执行结果，以供查询。同消息中间件一样，存储也可使用 RabbitMQ, redis 和 MongoDB 等。

MQ全称为Message Queue, 消息队列（MQ）是一种应用程序对应用程序的通信方法。MQ是消费-生产者模型的一个典型的代表，一端往消息队列中不断写入消息，而另一端则可以读取队列中的消息。

RabbitMQ是MQ的一种

安装RabbitMQ
sudo apt-get install rabbitmq-server

启动RabbitMQ
sudo rabbitmq-server -detached

停止RabbitMQ
sudo rabbitmqctl stop

设置RabbitMQ
sudo rabbitmqctl add_user carmack 123456

sudo rabbitmqctl add_vhost myvhost

sudo rabbitmqctl set_user_tags carmack mytag

sudo rabbitmqctl set_permissions -p myvhost carmack "." "." ".*"

rabbitmqctl list_queues -p myvhost

安装celery
pip install celery -i https://pypi.douban.com/simple

broker_url = 'amqp://carmack:123456@localhost:5672/myvhost'

运行celery worker
nohup celery -A tasks worker --loglevel=info &

执行任务

```
>>> from tasks import add
>>> result = add.delay(4, 4)
>>> result.ready()
>>> result.get(timeout=1)
>>>result.status
>>>result.id
```