

# Celery

---

## 1 定义

Celery 是一个简单、灵活且可靠的，处理大量消息的分布式系统

它是一个专注于实时处理的任务队列，同时也支持任务调度

中文官网: <http://docs.jinkan.org/docs/celery/>

在线安装 `sudo pip3 install -U Celery`

离线安装

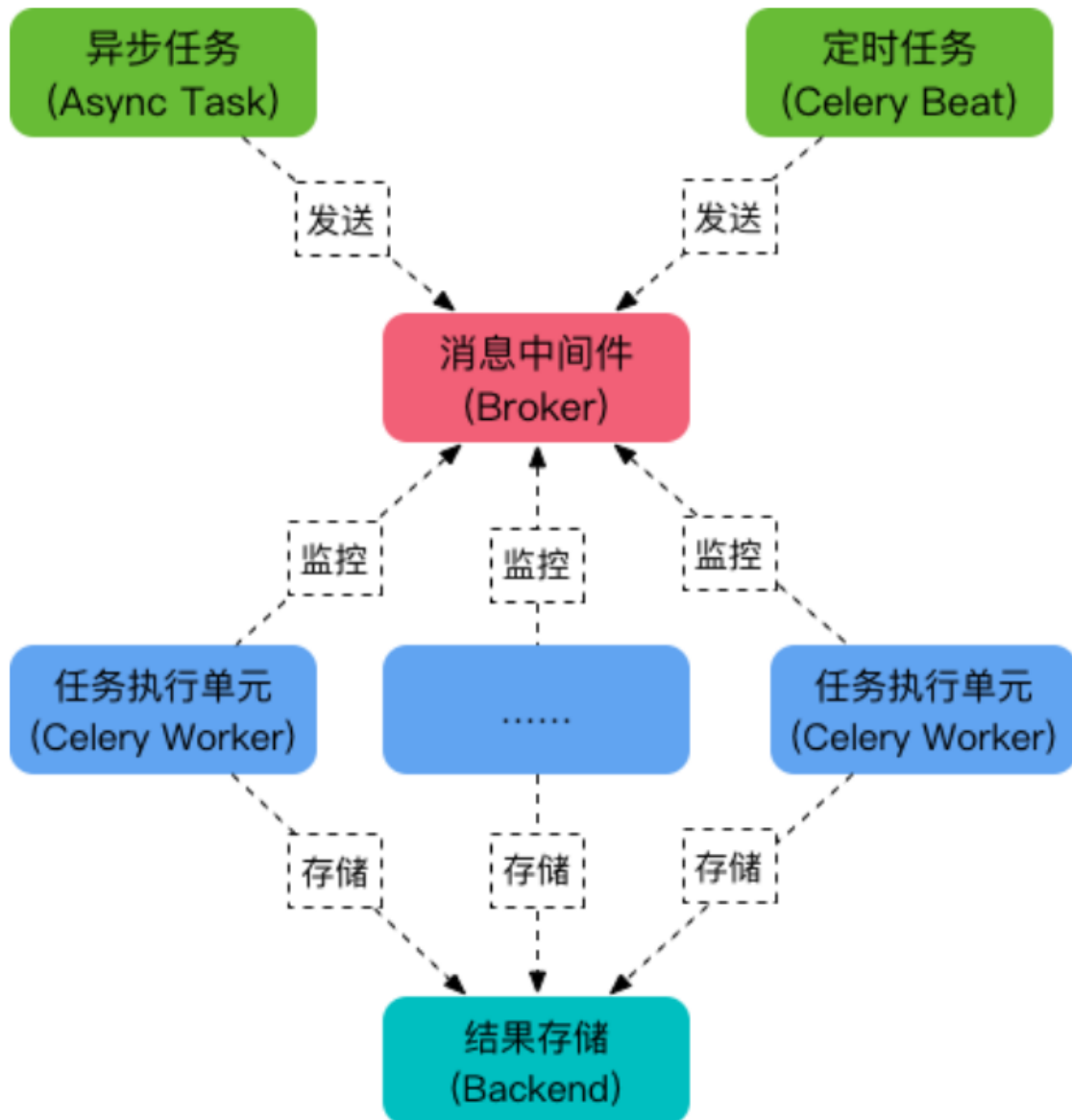
```
tar xvfz celery-0.0.0.tar.gz
cd celery-0.0.0
python3 setup.py build
python3 setup.py install
```

名词解释:

broker - 消息传输的中间件，生产者一旦有消息发送，将发至broker; 【RQ, redis】

backend - 用于存储消息/任务结果，如果需要跟踪和查询任务状态，则需添加要配置相关

worker - 工作者 - 消费/执行broker中消息/任务的进程



## 2 使用Celery

### 1, 创建worker

#创建 tasks.py 文件

```
from celery import Celery
```

#初始化celery, 指定broker

```

app = Celery('guoxiaonao',
broker='redis://:password@127.0.0.1:6379/1')

#若redis无密码, password可省略
#app = Celery('guoxiaonao',
broker='redis://:@127.0.0.1:6379/1')

# 创建任务函数
@app.task
def task_test():
    print("task is running....")

```

#Ubuntu 终端中, tasks.py文件同级目录下 执行  
celery -A tasks worker --loglevel=info  
#执行后终端显示如下, 证明成功!

```

tarena@tedu:~/PycharmProjects/test$ celery -A tasks worker --loglevel=info

----- celery@tedu v4.3.0 (rhubarb)
-----
--- * *** * -- Linux-5.0.0-32-generic-x86_64-with-Ubuntu-18.04-bionic 2019-11-05 22:4
-- * - **** ---
- ** ----- [config]
- ** ----- .> app:          guoxiaonao:0x7f4f6af314e0
- ** ----- .> transport:   redis://127.0.0.1:6379/1
- ** ----- .> results:    disabled://
- ** ----- .> concurrency: 1 (prefork)
- *** --- * --- .> task events: OFF (enable -E to monitor tasks in this worker)
-- ***** ---
--- ***** ---
----- [queues]
      .> celery          exchange=celery(key=celery)

[tasks]
  . tasks.task_test

[2019-11-05 22:46:23,785: INFO/MainProcess] Connected to redis://127.0.0.1:6379/1
[2019-11-05 22:46:23,838: INFO/MainProcess] mingle: searching for neighbors
[2019-11-05 22:46:24,942: INFO/MainProcess] mingle: all alone
[2019-11-05 22:46:25,024: INFO/MainProcess] celery@tedu ready.

```

## 2,创建生产者 - 推送任务

在tasks.py文件的同级目录进入 ipython3 执行 如下代  
码

```
from tasks import task_test
task_test.delay()
#执行后，worker终端中现如下
```

```
[2019-11-05 22:46:23,785: INFO/MainProcess] Connected to redis://127.0.0.1:6379/1
[2019-11-05 22:46:23,838: INFO/MainProcess] mingle: searching for neighbors
[2019-11-05 22:46:24,942: INFO/MainProcess] mingle: all alone
[2019-11-05 22:46:25,024: INFO/MainProcess] celery@tedu ready.
[2019-11-05 22:51:34,252: INFO/MainProcess] Received task: tasks.task_test[0c41fb77-fe71-4017-944f-b2a02e0671ab]
[2019-11-05 22:51:34,256: WARNING/ForkPoolWorker-1] task is running....
[2019-11-05 22:51:34,272: INFO/ForkPoolWorker-1] Task tasks.task_test[0c41fb77-fe71-4017-944f-b2a02e0671ab] succeeded in 0.01623172200015688s: None
```

## 存储执行结果

Celery提供存储任务执行结果的方案，需借助 redis 或 mysql 或Memcached 等

详情可见 <http://docs.celeryproject.org/en/latest/reference/celery.result.html#module-celery.result>

```
#创建 tasks_result.py
from celery import Celery
app = Celery('demo',

broker='redis://@127.0.0.1:6379/1',

backend='redis://@127.0.0.1:6379/2',

)

# 创建任务函数
@app.task
def task_test(a, b):
    print("task is running")
    return a + b
```

tasks\_result.py 同级目录终端中-启动celery worker

```
celery -A tasks_result worker --loglevel=info
```

```
tarena@tedu:~/PycharmProjects/test$ celery -A tasks_result worker --loglevel=info

----- celery@tedu v4.3.0 (rhubarb)
-----
--- * * * * * ---
--- * * * * * --- Linux-5.0.0-32-generic-x86_64-with-Ubuntu-18.04-bionic 2019-11-05 22:58:59
--- * * * * * ---
--- ** -----
--- ** ----- [config]
--- ** ----- > app: demo:0x7f57f4c933c8
--- ** ----- > transport: redis://127.0.0.1:6379/1
--- ** ----- > results: redis://127.0.0.1:6379/2
--- ** ----- > concurrency: 1 (prefork)
--- ** ----- > task events: OFF (enable -E to monitor tasks in this worker)
--- ** -----
--- ** -----
--- ** ----- [queues]
--- ** ----- .> celery exchange=celery(direct) key=celery
```

在相同目录下 打开终端创建生产者 - 同【上步】；执行成功后，可调用如下方法取得执行结果

```
from tasks_result import task_test
s = task_test.delay(10,100)
s.result
```

## 3 Django + Celery

### 1, 创建项目+应用

#常规命令

```
django-admin startproject test_celery
python manage.py startapp user
```

### 2, 创建celery.py

在settings.py同级目录下 创建 celery.py文件

文件内容如下:

```
from celery import Celery
```

```
from django.conf import settings
import os

# 为celery设置环境变量
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'chongci.settings')

# 创建应用
app = Celery("test_celery")
# 配置应用
app.conf.update(
    # 配置broker
    BROKER_URL='redis://:@127.0.0.1:6379/1',
)
# 设置app自动加载任务
app.autodiscover_tasks(settings.INSTALLED_APPS)
```

3, 在应用模块【user目录下】创建tasks.py文件

文件内容如下:

```
from chongci.celery import app
import time

@app.task
def task_test():
    print("task begin....")
    time.sleep(10)
    print("task over....")
```

4, 应用视图编写; 内容如下:

```
from django.http import HttpResponse
from .tasks import task_test
import datetime

def test_celery(request):
    task_test.delay()
    now = datetime.datetime.now()
    html = "return at %s"%
(now.strftime('%H:%M:%S'))
    return HttpResponse(html)
```

5, 分布式路由下添加 test\_celery函数对应路由, 此过程略

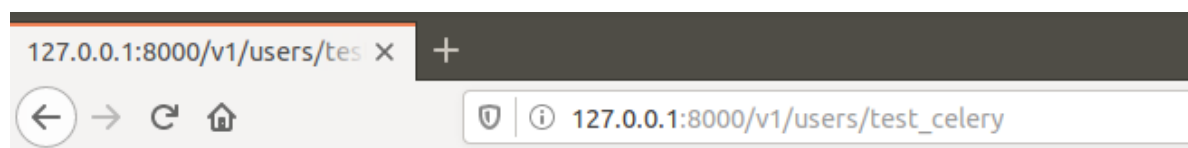
6, 启动django python3 manage.py runserver

7, 创建 celery worker

在项目路径下, 即test\_celery下 执行如下

```
celery -A test_celery worker -l info
```

8, 浏览器中执行对应url



**view return at 23:42:04**

worker终端中显示

```
[2019-11-05 23:42:04,362: INFO/MainProcess] Received task: user.tasks.task_25]
[2019-11-05 23:42:04,367: WARNING/ForkPoolWorker-1] task begin....
[2019-11-05 23:42:14,382: WARNING/ForkPoolWorker-1] task over....
[2019-11-05 23:42:14,383: INFO/ForkPoolWorker-1] Task user.tasks.task_25 succeeded in 10.016329415000655s: None
```