Celery

Distributed task queue system

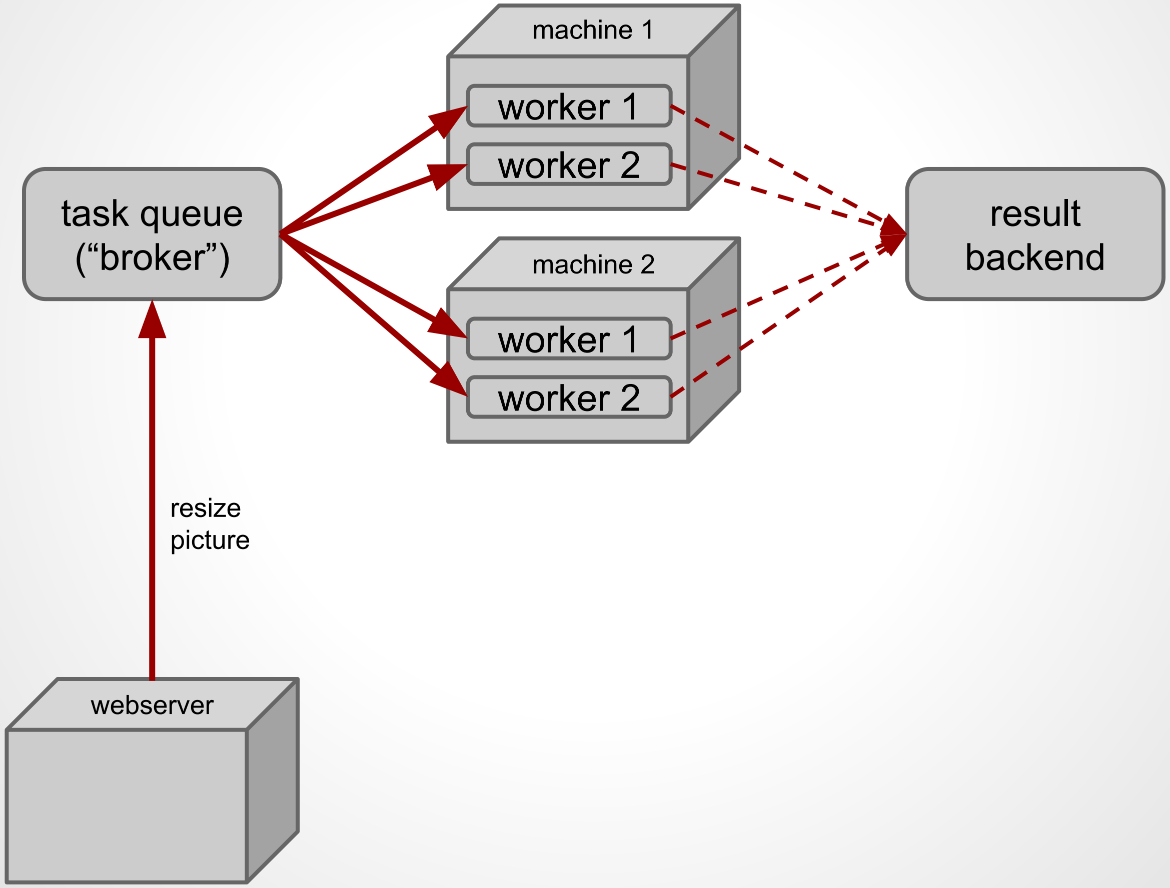
What is celery

- it’s a **distributed** task queue **system**
  - abstractions to put tasks in the task queue
  - abstractions to manage task results
  - services to run tasks from the task queue
  - monitoring tools
- can use almost any database or message queue
- works best with a task queue but can use other non-traditional transports like a sql db or redis

Possible uses

- asynchronous processing
  - emails
  - image processing
  - transcoding
  - batch import / processing
  - that slow api you need to use
- scrapping
- processing large amount of data

What is celery



Usage

```
$ pip install celery
$ sudo rabbitmqctl add_vhost test
$ sudo rabbitmqctl add_user test test
$ sudo rabbitmqctl set_permissions -p test test ".*" ".*" ".*"
$ cat demo.py
```

```
import urllib2
from celery import Celery
celery = Celery('demo',
    broker="amqp://test:test@localhost:5672/test",
    backend="amqp://test:test@localhost:5672/test",
)
@celery.task
def fetch(url):
    print 'GET: %r' % url
    return urllib2.urlopen(url).read()
```

Usage

```
$ python
Python 2.7.3 (default, Apr 10 2013, 06:20:15)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import demo
>>> result = demo.fetch.delay("http://python.org/")
>>> result
<AsyncResult: 27287957-f04f-47b4-b5f0-3fcb984dfa91>
>>> result.get()
'<!DOCTYPE html PUBLIC ......
```

Configuration

```
celery = Celery('demo')
celery.conf.update(
  SETTING=value
)
celery.config_from_object(
  'modulename'
)
```

Notable features

- events
- workflow
- rate limits
- time limits
- retry
- acks (reliable execution)
- routing
- task schedulers (like cron)
- various worker pools: process (fork), threaded, eventlet, gevent
- extensible

Brokers & backends overview

- rabbitmq
- redis
- mongodb
- sql (sqlalchemy or django)
- beanstalk

The canvas

Workflow primitives:

- **group**
  - apply tasks in parallel
- **chain**
  - apply tasks in sequence
- **chord**
  - apply task after group or task completes
- **map**, **starmap**, **chunks**
  - abstractions over **group**

Rate limits

```
@celery.task(rate_limit='1/s')
def fetch(url):
    print 'GET: %r' % url
    return urllib2.urlopen(url).read()
```

- can use “/h”, “/m”

Revokes & time limits

- **revoke**:

```
result = fetch.delay('http://python.org/')
result.revoke(terminate=True)
```

- **timelimit**:

```
@celery.task(time_limit=60)
def fetch(url):
   print 'GET: %r' % url
   return urllib2.urlopen(url).read()
```

Similar projects & alternatives

- Kuyruk
- huey
- pyres
- rq