Introduction à Celery

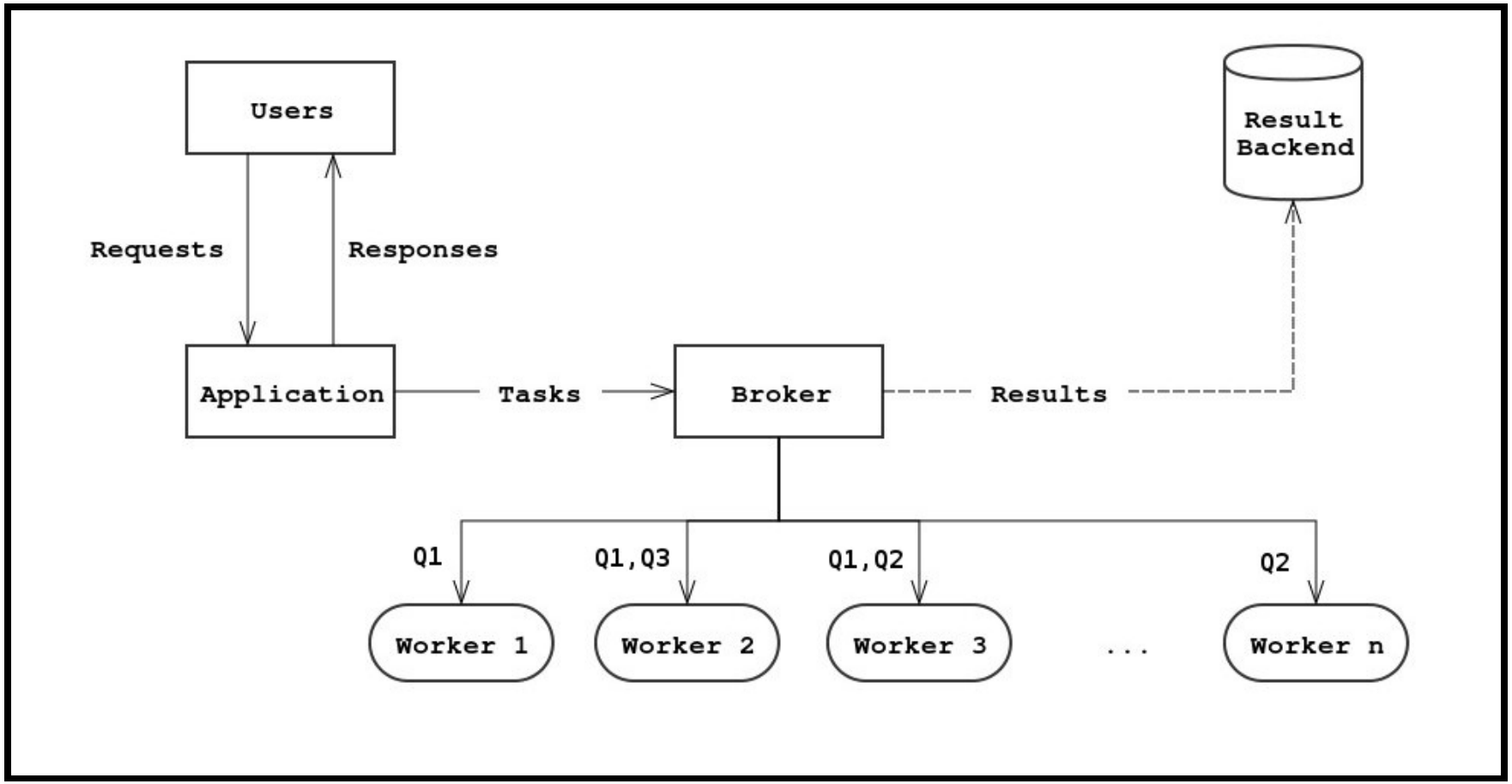
En deux mots?

«Distributed Task Queue»

En plus de deux mots?

Système **distribué** permettant d'envoyer des tâches **asynchrones** à des workers, en **temps réel** ou en différé.

En image?


Users
Result Backend
Requests
Responses
Application
Tasks
Broker
Results
Q1
Q1,Q3
Q1,Q2
Q2
Worker 1
Worker 2
Worker 3
...
Worker n

Broker

- RabbitMQ (par défaut)
- Redis
- mongoDB
- CouchDB
- DjangoDB
- SQLAlchemy
- ...

http://celery.readthedocs.org/en/latest/getting-started/brokers/index.html

Result Backend

- RabbitMQ (par défaut)
- Memcached
- Redis
- SQLAlchemy
- DjangoDB
- mongoDB
- custom

http://celery.readthedocs.org/en/latest/getting-started/first-steps-with-celery.html#keeping-results

Définition et configuration de l'application

```
# afpyro/tasks/celery.py
from __future__ import absolute_import
from celery import Celery

app = Celery('afpyro')
app.conf.update(
    BROKER_URL = 'amqp://',
    CELERY_RESULT_BACKEND = 'amqp://',
    CELERY_TASK_SERIALIZER = 'json',
    CELERY_ACCEPT_CONTENT = ['json'],
    CELERY_RESULT_SERIALIZER = 'json',
    CELERY_INCLUDE = [
        # all modules containing tasks
    ],
    CELERY_TIMEZONE = 'Europe/Paris',
    CELERY_ENABLE_UTC = True,
)
```

http://celery.readthedocs.org/en/latest/configuration.html

Définition d'une tâche

```python
# afpyro/tasks/email.py
from afpyro.tasks.celery import app

@app.task(queue='email')  # task registration
def spam_afpy_members(emails):
    ...


# afpyro/tasks/budget.py
from afpyro.tasks.celery import app

@app.task()
def generate_invoice_for_broken_plates():
    ...
```

Une tâche peut être une classe

```python
from celery import Task
from afpyro.tasks.celery import app


@app.task(queue='email')  # task registration
class SpamSender(Task):

    def __init__(self, emails):
        self.emails = emails

    def run(self):
        # send emails
```

Caractéristiques d'une "bonne" tâche?

- lente
- gourmande en mémoire
- dépend de système externes
- scalable
- indépendante
- output incertain

Exemples: envoyer des emails, générer un rapport pdf, cropper une image, etc

Une tâche peut échouer

```python
@app.task(bind=True, max_retries=3, default_retry_delay=60)
def send_twitter_status(self, oauth, tweet):
    try:
        twitter = Twitter(oauth)
        twitter.update_status(tweet)
    except (Twitter.FailWhaleError, Twitter.LoginError) as exc:
        raise self.retry(exc=exc)
```

Options utiles

```python
@app.task(max_retries=4, default_retry_delay=3*60)
def sometask(args):
    ...
```

- `max_retries` : nombre d'essai maximum (`None` → ∞)
- `default_retry_delay` : temps entre deux essais
- `ignore_result` : ne pas renvoyer les résultats de la tâche
- `throws` : liste d'exceptions "acceptables" (non synonymes d'erreur)
- ...

http://celery.readthedocs.org/en/latest/userguide/tasks.html#list-

Queues

Un worker peut dépiler une ou **plusieurs** queues.

Workers

Daemon python exécutant des tâches.

```
$ celery worker -A afpyro.tasks --loglevel=INFO -Q celery,email -n afpy
ro-w1
```

```
$ celery worker -A afpyro.tasks --loglevel=INFO -Q celery,email -n afpy
ro-w1
-------------- celery@afpyro-w1 v3.1.11 (Cipater)
---- *** * -----
---  * ** *  * -- Linux-3.11.10-7-desktop-x86_64-with-SuSE-13.1-x86_64
--  * - *** * ---
- * * ---------- [config]
- * * ---------- .> app:         afpyro:0x1135450
- * * ---------- .> transport:   mongodb://localhost:27017/celery
- * * ---------- .> results:     mongodb://localhost:27017
- ** * ---  * --- .> concurrency: 4 (prefork)
-- ****** * ----
--- **** * ----- [queues]
 -------------- .> celery          exchange=celery(direct) key=celery
                .> email           exchange=email(direct) key=email

[tasks]
  . afpyro.email.spam_afpy_members
  . afpyro.budget.generate_invoice_for_broken_plates

[2014-04-24 13:29:33,211: INFO/MainProcess] Connected to mongodb://loca
lhost:27017/celery
[2014-04-24 13:29:33,248: WARNING/MainProcess] celery@afpyro-w1 ready
[2014-04-24 13:29:33,267: INFO/MainProcess] Events of group {task} enab
led by remote.
```

Lancer une tâche

```
>>> from afpyro.tasks.email import spam_afpy_members
>>> emails = ['arthur@afpy.org', 'arthur2@afpy.org', 'arthur3@afpy.org'
]
>>> result = spam_afpy_members.apply_async(emails)
>>> result
<AsyncResult: 7eb209cb-e71c-479e-bfd3-14cfc28e92ed>
>>> result.state
'PENDING'
# time passes...
>>> result.state
'SUCCESS'
>>> result.successful()
True
>>> result.get()
['bounced', 'sent', 'sent']
```

http://celery.readthedocs.org/en/latest/userguide/calling.html

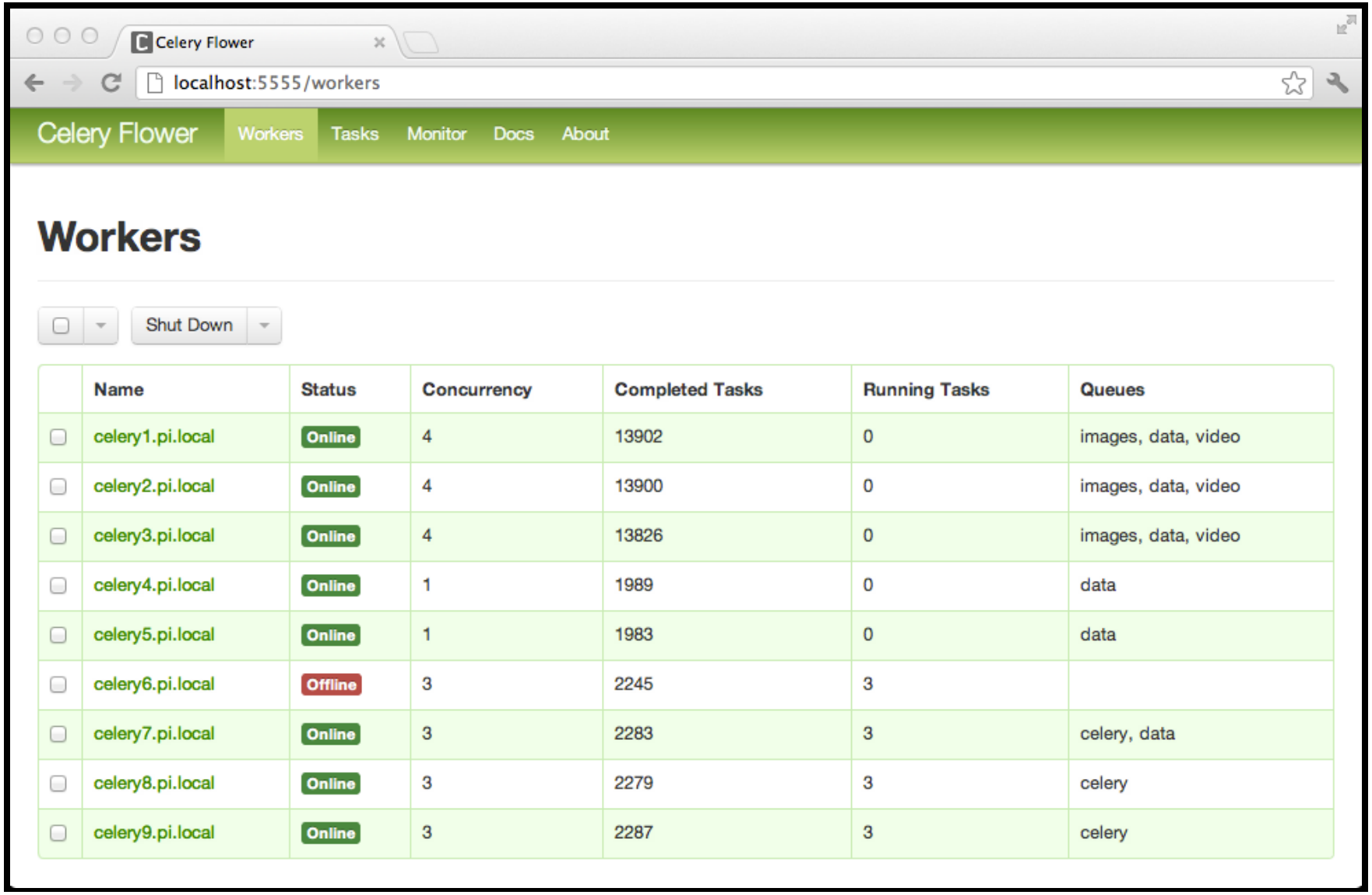
Lancer une tâche, cheatsheet

```
# always a shortcut to .apply_async.
>>> T.delay(arg, kwarg=value)
>>> T.apply_async((arg, ), {'kwarg': value})
# executes 10 seconds from now.
>>> T.apply_async(countdown=10)
# executes 10 seconds from now, specifed using eta
>>> T.apply_async(eta=now + timedelta(seconds=10))
# executes in one minute from now, but expires after 2 minutes.
>>> T.apply_async(countdown=60, expires=120)
# expires in 2 days, set using datetime.
>>> T.apply_async(expires=now + timedelta(days=2))
```

http://celery.readthedocs.org/en/latest/userguide/calling.html#basics

Flower: une interface web de gestion de workers/tâches

```
$ pip install flower
$ celery flower -A afpyro.tasks --port 5555
```


Celery Flower
localhost:5555/workers
Celery Flower
Workers
Tasks
Monitor
Docs
About
Workers
Shut Down
Name
Status
Concurrency
Completed Tasks
Running Tasks
Queues
celery1.pi.local
Online
4
13902
0
images, data, video
celery2.pi.local
Online
4
13900
0
images, data, video
celery3.pi.local
Online
4
13826
0
images, data, video
celery4.pi.local
Online
1
1989
0
data
celery5.pi.local
Online
1
1983
0
data
celery6.pi.local
Offline
3
2245
3
celery7.pi.local
Online
3
2283
3
celery, data
celery8.pi.local
Online
3
2279
3
celery
celery9.pi.local
Online
3
2287
3
celery

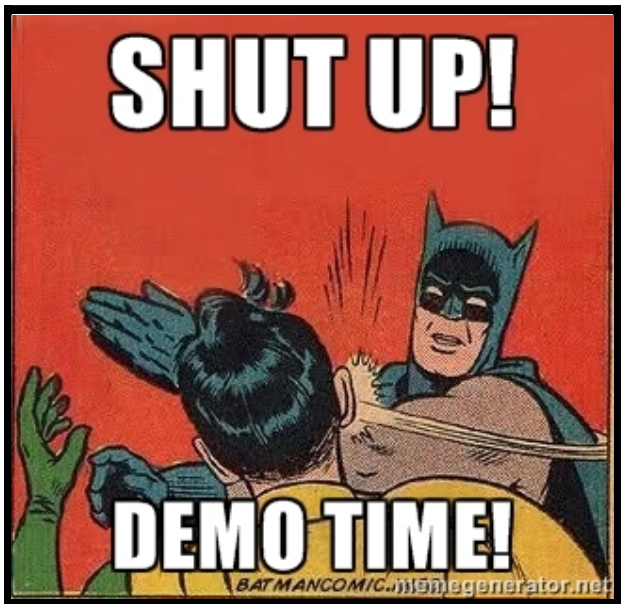
😫 → 😍: l'API REST de Flower

- lancer une tâche
- s'enquérir du status d'une tâche
- révoquer une tâche
- ajouter des workers
- éteindre un worker
- ajouter une queue à un worker
- 1000 autres trucs

Très utiles pour les infrastructures multi-langages / frameworks.

http://flower.readthedocs.org/en/latest/api.html


DEMO?
AIN'T NOBODY GOT TIME FO DAT
memegenerator.net


SHUT UP!
DEMO TIME!
BATMANCOMIC.
memegenerator.net

Autres trucs

- Sous-tâches et autres constructions géométriques (chain, starmap, chord, etc)
- Tâches programmées à une heure précise (celerybeat)
- Celery s'intègre facilement avec Flask et Django
- lisez la doc