Distributed Queue Management

CELERY

Fatih Erikli
http://fatiherikli.com
fatiherikli@gmail.com

What is Celery ?

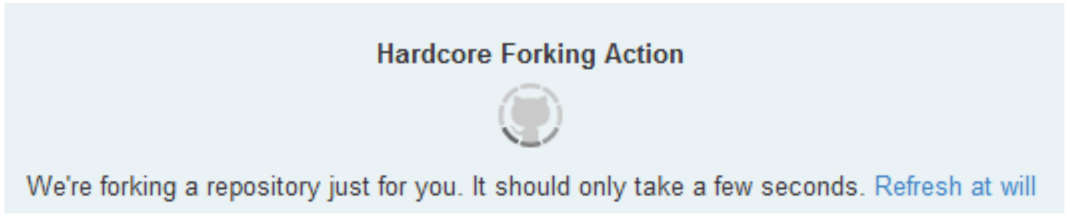
- Distributed & asynchronous message queue implementation.

- It's just wrapper, not a broker.

- Default message broker is RabbitMQ.

Why use it?

- Excluding long-process jobs from request & response cycle.
- Minimizing request & response cycle duration.
- Distributing jobs to different machines.
- Schedulable & retryable jobs.



Hardcore Forking Action

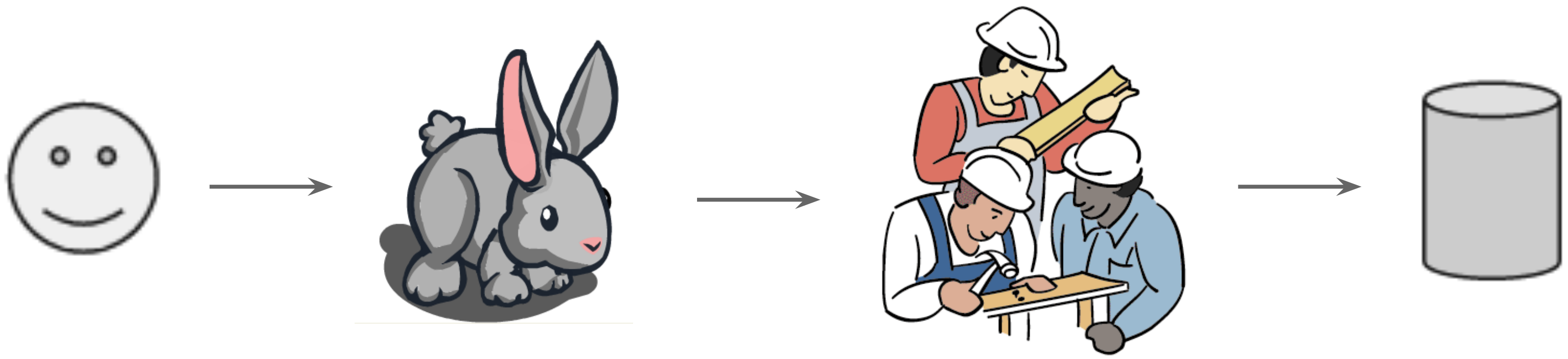
We're forking a repository just for you. It should only take a few seconds. Refresh at will

HIPO

Use cases

- Email jobs
- Long-process database operations (e.g. denormalizing)
- Communication with external API's
- Image, video processing
- Search Indexing

How it works



User makes a request. For example; a django view.

Broker redirects to related worker for this job.

If job is completed successfully or fails, result is sent to result store.

MondoDB, RabbitMQ, Redis, Django Database

Installation

RabbitMQ

```
$ apt-get install rabbitmq-server
```

(will start after the installation.)

Celery

```
$ pip install celery
$ pip install django_celery
```

Settings.py

```
INSTALLED_APPS += ('djcelery', )
```

HIPO

Configuration

```
BROKER_URL = "amqp://guest:guest@localhost:5672//"

CELERY_RESULT_BACKEND = "database"

CELERY_RESULT_DBURI = "sqlite:///mydatabase.db"

CELERYD_CONCURRENCY = 10
```

A job

app/tasks.py

```
from celery.task import task
import requests

@task(queue='check-site', name='web_site_status')
def web_site_status(url):
    """
    Down for everyone or just me !
    """
    status_code = requests.get(url=url).status_code
    return status_code
```

HIPO

Workers

```
./manage.py celeryd -Q check-site
```

```
./manage.py celeryd -Q email
```

```
./manage.py celeryd -Q image
```

```
./manage.py celeryd -Q video
```

HIPO

Calling a job

./manage.py shell

```
>>> from app.tasks import web_site_status
>>> task = web_site_status.delay('http://google.com')
# asynchronous request is started

>>> task.task_id
'7b233971-36d4-4e9a-a4e9-f8d76fd9de8e'
# wait for completing task and get result.
>>> task.get()
200
```

HIPO

djcelery.views

urls.py

```
urlpatterns = patterns('',

    url(r'^check-site$', 'djcelery.views.apply',
    {'task_name':'web_site_status'}, name='check',),

     url(r'^get-status/(.+)$', 'djcelery.views.
task_status',
     {},name='status',  ),

)
```

Periodic tasks with ...

settings.py

```
from datetime import timedelta

CELERYBEAT_SCHEDULE = {
    "runs-every-ten-minute": {
        "task": "web_site_status",
        "schedule": timedelta(minute=10),
        "args": ("http://google.com")
    },
}
```

HIPO

... celerybeat

```
./manage.py celerybeat
```

or if you using just one worker;

```
./manage.py celeryd -B
```

HIPO

Retrying

app/tasks.py

```
@task(queue='check-site', name='web_site_status')
def web_site_status(url):
    """
    Down for everyone or just me !
    """
    try:
        status_code = requests.get(url=url).status_code
    except Exception as e:
        web_site_status.retry(exc=e, countdown=60)
    return status_code
```

HIPO

Demo Application
http://pyist-celery-demo.myadslot.com:8000/

Source Code
https://github.com/fatiherikli/downforeveryoneorjustme

?

Celery
http://celeryproject.org

RabbitMQ
http://rabbitmq.com

Thanks :)