Celery - Distributed Task Queue

Mike DeLaurentis

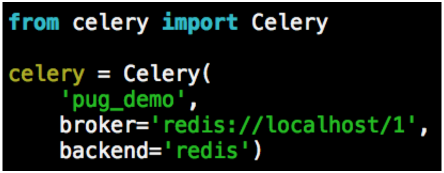
May 21, 2013

What is it?

- Distributed task queue
- Can run tasks in one process,
- ... or mulitple processes on one machine
- ... or multiple machines

Setting up

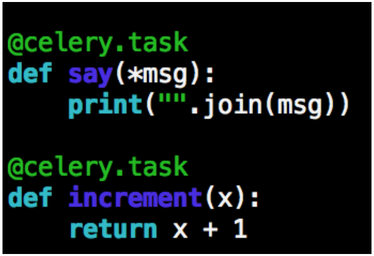
- Installs easily with pip
- Needs a "broker" (like RabbitMQ, Redis, or relational DB) to hold tasks
- May need a "backend" if you want direct access to results
- Config is easy,
- either with a simple file
- or programmatically



```
from celery import Celery

celery = Celery(
    'pug_demo',
    broker='redis://localhost/1',
    backend='redis')
```

Writing tasks

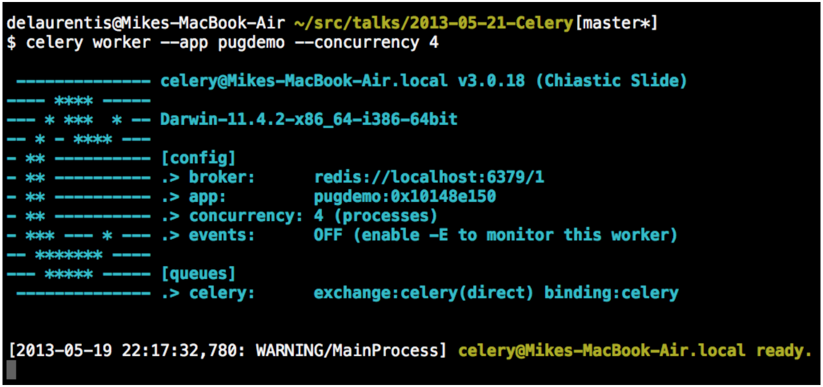
- Define tasks as decorated functions
- Takes serializable args
- Can return some result
- … or store it elsewhere



```
@celery.task
def say(*msg):
    print("".join(msg))

@celery.task
def increment(x):
    return x + 1
```

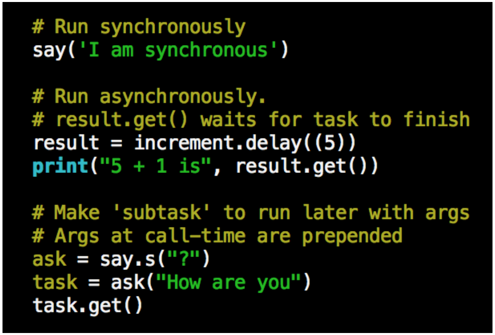
Running workers

- Start a worker by running 'celery'
- Identify module with tasks using `--app`
- Specify number of threads with `--concurrency`



```
delaurentis@Mikes-MacBook-Air ~/src/talks/2013-05-21-Celery[master*]
$ celery worker --app pugdemo --concurrency 4

 -------------- celery@Mikes-MacBook-Air.local v3.0.18 (Chiastic Slide)
---- **** -----
--- * ***  * -- Darwin-11.4.2-x86_64-i386-64bit
-- * - **** ---
- ** ---------- [config]
- ** ---------- .> broker:      redis://localhost:6379/1
- ** ---------- .> app:         pugdemo:0x10148e150
- ** ---------- .> concurrency: 4 (processes)
- *** --- * --- .> events:      OFF (enable -E to monitor this worker)
-- ******* ----
--- ***** ----- [queues]
 -------------- .> celery:      exchange:celery(direct) binding:celery


[2013-05-19 22:17:32,780: WARNING/MainProcess] celery@Mikes-MacBook-Air.local ready.
```

Submitting tasks



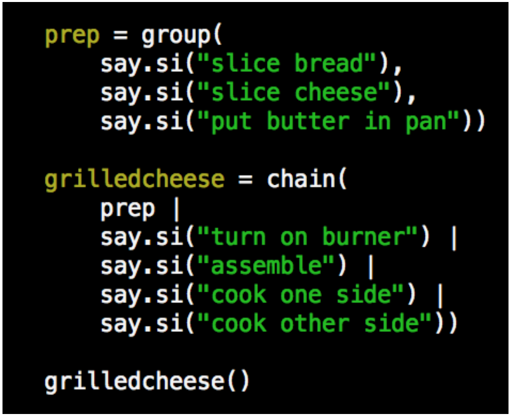
```
# Run synchronously
say('I am synchronous')

# Run asynchronously.
# result.get() waits for task to finish
result = increment.delay((5))
print("5 + 1 is", result.get())

# Make 'subtask' to run later with args
# Args at call-time are prepended
ask = say.s("?")
task = ask("How are you")
task.get()
```

- Calling directly will run in process
- Call `delay(*args)` to put on queue
- Call `s(*args)` to make task object

Creating workflows

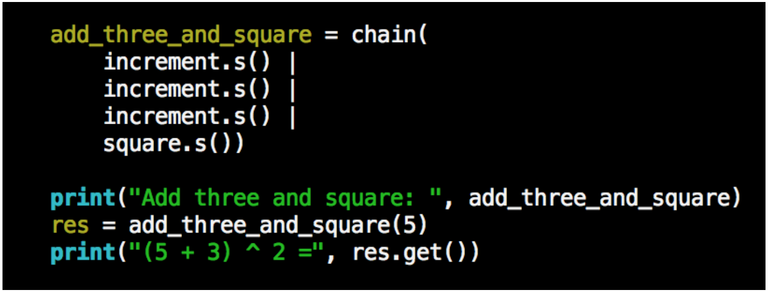
- Build a workflow programatically, out of:
  - chain - List of tasks that must be run in order
  - group - Tasks that can be run simultaneously
  - chord - Group with callback
  - chunks - Split a task with a long list of args into smaller tasks
- Compose these elements to make complex workflows

Immutable workflow



```
prep = group(
    say.si("slice bread"),
    say.si("slice cheese"),
    say.si("put butter in pan"))

grilledcheese = chain(
    prep |
    say.si("turn on burner") |
    say.si("assemble") |
    say.si("cook one side") |
    say.si("cook other side"))

grilledcheese()
```

Mutable workflow



```
add_three_and_square = chain(
    increment.s() |
    increment.s() |
    increment.s() |
    square.s())

print("Add three and square: ", add_three_and_square)
res = add_three_and_square(5)
print("(5 + 3) ^ 2 =", res.get())
```

Thanks!