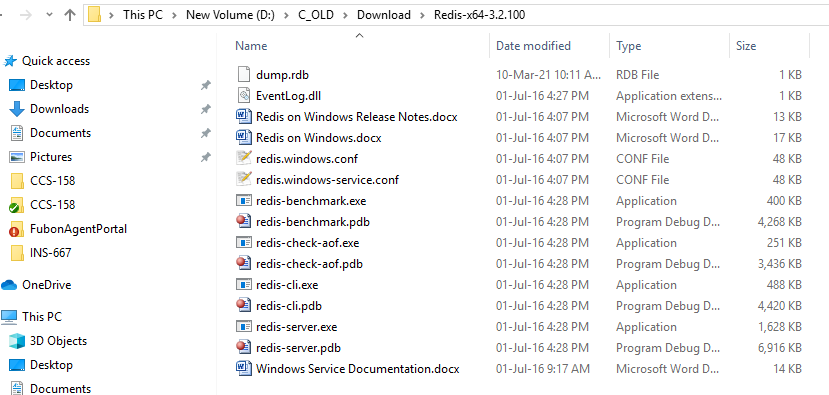
**Run celery from Customer Portal project on local**

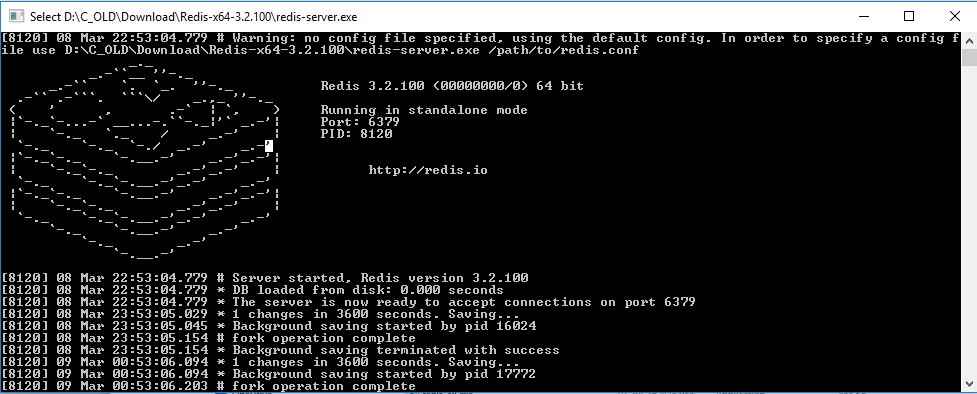
**Prepare: install django-celery-result, django-celery,… via pip**

Step 1: run redis server

Download and install redis

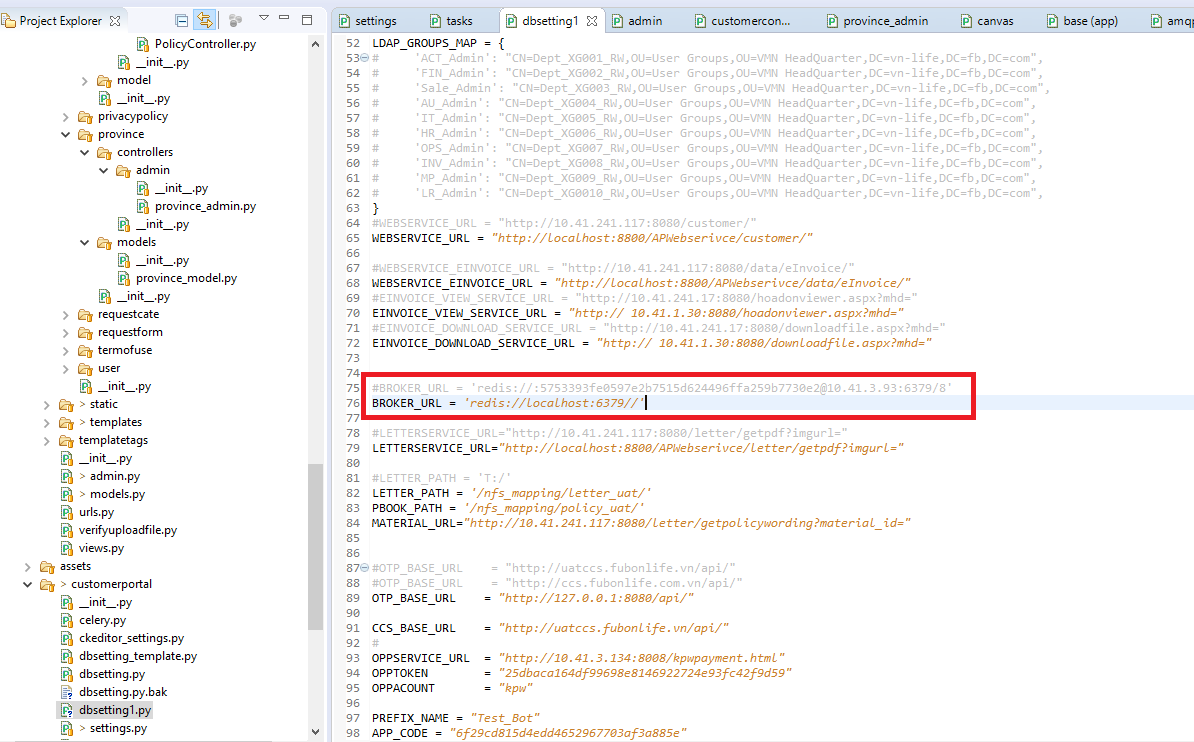


Double click to run redis-server.exe:



Step 2: setup Bocker on customerportal settings file:

BROKER\_URL = 'redis://localhost:6379//'



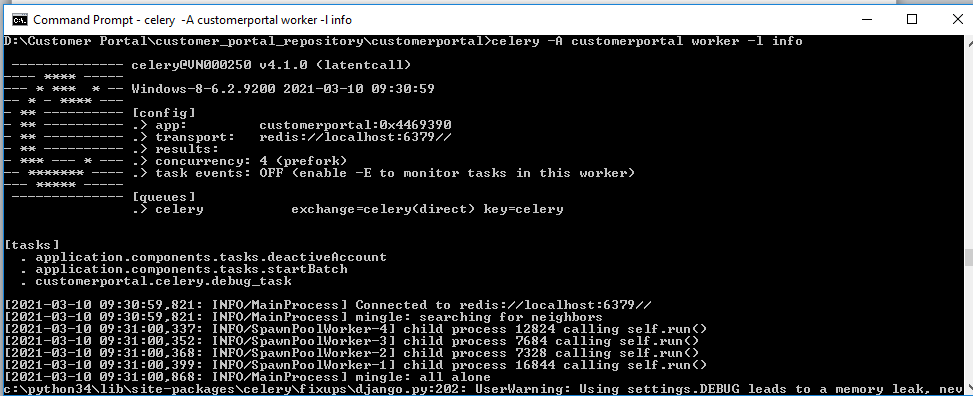
Step 3: run worker

Open another cmd terminal,

Cd to project root (where contain manage.py file),

run: celery -A customerportal worker -l info

or celery -A FubonAgentPortal worker -l info -P eventlet (----- pip install eventlet if need)



Step 4: run celery

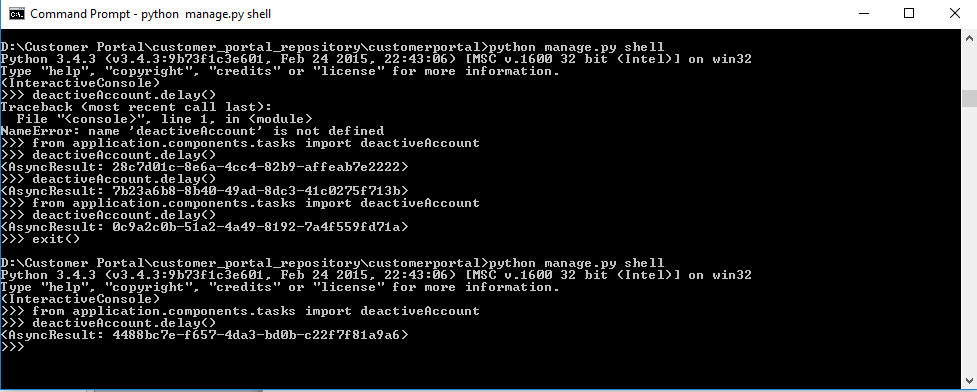
Open another cmd terminal,

Cd to project root (where contain manage.py file),

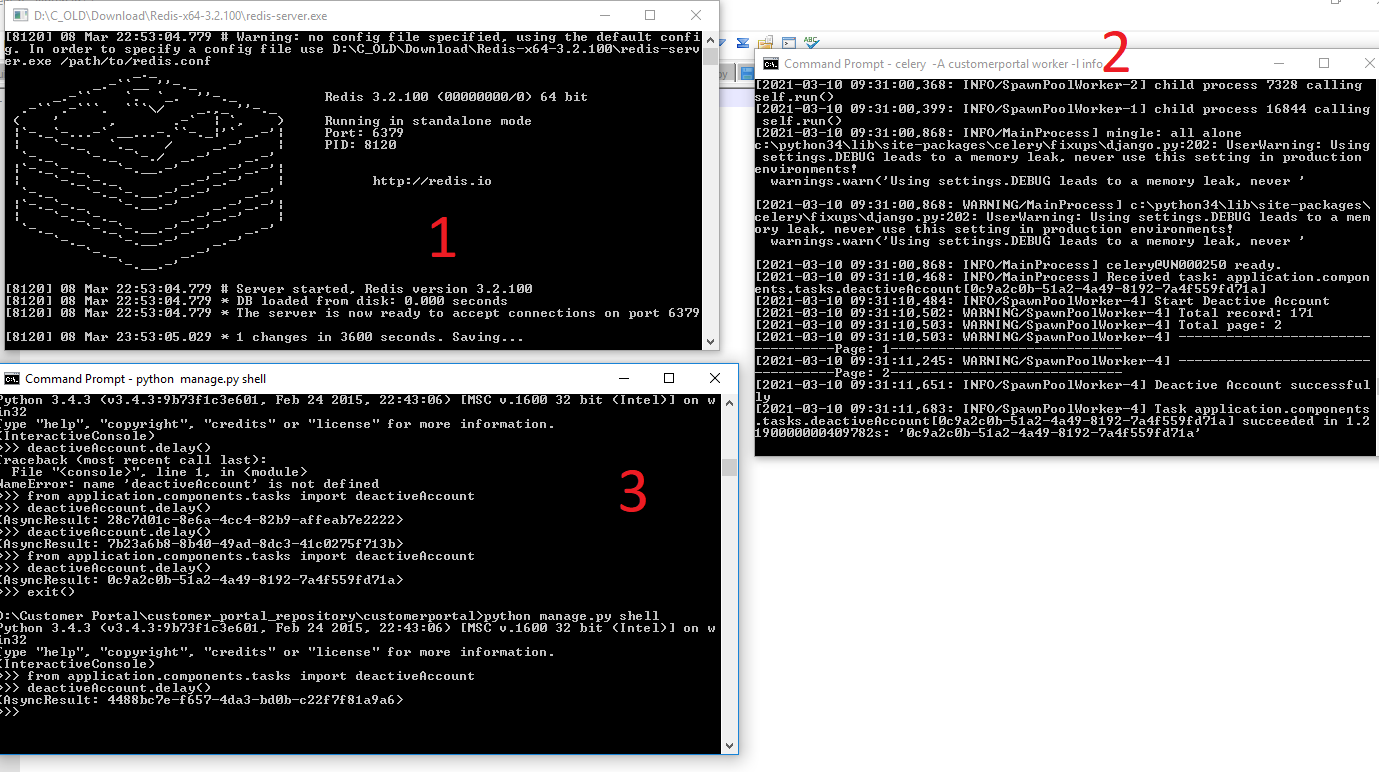
run: python manage.py shell

import task run: from application.components.tasks import deactiveAccount

run task run: deactiveAccount.delay() (deactiveAccount is task function, delay() used for run task )



Finished:



Linked document: <https://www.codingforentrepreneurs.com/blog/celery-redis-django>

Celery + Redis + Django

November 8, 2020 ·[Justin Mitchel](https://www.codingforentrepreneurs.com/u/jmitchel3)·[Beat](https://www.codingforentrepreneurs.com/t/beat)·[Celery](https://www.codingforentrepreneurs.com/t/celery)·[Django](https://www.codingforentrepreneurs.com/t/django)·[Django Scheduling](https://www.codingforentrepreneurs.com/t/django-scheduling)·[Git](https://www.codingforentrepreneurs.com/t/git)·[Heroku](https://www.codingforentrepreneurs.com/t/heroku)·[Offloading Tasks](https://www.codingforentrepreneurs.com/t/offloading-tasks)·[Production](https://www.codingforentrepreneurs.com/t/production)·[Redis](https://www.codingforentrepreneurs.com/t/redis)·[Scheduler](https://www.codingforentrepreneurs.com/t/scheduler)·[Scheduling](https://www.codingforentrepreneurs.com/t/scheduling)·[Worker](https://www.codingforentrepreneurs.com/t/worker)

*Celery* is a task queue with focus on real-time processing, while also supporting task scheduling.

*Redis* is a message broker. This means it handles the queue of "messages" between Django and Celery.

*Django* is a web framework made for perfectionists with deadlines.

All three work together to make real-time magic.

Learn Celery x Redis x Django on [Time & Tasks 2](https://www.codingforentrepreneurs.com/projects/time-tasks-2).

============

Do you have a Django Project? If not, use [this](http://kirr.co/itznm2/).

[This post](http://kirr.co/itznm2/) shows you how to create a basic Django project.

Tech Stack

* Redis
* Celery (v4+)
* Django (v2.2+)
* Python (v3.6+)

Assumptions

This project assumes you have a [Django Project](https://www.codingforentrepreneurs.com/t/django) already created. If you do not, consider doing [Try Django 2.2](https://www.codingforentrepreneurs.com/projects/try-django-2-2).

Install Redis

1. Windows - [Blog post](https://www.codingforentrepreneurs.com/blog/redis-on-windows/) - [Install Redis via Memurai (video)](https://www.codingforentrepreneurs.com/projects/setup-redis/install-redis-windows-using-memurai) - [Install via Docker (video)](https://www.codingforentrepreneurs.com/projects/setup-redis/install-redis-windows-using-memurai) 2. Mac

* [Blog post](https://www.codingforentrepreneurs.com/blog/install-redis-mac-and-linux)
* [Install Redis via Homebrew (video)](https://www.codingforentrepreneurs.com/projects/setup-redis/install-redis-macos-using-homebrew)
* [Install Redis via Docker (video)](https://www.codingforentrepreneurs.com/projects/setup-redis/install-redis-macos-using-docker)

3. Linux - [Blog post](https://www.codingforentrepreneurs.com/blog/install-redis-mac-and-linux) - Hello Linux Series [first post](https://www.codingforentrepreneurs.com/blog/hello-linux/) or [video tutorial](https://www.codingforentrepreneurs.com/projects/hello-linux)

Celery, Redis & Django

1. Create Virtual Environment

Copy

cd path/to/project/

Copy

python3 -m venv .

You can use any virtual environment manager you choose (pipenv, virtualenv, venv, poetry, etc). venv is built in to Python 3+ which is why we use it.

2. Activate Virtual Environment.

**Mac/Linux**

Copy

$ source bin/activate

(yourproject) $

**Windows**

Copy

> .\Scripts\activate

(yourproject) >

3. Install Celery & Redis Python Packages

Copy

pip install celery==4.4.7

pip install redis

pip install django-celery-beat

pip install django-celery-results

pip freeze > requirements.txt

At the time of this writing, celery==5.0.2 has been released. This version does not yet support django-celery-results so we stick with celery==4.4.7

4. Update Django settings.py:

Copy

INSTALLED\_APPS += [

'django\_celery\_beat',

'django\_celery\_results',

]

CELERY\_RESULT\_BACKEND = "django-db"

5. Run migrations:

Copy

python manage.py makemigrations

python manage.py migrate

6. Create celery.py to setup Celery app:

In this case, my django project is named cfehome.

* Navigate to root project config module (where settings and urls modules are)
* Create a celery.py file with the contents:

Copy

# yourvenv/cfehome/celery.py

from \_\_future\_\_ import absolute\_import, unicode\_literals # for python2

import os

from celery import Celery

# set the default Django settings module for the 'celery' program.

# this is also used in manage.py

os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'cfehome.settings')

## Get the base REDIS URL, default to redis' default

BASE\_REDIS\_URL = os.environ.get('REDIS\_URL', 'redis://localhost:6379')

app = Celery('cfehome')

# Using a string here means the worker don't have to serialize

# the configuration object to child processes.

# - namespace='CELERY' means all celery-related configuration keys

# should have a `CELERY\_` prefix.

app.config\_from\_object('django.conf:settings', namespace='CELERY')

# Load task modules from all registered Django app configs.

app.autodiscover\_tasks()

app.conf.broker\_url = BASE\_REDIS\_URL

# this allows you to schedule items in the Django admin.

app.conf.beat\_scheduler = 'django\_celery\_beat.schedulers.DatabaseScheduler'

7. Update project configuration folder's \_\_init\_\_.py file:

It's the \_\_init\_\_.py located in the same directory as settings.py

Copy

# ourvenv/cfehome/\_\_init\_\_.py

# This will make sure the app is always imported when

# Django starts so that shared\_task will use this app.

from .celery import app as celery\_app # noqa

8. Create tasks.py in any Django app (a valid app in INSTALLED\_APPS):

The app.autodiscover\_tasks() configuration from above will automatically find these tasks.

You may have to restart your worker (more on this below) to ensure these are registered

Copy

import random

from celery import shared\_task

@shared\_task(name="sum\_two\_numbers")

def add(x, y):

return x + y

@shared\_task(name="multiply\_two\_numbers")

def mul(x, y):

total = x \* (y \* random.randint(3, 100))

return total

@shared\_task(name="sum\_list\_numbers")

def xsum(numbers):

return sum(numbers)

9. Test tasks:

1. Open a terminal window, and run a **celery worker** with in your project root (where manage.py lives). Our project is cfehome so change your project name accordingly.

Copy

celery -A yourproject worker -l info

# like

celery -A cfehome worker -l info

2. Open another terminal window, in your Django project python manage.py shell:

Copy

>>> from yourapp.tasks import add, mul, xsum

>>> add(1,3)

4

>>> add.delay(1,3)

<AsyncResult: 7bb03f9a-5702-4661-b737-2bc54ed9f558>

delay is the key here. It offloads this task to the worker process. We discuss this in depth in the [Time & Tasks 2](https://www.codingforentrepreneurs.com/projects/time-tasks-2) series.

If you look at your celery worker, you should see something like:

Copy

[2016-11-08 22:05:22,686: INFO/PoolWorker-5] Task sum\_two\_numbers[7bb03f9a-5702-4661-b737-2bc54ed9f558] succeeded in 0.0004559689841698855s: 21

10. Setup Schedule to Run Tasks [Docs](http://docs.celeryproject.org/en/latest/userguide/periodic-tasks.html):

Copy

# yourvirtualenv/cfehome/celery.py

from celery.schedules import crontab

app.conf.beat\_schedule = {

'add-every-minute-contrab': {

'task': 'multiply\_two\_numbers',

'schedule': crontab(hour=7, minute=30, day\_of\_week=1),

'args': (16, 16),

},

'add-every-5-seconds': {

'task': 'multiply\_two\_numbers',

'schedule': 5.0,

'args': (16, 16)

},

'add-every-30-seconds': {

'task': 'tasks.add',

'schedule': 30.0,

'args': (16, 16)

},

}

11. Run Scheduled Tasks With Celery Beat:

Open another Terminal window to run scheduled tasks:

Copy

celery -A cfehome woker --beat -l info -S django

* -S django tells celery to use the Django database scheduler.

You can also have the beat server run as it's own process with celery -A cfehome beat -l info

12. 3 Processes running:

**Django**

Copy

$ python manage.py runserver

**Celery Worker & Beat** (this can be 2 processes)

Copy

# celery worker & beat

$ celery -A cfehome worker --beat -S django -l info

**redis server**

Copy

$ redis-server