**Celery & Redis**

Upload post on Instagram but make sure user experience should not get affected while it taking time.

**Why celery?**

* Takes more time to run
* We don’t want user to get stuck
* User can perform other operation while celery getting executed.

**Example:**

Email Sending

Schedule Task

Process Upload file

Send Notification

Model task for machine learning

Payment Gateway Operations

Retry Failed transactions

**[Image]**

We will temp stored message to redis, we will use .delay method using that we can get task message from there & pass it to celery

We can store task message to redis but not notifications until it get consumed.

**Redis is used to store the task message temporarily, like:**

**json**

**CopyEdit**

**{**

**"task": "myapp.tasks.send\_notification",**

**"args": ["user@example.com", "Welcome!"],**

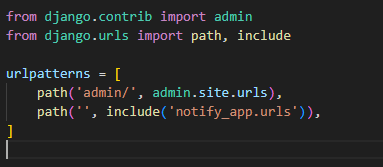
**"kwargs": {},**

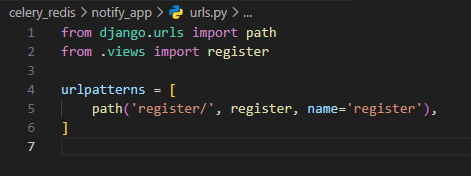
**...**

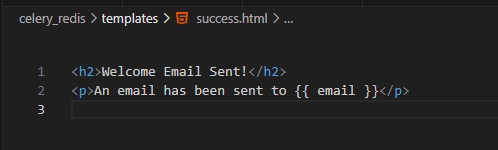
1. Waits in Redis
2. Picked up by celery worker
3. Executed runs send notifications
4. Removed after that

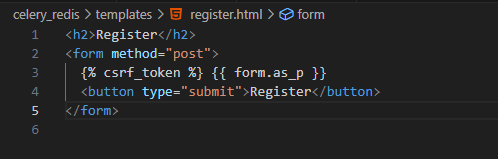
**Celery + Redis to send email notifications asynchronously.**

1. **Create project & do basic setup**

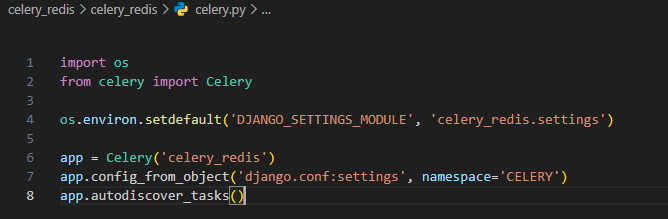
****

****

****

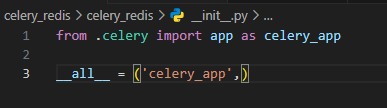
****

1. **Create celery.py**

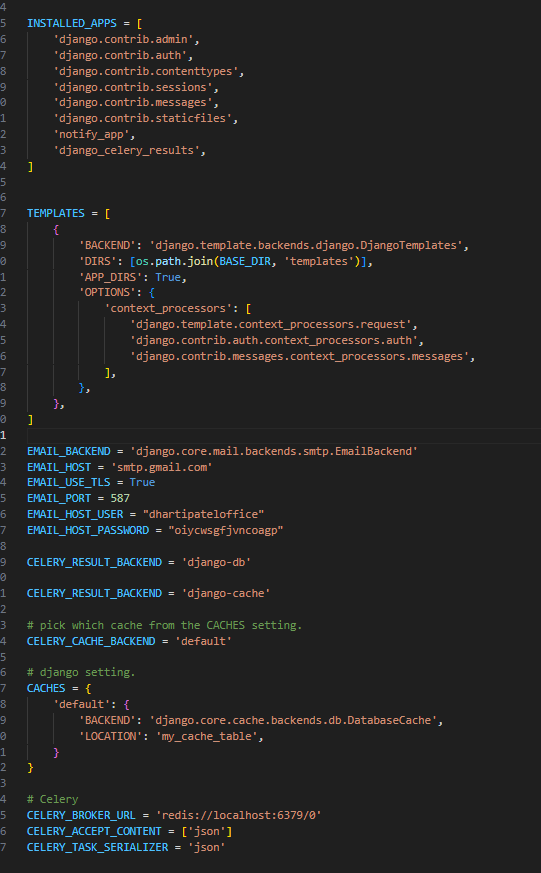
****

The uppercase name-space means that all [Celery configuration options](https://docs.celeryq.dev/en/latest/userguide/configuration.html#configuration) must be specified in uppercase instead of lowercase, and start with CELERY\_, so for example the [task\_always\_eager](https://docs.celeryq.dev/en/latest/userguide/configuration.html" \l "std-setting-task_always_eager) setting becomes CELERY\_TASK\_ALWAYS\_EAGER, and the [broker\_url](https://docs.celeryq.dev/en/latest/userguide/configuration.html" \l "std-setting-broker_url) setting becomes CELERY\_BROKER\_URL. This also applies to the workers settings, for instance, the [worker\_concurrency](https://docs.celeryq.dev/en/latest/userguide/configuration.html" \l "std-setting-worker_concurrency) setting becomes CELERY\_WORKER\_CONCURRENCY.

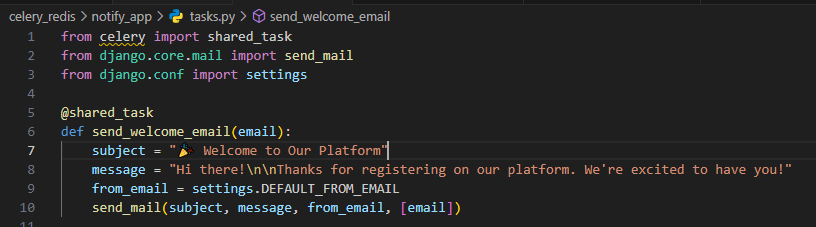
1. **Init file to run process at starting -** This ensures that the app is loaded when Django starts so that the @shared\_task decorator (mentioned later) will use it

****

1. **Add configuration to Settings File**

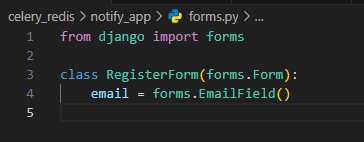
****

1. **Use @shared\_task decorator**

****

The @shared\_task decorator lets you create tasks without having any concrete app instance

1. **Trigger task**

****

Common practice for reusable apps is to define all tasks in a separate tasks.py module, and Celery does have a way to auto-discover these modules:

app.autodiscover\_tasks()

- app1/

- tasks.py

- models.py

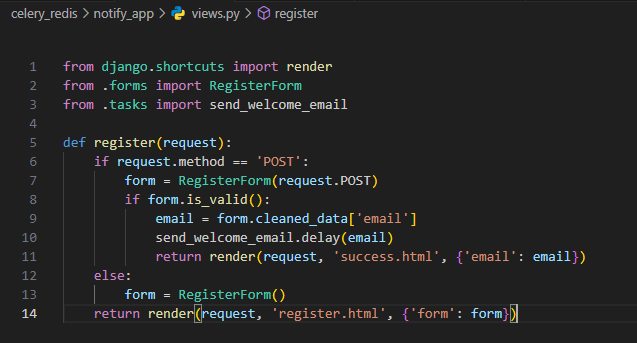
- app2/

- tasks.py

- models.py

This way you don’t have to manually add the individual modules to the [CELERY\_IMPORTS](https://docs.celeryq.dev/en/latest/userguide/configuration.html#std-setting-imports) setting.

Finally, the debug\_task example is a task that dumps its own request information. This is using the new bind=True task option introduced in Celery 3.1 to easily refer to the current task instance.

****

Celery 5.4 introduced a handy shortcut for this, using a [DjangoTask](https://docs.celeryq.dev/en/latest/reference/celery.contrib.django.task.html" \l "celery.contrib.django.task.DjangoTask" \o "celery.contrib.django.task.DjangoTask). Instead of calling [delay()](https://docs.celeryq.dev/en/latest/reference/celery.app.task.html#celery.app.task.Task.delay), you should call [delay\_on\_commit()](https://docs.celeryq.dev/en/latest/reference/celery.contrib.django.task.html#celery.contrib.django.task.DjangoTask.delay_on_commit):

This API takes care of wrapping the call into the [on\_commit](https://docs.djangoproject.com/en/stable/topics/db/transactions/" \l "django.db.transaction.on_commit) hook for you. In rare cases where you want to trigger a task without waiting, the existing [delay()](https://docs.celeryq.dev/en/latest/reference/celery.app.task.html#celery.app.task.Task.delay) API is still available.

One key difference compared to the delay method, is that delay\_on\_commit will NOT return the task ID back to the caller. The task is not sent to the broker when you call the method, only when the Django transaction finishes. If you need the task ID, best to stick to [delay()](https://docs.celeryq.dev/en/latest/reference/celery.app.task.html#celery.app.task.Task.delay).

1. **Install dependencies**

pip install Django

pip install django celery redis

1. **Start work process**

.venv/Scripts/activate

pip install django celery redis

cd celery\_redis

**Open 3 different terminal for Docker redis, django server, celery**

docker run -p 6379:6379 redis

python manage.py createcachetable

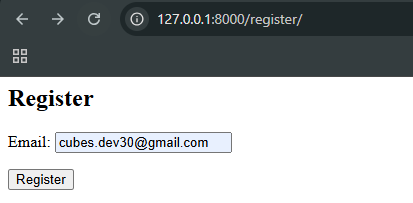
python manage.py make migrations

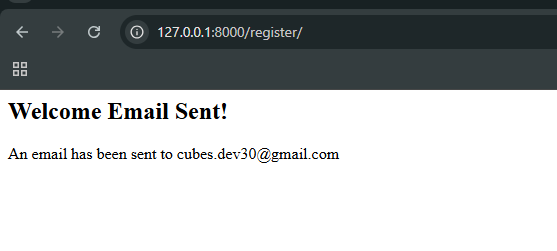
python manage.py migrate

python manage.py runserver

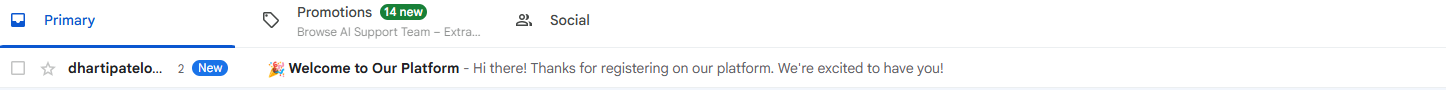
celery -A celery\_redis worker --loglevel=info --pool=solo (pool for windows)

**Final Output**

****

****

**Receive this mail via celery process which was stored in redis.**

****

Note--

CELERY\_BROKER\_URL = 'redis://localhost:6379/0' (on local)

CELERY\_BROKER\_URL = 'redis://192.168.1.50:6379/0' (on service)