Retrieve list of tasks in a queue in Celery

Asked 12 years, 6 months ago Modified 11 days ago Viewed 248k times

215

How can I retrieve a list of tasks in a queue that are yet to be processed?

python celery

Share Improve this question Follow

asked Apr 4, 2011 at 21:35

bradley.ayers
37.2k • 14 • 94 • 100

3 RabbitMQ, but I want to retrieve this list inside Python. – bradley.ayers Apr 5, 2011 at 20:18

19 Answers

Sorted by: Highest score (default)

241

EDIT: See other answers for getting a list of tasks in the queue.

You should look here: Celery Guide - Inspecting Workers

Basically this:

```
my_app = Celery(...)

# Inspect all nodes.
i = my_app.control.inspect()

# Show the items that have an ETA or are scheduled for later processing
i.scheduled()

# Show tasks that are currently active.
i.active()

# Show tasks that have been claimed by workers
i.reserved()
```

Depending on what you want

Share Improve this answer Follow

edited Apr 6, 2021 at 16:34

Mark Mishyn
3,971 • 2 • 28 • 30

answered Feb 20, 2012 at 22:35

semarj
2,467 • 1 • 14 • 4

15 I tried that, but it's realy slow (like 1 sec). I'm using it syncrhonously in a tornado app to monitor progress, so it has to be fast. – julienfr112 Apr 21, 2013 at 17:08

67 This will not return a list of tasks in the queue that have yet to be processed. – Ed J May 23, 2014 at 19:48

12 Use `i.reserved()` to get a list of queued tasks. – dwitvliet Jun 13, 2014 at 20:25

11 When specifying the worker I had to use a list as argument: `inspect(['celery@Flatty'])`. Huge speed improvement over `inspect()`. – Adversus Dec 14, 2015 at 12:46

12 This does not answer the question. I have no idea why is this answer accepted... :) – DejanLekic Mar 21, 2017 at 23:55

64

If you are using **Celery+Django** simplest way to inspect tasks using commands directly from your terminal in your *virtual environment* or using a *full path* to celery:

Doc: http://docs.celeryproject.org/en/latest/userguide/workers.html?highlight=revoke#inspecting-workers

```
$ celery inspect reserved
$ celery inspect active
$ celery inspect registered
$ celery inspect scheduled
```

Also if you are using **Celery+RabbitMQ** you can **inspect the list of queues** using the following command:

More info: https://linux.die.net/man/1/rabbitmqctl

```
$ sudo rabbitmqctl list_queues
```

Share Improve this answer Follow

edited Feb 28, 2020 at 12:13

Rune Kaagaard
6,643 ● 2 ● 38 ● 28

answered Apr 11, 2019 at 11:48

Alexandr S.
1,605 ● 1 ● 14 ● 16

16 If you have a define project, you can use `celery -A my_proj inspect reserved` – sashaboulouds Aug 27, 2019 at 19:09

10 This, again, does not answer the question. – DejanLekic May 20, 2021 at 10:09

I'm here because my Celery server is currently overloaded with tasks. This approach doesn't help, because it just hangs. So do things like app.control.inspect().active(), from another answer. I just want to kill some jobs so my server's functional again... – Apollo Grace yesterday

55

if you are using rabbitMQ, use this in terminal:

```
sudo rabbitmqctl list_queues
```

it will print list of queues with number of pending tasks. for example:

```
Listing queues ...
0b27d8c59fba4974893ec22d478a7093    0
0e0a2da9828a48bc86fe993b210d984f    0
10@torob2.celery.pidbox 0
11926b79e30a4f0a9d95df61b6f402f7    0
15c036ad25884b82839495fb29bd6395    1
celerey_mail_worker@torob2.celery.pidbox    0
celery  166
```

```
celeryev.795ec5bb-a919-46a8-80c6-5d91d2fcf2aa   0
celeryev.faa4da32-a225-4f6c-be3b-d8814856d1b6   0
```

the number in right column is number of tasks in the queue. in above, celery queue has 166 pending task.

Share Improve this answer Follow

answered Apr 9, 2015 at 11:55

Ali
6,368 ● 3 ● 38 ● 48

2 I am familiar with this when I have sudo privileges, but I want an unprivileged, system user to be able to check - any suggestions? – sage Aug 12, 2016 at 5:31

In addition you can pipe this through `grep -e "^celery\s" | cut -f2` to extract that `166` if you want to process that number later, say for stats. – jamesc Nov 1, 2017 at 11:17

38

If you don't use prioritized tasks, this is actually pretty simple if you're using Redis. To get the task counts:

```
redis-cli -h HOST -p PORT -n DATABASE_NUMBER llen QUEUE_NAME
```

But, prioritized tasks use a different key in redis, so the full picture is slightly more complicated. The full picture is that you need to query redis for every priority of task. In python (and from the Flower project), this looks like:

```
PRIORITY_SEP = '\x06\x16'
DEFAULT_PRIORITY_STEPS = [0, 3, 6, 9]


def make_queue_name_for_pri(queue, pri):
    """Make a queue name for redis

    Celery uses PRIORITY_SEP to separate different priorities of tasks into
    different queues in Redis. Each queue-priority combination becomes a key in
    redis with names like:

     - batch1\x06\x163 <-- P3 queue named batch1

    There's more information about this in Github, but it doesn't look like it
    will change any time soon:

      - https://github.com/celery/kombu/issues/422

    In that ticket the code below, from the Flower project, is referenced:

      - https://github.com/mher/flower/blob/master/flower/utils/broker.py#L135

    :param queue: The name of the queue to make a name for.
    :param pri: The priority to make a name with.
    :return: A name for the queue-priority pair.
    """
    if pri not in DEFAULT_PRIORITY_STEPS:
        raise ValueError('Priority not in priority steps')
    return '{0}{1}{2}'.format(*((queue, PRIORITY_SEP, pri) if pri else
                                (queue, '', '')))


def get_queue_length(queue_name='celery'):
    """Get the number of tasks in a celery queue.

    :param queue_name: The name of the queue you want to inspect.
    :return: the number of items in the queue.
    """
    priority_names = [make_queue_name_for_pri(queue_name, pri) for pri in
                      DEFAULT_PRIORITY_STEPS]
```

```
    r = redis.StrictRedis(
        host=settings.REDIS_HOST,
        port=settings.REDIS_PORT,
        db=settings.REDIS_DATABASES['CELERY'],
    )
    return sum([r.llen(x) for x in priority_names])
```

If you want to get an actual task, you can use something like:

```
redis-cli -h HOST -p PORT -n DATABASE_NUMBER lrange QUEUE_NAME 0 -1
```

From there you'll have to deserialize the returned list. In my case I was able to accomplish this with something like:

```
r = redis.StrictRedis(
    host=settings.REDIS_HOST,
    port=settings.REDIS_PORT,
    db=settings.REDIS_DATABASES['CELERY'],
)
l = r.lrange('celery', 0, -1)
pickle.loads(base64.decodestring(json.loads(l[0])['body']))
```

Just be warned that deserialization can take a moment, and you'll need to adjust the commands above to work with various priorities.

Share Improve this answer Follow

edited May 11, 2017 at 18:14

answered Apr 15, 2017 at 0:02

mlissner
17.4k ● 18 ● 107 ● 169

After using this in production, I've learned that it fails if you use prioritized tasks, due to the design of Celery. – mlissner May 11, 2017 at 5:05

2 I've updated the above to handle prioritized tasks. Progress! – mlissner May 11, 2017 at 18:15

10 Just to spell things out, the `DATABASE_NUMBER` used by default is `0`, and the `QUEUE_NAME` is `celery`, so `redis-cli -n 0 llen celery` will return the number of queued messages. – Vineet Bansal Nov 6, 2019 at 18:42

2 It always return 0. – Mark Mishyn Apr 6, 2021 at 9:22

1 The problem I experience with this solution : if you revoke a celery task that is waiting in the queue, it stays in the redis queue. And number of tasks returned by lrange is not correct. – Rom1 Oct 15, 2021 at 14:19

16

To retrieve tasks from backend, use this

```
from amqplib import client_0_8 as amqp
conn = amqp.Connection(host="localhost:5672 ", userid="guest",
                       password="guest", virtual_host="/", insist=False)
chan = conn.channel()
name, jobs, consumers = chan.queue_declare(queue="queue_name", passive=True)
```

Share Improve this answer Follow

edited Apr 14, 2017 at 21:18

mlissner
17.4k ● 18 ● 107 ● 169

answered Oct 19, 2013 at 11:43

ashish
2,100 ● 3 ● 17 ● 16

4 but 'jobs' gives only number of tasks in queue – bitnik Mar 30, 2017 at 11:30

3 See stackoverflow.com/a/57807913/9843399 for related answer that gives you the names of the tasks. – Caleb Syring Sep 5, 2019 at 14:42

A copy-paste solution for Redis with json serialization:

```
def get_celery_queue_items(queue_name):
    import base64
    import json

    # Get a configured instance of a celery app:
    from yourproject.celery import app as celery_app

    with celery_app.pool.acquire(block=True) as conn:
        tasks = conn.default_channel.client.lrange(queue_name, 0, -1)
        decoded_tasks = []

    for task in tasks:
        j = json.loads(task)
        body = json.loads(base64.b64decode(j['body']))
        decoded_tasks.append(body)

    return decoded_tasks
```

It works with Django. Just don't forget to change `yourproject.celery`.

Share Improve this answer Follow

edited Jun 8, 2018 at 14:22

answered May 4, 2018 at 22:01

Max Malysh

29.5k • 19 • 111 • 117

5 If you're using the pickle serializer, then you can change the `body =` line to `body = pickle.loads(base64.b64decode(j['body']))`. – Jim Hunziker Jul 6, 2018 at 18:08

i have this error ! module 'celery.app' has no attribute 'pool' – Sadegh-khan Jul 22 at 12:20

This worked for me in my application:

```
def get_queued_jobs(queue_name):
    connection = <CELERY_APP_INSTANCE>.connection()

    try:
        channel = connection.channel()
        name, jobs, consumers = channel.queue_declare(queue=queue_name,
passive=True)
        active_jobs = []

        def dump_message(message):
            active_jobs.append(message.properties['application_headers']['task'])

        channel.basic_consume(queue=queue_name, callback=dump_message)

        for job in range(jobs):
            connection.drain_events()

        return active_jobs
    finally:
        connection.close()
```

`active_jobs` will be a list of strings that correspond to tasks in the queue.

Don't forget to swap out CELERY_APP_INSTANCE with your own.

Thanks to @ashish for pointing me in the right direction with his answer here: https://stackoverflow.com/a/19465670/9843399

Share Improve this answer Follow

edited Apr 17 at 16:23

answered Sep 5, 2019 at 14:41

Caleb Syring

1,107 ● 1 ● 11 ● 20

in my case `jobs` is always zero... any idea? – daveoncode Apr 29, 2020 at 10:29

@daveoncode I don't think that's enough information for me to respond helpfully. You could open your own question. I don't think it would be a duplicate of this one if you specify that you want to retrieve the information in python. I'd go back to stackoverflow.com/a/19465670/9843399, which is what I based my answer off of, and make sure that works first. – Caleb Syring Apr 30, 2020 at 14:21

2 @CalebSyring This is the first approach that really shows me the queued tasks. Very nice. The only problem for me is that the list append does not seem to work. Any ideas how i can make the callback function write to the list? – Varlor Jul 7, 2020 at 14:26

@Varlor I'm sorry, someone made an improper edit to my answer. You can look in the edit history for the original answer, which will most likely work for you. I'm working on getting this fixed. (EDIT: I just went in and rejected the edit, which had an obvious python error. Let me know if this fixed your problem or not.) – Caleb Syring Jul 7, 2020 at 14:51

@CalebSyring I now used your code in a class, having the list as a class attribute works! – Varlor Jul 8, 2020 at 11:44

7

The celery inspect module appears to only be aware of the tasks from the workers perspective. If you want to view the messages that are in the queue (yet to be pulled by the workers) I suggest to use pyrabbit, which can interface with the rabbitmq http api to retrieve all kinds of information from the queue.

An example can be found here: Retrieve queue length with Celery (RabbitMQ, Django)

Share Improve this answer Follow

edited May 23, 2017 at 10:31

Community Bot

1 ● 1

answered Aug 30, 2016 at 14:26

Paul in 't Hout

421 ● 5 ● 8

5

I think the only way to get the tasks that are waiting is to keep a list of tasks you started and let the task remove itself from the list when it's started.

With rabbitmqctl and list_queues you can get an overview of how many tasks are waiting, but not the tasks itself:
http://www.rabbitmq.com/man/rabbitmqctl.1.man.html

If what you want includes the task being processed, but are not finished yet, you can keep a list of you tasks and check their states:

```
from tasks import add
result = add.delay(4, 4)

result.ready() # True if finished
```

Or you let Celery store the results with CELERY_RESULT_BACKEND and check which of your tasks are not in there.

Share Improve this answer Follow

answered Apr 13, 2011 at 10:36

Sebastian Blask

2,890 ● 1 ● 17 ● 29

5

As far as I know Celery does not give API for examining tasks that are waiting in the queue. This is broker-specific. If you use Redis as a broker for an example, then examining tasks that are waiting in the `celery` (default) queue is as simple as:

1. connect to the broker
2. list items in the `celery` list (LRANGE command for an example)

Keep in mind that these are tasks WAITING to be picked by available workers. Your cluster may have some tasks running - those will not be in this list as they have already been picked.

The process of retrieving tasks in particular queue is broker-specific.

Share Improve this answer Follow

edited May 20, 2021 at 10:08

answered May 4, 2018 at 8:48

DejanLekic
18.8k ● 4 ● 46 ● 77

2

I've come to the conclusion the best way to get the number of jobs on a queue is to use `rabbitmqctl` as has been suggested several times here. To allow any chosen user to run the command with `sudo` I followed the instructions here (I did skip editing the profile part as I don't mind typing in sudo before the command.)

I also grabbed jamesc's `grep` and `cut` snippet and wrapped it up in subprocess calls.

```
from subprocess import Popen, PIPE
p1 = Popen(["sudo", "rabbitmqctl", "list_queues", "-p", "[name of your virtula
host"], stdout=PIPE)
p2 = Popen(["grep", "-e", "^celery\s"], stdin=p1.stdout, stdout=PIPE)
p3 = Popen(["cut", "-f2"], stdin=p2.stdout, stdout=PIPE)
p1.stdout.close()
p2.stdout.close()
print("number of jobs on queue: %i" % int(p3.communicate()[0]))
```

Share Improve this answer Follow

edited Nov 16, 2017 at 6:11

answered Nov 16, 2017 at 4:23

Peter Shannon
191 ● 1 ● 3

2

If you control the code of the tasks then you can work around the problem by letting a task trigger a trivial retry the first time it executes, then checking `inspect().reserved()`. The retry registers the task with the result backend, and celery can see that. The task must accept `self` or `context` as first parameter so we can access the retry count.

```
@task(bind=True)
def mytask(self):
    if self.request.retries == 0:
        raise self.retry(exc=MyTrivialError(), countdown=1)
    ...
```

This solution is broker agnostic, ie. you don't have to worry about whether you are using RabbitMQ or Redis to store the tasks.

EDIT: after testing I've found this to be only a partial solution. The size of reserved is limited to the prefetch setting for the worker.

Share Improve this answer Follow

edited Oct 14, 2018 at 20:37

answered Oct 14, 2018 at 19:21

hedleyroos
341 ● 3 ● 9

1

```
from celery.task.control import inspect
def key_in_list(k, l):
    return bool([True for i in l if k in i.values()])

def check_task(task_id):
    task_value_dict = inspect().active().values()
```

```
    for task_list in task_value_dict:
        if self.key_in_list(task_id, task_list):
            return True
    return False
```

Share Improve this answer Follow

edited Jul 21, 2019 at 6:45

answered Jul 21, 2019 at 6:31

张朝龙
19 ● 2

3 For Celery > 5, you can try: `from your_app.celery import app` and then for example: `app.control.inspect().active()` – epineda Oct 2, 2021 at 18:03

1

```
inspector = current_celery_app.control.inspect()
scheduled = list(inspector.scheduled().values())[0]
active = list(inspector.active().values())[0]
reserved = list(inspector.reserved().values())[0]
registered = list(inspector.registered().values())[0]
lst = [*scheduled, *active, *reserved]
for i in lst:
    if job_id == i['id']:
        print("Job found")
```

Share Improve this answer Follow

answered Jun 29 at 13:47

Shirantha Madusanka
1,461 ● 1 ● 11 ● 16

0

With `subprocess.run`:

```
import subprocess
import re
active_process_txt = subprocess.run(['celery', '-A', 'my_proj', 'inspect',
'active'],

stdout=subprocess.PIPE).stdout.decode('utf-8')
return len(re.findall(r'worker_pid', active_process_txt))
```

Be careful to change `my_proj` with `your_proj`

Share Improve this answer Follow

answered Aug 27, 2019 at 19:38

sashaboulouds
1,586 ● 11 ● 16

1 This is not an answer to the question. This gives list of active tasks (tasks that are currently running). The question is about how to list tasks that are waiting in the queue. – DejanLekic May 20, 2021 at 10:05

0

To get the number of tasks on a queue you can use the flower library, here is a simplified example:

```
import asyncio
from flower.utils.broker import Broker
from django.conf import settings

def get_queue_length(queue):
    broker = Broker(settings.CELERY_BROKER_URL)
    queues_result = broker.queues([queue])
```

```
    res = asyncio.run(queues_result) or [{ "messages": 0 }]
    length = res[0].get('messages', 0)
```

Share Improve this answer Follow

edited Jul 6 at 14:10

answered Aug 18, 2022 at 13:33

Rafen
138 ● 1 ● 3

Here it works for me without remove messages in queue

```
def get_broker_tasks() -> []:
    conn = <CELERY_APP_INSTANCE>.connection()

    try:
        simple_queue = conn.SimpleQueue(queue_name)
        queue_size = simple_queue.qsize()
        messages = []

        for i in range(queue_size):
            message = simple_queue.get(block=False)
            messages.append(message)

        return messages
    except:
        messages = []
        return messages
    finally:
        print("Close connection")
        conn.close()
```

Don't forget to swap out CELERY_APP_INSTANCE with your own.

@Owen: Hope my solution meet your expectations.

Share Improve this answer Follow

answered Aug 8 at 3:51

HungNV
11 ● 3

```
def get_queue_length(total_tasks: int, queue_name: str, node_name: str):
    queue_size = 0
    inspector = app.control.inspect()
    stats = inspector.stats()
    if stats is not None:
        if f"celery@{node_name}" in stats.keys():
            total = stats[f"celery@{node_name}"]["total"]
            if queue_name in total.keys():
                active_tasks = total[queue_name]
                if int(total_tasks) > int(active_tasks):
                    queue_size = total_tasks - active_tasks
    return queue_size
```

This leverages `celery`'s `control` and `inspect` commands but also keeps an eye on the tasks that have been submitted.

This alone doesn't really work unless you have some sort of loop that is enqueueing items, like the following:

```
total_tasks = 0
max_queue_length = 100 # choose your number
queue = "celery_queue"
full_queue_name = "YourCeleryApp.your_celery_queue_name"
```

```
for item in list_of_tasks
    total_tasks+=1
    queue_length = get_queue_length(total_tasks=total_tasks,
queue_name=full_queue_name  node_name=node_name)
    while int(queue_length) >= max_queue_length:
        time.sleep(10)
        queue_length = get_queue_length(total_tasks=total_tasks,
queue_name=full_queue_name , node_name=node_name)
    your_celery_task.apply_async(kwargs={},queue=queue)
```

With this what's happening is the following:

1. Keep track of how many items have been submitted
2. The above code will get the `total` which is the number of tasks that have been processed by a specific worker in a particular queue.
3. We check whether the number of total tasks submitted is greater than our `active_tasks` or the tasks that have been processed by celery.

What this means is that if there are `50` tasks submitted and `30` have been processed, then there are `50-30 = 20` tasks in the queue

Share Improve this answer Follow

answered Sep 6 at 4:53

Kevin Cohen
1,241 ● 2 ● 15 ● 22

0

I found a usecase from the Flower codebase to get the broker queue length. It's fast as broker access.

```
app = Celery("tasks")

from flower.utils.broker import Broker

broker = Broker(
    app.connection(connect_timeout=1.0).as_uri(include_password=True),
    broker_options=app.conf.broker_transport_options,
    broker_use_ssl=app.conf.broker_use_ssl,
)

async def queue_length():
    queues = await broker.queues(["celery"])
    return queues[0].get("messages")
```

Share Improve this answer Follow

answered Sep 25 at 21:29

Valentino
1