

Celery: how to get queue size in a reliable and testable way

Asked 3 years, 4 months ago Modified today Viewed 8k times



I'm losing my mind trying to find a reliable and testable way to get the number of tasks contained in a given Celery queue.

7



I've already read these two related discussions:

- [Django Celery.get task count](#)

Note: I'm not using Django nor any other Python web framework.

- [Retrieve list of tasks in a queue in Celery](#)



But I have not been able to solve my issue using the methods described in those threads.

I'm using Redis as backend, but I would like to have a backend independent and flexible solution, especially for tests.

This is my current situation: I've defined an `EnhancedCelery` class which inherits from `Celery` and adds a couple of methods, specifically `get_queue_size()` is the one I'm trying to properly implement/test.

The following is the code in my test case:

```
celery_test_app = EnhancedCelery(__name__)

# this is needed to avoid exception for ping command
# which is automatically triggered by the worker once started
celery_test_app.loader.import_module('celery.contrib.testing.tasks')

# in memory backend
celery_test_app.conf.broker_url = 'memory://'
celery_test_app.conf.result_backend = 'cache+memory://'

# We have to setup queues manually,
# since it seems that auto queue creation doesn't work in tests :(
celery_test_app.conf.task_create_missing_queues = False
celery_test_app.conf.task_default_queue = 'default'
celery_test_app.conf.task_queues = (
    Queue('default', routing_key='task.#'),
    Queue('queue_1', routing_key='q1'),
    Queue('queue_2', routing_key='q2'),
    Queue('queue_3', routing_key='q3'),
)
celery_test_app.conf.task_default_exchange = 'tasks'
celery_test_app.conf.task_default_exchange_type = 'topic'
celery_test_app.conf.task_default_routing_key = 'task.default'
celery_test_app.conf.task_routes = {
    'sample_task': {
        'queue': 'default',
        'routing_key': 'task.default',
    },
    'sample_task_in_queue_1': {
        'queue': 'queue_1',
        'routing_key': 'q1',
    },
    'sample_task_in_queue_2': {
        'queue': 'queue_2',
        'routing_key': 'q2',
    },
    'sample_task_in_queue_3': {
        'queue': 'queue_3',
        'routing_key': 'q3',
    },
}
```

```

}

@celery_test_app.task()
def sample_task():
    return 'sample_task_result'

@celery_test_app.task(queue='queue_1')
def sample_task_in_queue_1():
    return 'sample_task_in_queue_1_result'

@celery_test_app.task(queue='queue_2')
def sample_task_in_queue_2():
    return 'sample_task_in_queue_2_result'

@celery_test_app.task(queue='queue_3')
def sample_task_in_queue_3():
    return 'sample_task_in_queue_3_result'

class EnhancedCeleryTest(TestCase):
    def test_get_queue_size_returns_expected_value(self):
        def add_task(task):
            task.apply_async()

        with start_worker(celery_test_app):
            for _ in range(7):
                add_task(sample_task_in_queue_1)

            for _ in range(4):
                add_task(sample_task_in_queue_2)

            for _ in range(2):
                add_task(sample_task_in_queue_3)

            self.assertEqual(celery_test_app.get_queue_size('queue_1'), 7)
            self.assertEqual(celery_test_app.get_queue_size('queue_2'), 4)
            self.assertEqual(celery_test_app.get_queue_size('queue_3'), 2)

```

Here are my attempts to implement `get_queue_size()`:

1. This always returns zero (`jobs == 0`):

```

def get_queue_size(self, queue_name: str) -> Optional[int]:
    with self.connection_or_acquire() as connection:
        channel = connection.default_channel

        try:
            name, jobs, consumers = channel.queue_declare(queue=queue_name,
passive=True)
            return jobs
        except (ChannelError, NotFound):
            pass

```

2. This also always returns zero:

```

def get_queue_size(self, queue_name: str) -> Optional[int]:
    inspection = self.control.inspect()

    return inspection.active() # zero!

    # or:
    return inspection.scheduled() # zero!

```

```
# or:
return inspection.reserved() # zero!
```

3. This works by returning the expected number for each queue, but only in the test environment, because the `channel.queues` property does not exist when using the redis backend:

```
def get_queue_size(self, queue_name: str) -> Optional[int]:
    with self.connection_or_acquire() as connection:
        channel = connection.default_channel

        if hasattr(channel, 'queues'):
            queue = channel.queues.get(queue_name)

            if queue is not None:
                return queue.unfinished_tasks
```

python redis celery

Share Improve this question Follow

edited Dec 31, 2020 at 19:13
damon
14.5k 14 57 75

asked Apr 29, 2020 at 13:16
daveoncode
18.9k 15 104 159

3 Answers

Sorted by: Highest score (default)

- ▲

5

None of the solutions you mentioned are entirely correct in my humble opinion. As you already mentioned this is backend-specific so you would have to wrap handlers for all backends supported by Celery to provide backend-agnostic queue inspection. In the Redis case you have to directly connect to Redis and LLEN the queue you want to inspect. In the case of RabbitMQ you find this information in completely different way. Same story with SQS...
- ▼

This has all been discussed in the [Retrieve list of tasks in a queue in Celery](#) thread...
- 🔖

🕒

Finally, there is a reason why Celery does not provide this functionality out of box - the information is, I believe, useless. By the time you get what is in the queue it may already be empty!
- If you want to monitor what is going on with your queues I suggest another approach. - [Write your own real-time monitor](#). The example just captures task-failed events, but you should be able to modify it easily to capture all events you care about, and gather data about those tasks (queue, time, host it was executed on, etc). [Clearly](#) is an example how it is done in a more serious project.

Share Improve this answer Follow

edited Nov 23, 2022 at 9:11

answered Apr 29, 2020 at 13:27
DejanLekic
18.8k 4 46 77

- ▲

1

You can see how it's implemented in the Flower (real-time monitor for Celery) [here](#) They have different Broker class implementation for [redis](#) and [rabbitmq](#).
- ▼

Another way - use celery's [task events](#): calculate how many tasks were sent and how many were succeed/failed

Share Improve this answer Follow

answered Apr 29, 2020 at 13:38
Andrey Khoronko
653 3 6

Celery + Redis backend: How to limit queue size?

Asked 3 years, 11 months ago Modified 11 months ago Viewed 3k times

▲

4

▼

Is there a way to limit queue size when I run [Celery](#) with [Redis](#) backend?
Something like [x-max-length](#) in queue predeclare for [rabbitmq](#)

▼

python

redis

celery


celeryd

🔖

🕒

Share Follow

asked Oct 9, 2019 at 13:46



Maksym Polshcha

18k ● 8 ● 52 ● 77

3 Answers

Sorted by:

Highest score (default) ⚡

▲


1

▼

As far as I know that is not possible with Redis as backend.

Share Follow

answered Oct 9, 2019 at 14:13



DejanLekic

18.8k ● 4 ● 46 ● 77

🔖

🕒

Seems to be true. – [Maksym Polshcha](#) Oct 9, 2019 at 14:28

▲


0

▼

I think you are looking for prefetch limits in celery. Check it out [docs](#).

Share Follow

answered Oct 9, 2019 at 13:49



Bharat Gera

800 ● 1 ● 4 ● 13

🔖

🕒

The prefetch limit is a limit for the number of tasks a worker can reserve for itself. I need to limit my queues. – [Maksym Polshcha](#) Oct 9, 2019 at 13:59

That's interesting! Limiting a queue for executing the task, I haven't come across anything like that before. Please do share if you find out the way for the same.
– [Bharat Gera](#) Oct 9, 2019 at 15:36

Trouble trying to get size of Celery queue using redis-cli (for a Django app)

Asked 1 year, 11 months ago Modified 1 year, 11 months ago Viewed 2k times



3



I'm using `Django==2.2.24` and `celery[redis]==4.4.7`.

I want to get the length of my celery queues, so that I can use this information for autoscaling purposes in AWS EC2.

I found the following piece of documentation:

<https://docs.celeryproject.org/en/v4.4.7/userguide/monitoring.html#redis>

Redis

If you're using Redis as the broker, you can monitor the Celery cluster using the `redis-cli` command to list lengths of queues. Inspecting queues

Finding the number of tasks in a queue:

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

The default queue is named `celery`. To get all available queues, invoke:

```
$ redis-cli -h HOST -p PORT -n DATABASE_NUMBER keys \*
```

Note

Queue keys only exists when there are tasks in them, so if a key doesn't exist it simply means there are no messages in that queue. This is because in Redis a list with no elements in it is automatically removed, and hence it won't show up in the keys command output, and llen for that list returns 0. Also, if you're using Redis for other purposes, the output of the keys command will include unrelated values stored in the database. The recommended way around this is to use a dedicated DATABASE_NUMBER for Celery, you can also use database numbers to separate Celery applications from each other (virtual hosts), but this won't affect the monitoring events used by for example Flower as Redis pub/sub commands are global rather than database based.

Now, my Celery configuration (in Django) has the following relevant part:

```
CELERY_QUEUES = (  
    Queue('default', Exchange('default'), routing_key='default'),  
    Queue('email', Exchange('email'), routing_key='email'),  
    Queue('haystack', Exchange('haystack'), routing_key='haystack'),  
    Queue('thumbnails', Exchange('thumbnails'), routing_key='thumbnails'),  
)
```

So I tried this:

```
$ redis-cli -n 0 -h ${MY_REDIS_HOST} -p 6379 llen haystack
```

(yes, celery is configured to use redis database number 0)

I tried my 4 queues, and I always get 0, when this is simply not possible. Some of these queues are usually very active, or my website wouldn't be working properly.

One key part of the documentation is that I can list the available queues, so I tried it:

```
$ redis-cli -n 0 -h ${MY_REDIS_HOST} -p 6379 keys \*
```

And I get about 20,000 lines of something like this:

```
celery-task-meta-b30fb605-d7b6-48db-b8cd-493458566876
celery-task-meta-e10ec56c-6601-420b-9f87-de6455968e76
celery-task-meta-14558a3a-1153-4f02-91f8-614bc29f6775
celery-task-meta-4c266854-512b-48af-8356-c786c507eb9e
celery-task-meta-e4ad4298-3d74-4986-8831-4c4d3c3e79f2
celery-task-meta-dfab0202-3975-46ce-9670-0d4cf3e278db
celery-task-meta-494fcb21-5995-495d-8980-0d8aa7edf0b8
celery-task-meta-345c4857-87f9-4e3f-8028-a6ef8cf93f5d
celery-task-meta-a4a48d00-68dc-4d30-87dd-869d2a20c347
celery-task-meta-d14fc394-6415-442b-8a5d-c9a4f37a9509
```

If I exclude all the `celery-task-meta` lines:

```
$ redis-cli -n 0 -h ${MY_REDIS_HOST} -p 6379 keys \* | grep -v celery-task-meta
```

I get this:

```
_kombu.binding.celeryev
_kombu.binding.default
_kombu.binding.thumbnails
_kombu.binding.email
unacked
_kombu.binding.celery.pidbox
_kombu.binding.haystack
unacked_index
_kombu.binding.reply.celery.pidbox
```

I tried to use the `celery` CLI to get the information, and this is some relevant output:

```
$ celery --app my-app inspect active_queues

-> celery@683a8e8bc84f: OK
  * {'name': 'thumbnails', 'exchange': {'name': 'thumbnails', 'type': 'direct',
'arguments': None, 'durable': True, 'passive': False, 'auto_delete': False,
'delivery_mode': None, 'no_declare': False}, 'routing_key': 'thumbnails',
'queue_arguments': None, 'binding_arguments': None, 'consumer_arguments': None,
'durable': True, 'exclusive': False, 'auto_delete': False, 'no_ack': False,
'alias': None, 'bindings': [], 'no_declare': None, 'expires': None,
'message_ttl': None, 'max_length': None, 'max_length_bytes': None,
'max_priority': None}
-> celery@bf11d4c3bd6f: OK
  * {'name': 'email', 'exchange': {'name': 'email', 'type': 'direct',
'arguments': None, 'durable': True, 'passive': False, 'auto_delete': False,
'delivery_mode': None, 'no_declare': False}, 'routing_key': 'email',
'queue_arguments': None, 'binding_arguments': None, 'consumer_arguments': None,
'durable': True, 'exclusive': False, 'auto_delete': False, 'no_ack': False,
'alias': None, 'bindings': [], 'no_declare': None, 'expires': None,
'message_ttl': None, 'max_length': None, 'max_length_bytes': None,
'max_priority': None}
-> celery@86151417b361: OK
  * {'name': 'default', 'exchange': {'name': 'default', 'type': 'direct',
'arguments': None, 'durable': True, 'passive': False, 'auto_delete': False,
'delivery_mode': None, 'no_declare': False}, 'routing_key': 'default',
'queue_arguments': None, 'binding_arguments': None, 'consumer_arguments': None,
```

```
'durable': True, 'exclusive': False, 'auto_delete': False, 'no_ack': False,
'alias': None, 'bindings': [], 'no_declare': None, 'expires': None,
'message_ttl': None, 'max_length': None, 'max_length_bytes': None,
'max_priority': None}
-> celery@9a5360a82f14: OK
* {'name': 'haystack', 'exchange': {'name': 'haystack', 'type': 'direct',
'arguments': None, 'durable': True, 'passive': False, 'auto_delete': False,
'delivery_mode': None, 'no_declare': False}, 'routing_key': 'haystack',
'queue_arguments': None, 'binding_arguments': None, 'consumer_arguments': None,
'durable': True, 'exclusive': False, 'auto_delete': False, 'no_ack': False,
'alias': None, 'bindings': [], 'no_declare': None, 'expires': None,
'message_ttl': None, 'max_length': None, 'max_length_bytes': None,
'max_priority': None}
```

and

```
$ celery --app my-app inspect scheduled

-> celery@683a8e8bc84f: OK
- empty -
-> celery@86151417b361: OK
- empty -
-> celery@bf11d4c3bd6f: OK
- empty -
-> celery@9a5360a82f14: OK
- empty -
```

The command above seems to work well: if there are active tasks, the task is shown there, even tho in my copy/paste it says `empty`.

So, does anybody know what I might be doing wrong and why I can't get the real size of my queues?

Thanks!

django

redis

celery

Share Improve this question Follow

edited Oct 13, 2021 at 17:01

asked Oct 13, 2021 at 16:36



Salvatore Iovene

2,074 1 17 31

Celery processes through tasks extremely fast (at least in our case), so it could be that you just got unlucky and queried Redis when there were no tasks in the queue. I would try making a task that sleeps for a long time in your test environment and call it in a loop so Celery can't process through them all before you query Redis.

– [Almenon](#) Sep 8, 2022 at 20:03