Taskrabbit

Release 0.1a

Chris Lawlor

Jun 12, 2022

CONTENTS:

CHAPTER ONE

QUICK START

Try out Taskrabbit with Docker, using a simple provided Celery app.

1.1 Setup

Clone the repository:

```
git clone https://github.com/chrislawlor/taskrabbit.git
cd taskrabbit
```

Now, create a `taskrabbit.ini` file in the local directory with the following contents:

```
[taskrabbit]
store = taskrabbit.stores.postgres.PostgresTaskStore
log_level = INFO

[store]
username = taskrabbit
password = password
host = postgres
port = 5432
db = taskrabbit

[rabbitmq]
username = guest
password = guest
host = rabbit
consumer_prefetch_count = 10
```

1.2 Demonstration

1. Start the RabbitMQ and PostgreSQL servers:

   ```
   docker-compose up -d rabbit postgres
   ```

2. In a browser, log in to the management interface at http://localhost:15672, using the credentials `guest`, `guest` for the username and password.

3. In the terminal run `make tasks`. Watch the overview in the RabbitMQ management interface as tasks are created. In the Queues tab (http://localhost:15672/#/queues), we can see that tasks have been created in two queues: arithmetic, and geometry.

   One of the features of Taskrabbit is the capability of removing duplicate tasks. So that we can demonstrate this later, run `make tasks` again now.

4. Build the taskrabbit image:

   ```
   make build
   ```

5. Get a shell in the taskrabbit container:

   ```
   docker-compose run --rm app bash
   ```

6. Run `taskr --help` to see the help message, which shows a list of subcommands.

7. We'll use the `drain` command to drain tasks from the arithmetic queue:

   ```
   taskr drain arithmetic
   ```

8. After a few moments, we should see a summary of the tasks we've drained from the queue:

   ```
   Stored tasks:
   ┌────────────────┬───────┐
   │ Task           │ Count │
   ├────────────────┼───────┤
   │ tasks.add      │ 20000 │
   ├────────────────┼───────┤
   │ tasks.multiply │ 20000 │
   └────────────────┴───────┘
   ```

9. Now that we have some stored tasks, we can use the `store` subcommand to interact with them. Lets take a look at the first task in the store:

   ```
   taskr store list --limit 1

   ┌──────────────────────────────────────┬────────────────┬────────┬────────┬─────────────
   │ ID                                   │ Task           │ Args   │ Kwargs │ Routing␣
   ↪Key   │
   ├──────────────────────────────────────┼────────────────┼────────┼────────┼─────────────
   │ e1386dc8-faf1-4af2-9210-18c5e00206f1 │ tasks.multiply │ (1, 1) │ {}     │␣
   ↪arithmetic.# │
   └──────────────────────────────────────┴────────────────┴────────┴────────┴─────────────
   ```

   we can see that a task named `tasks.multiply` has been retrieved, with the positional arguments `(1, 1)` and no keyword arguments.

10. Earlier, we ran `make tasks` twice, which put duplicated tasks in our task queues. Let's remove them now:

    ```
    taskr store dedupe
    ```

    Since this is a destructive operation, Taskrabbit asks us to confirm our command. Enter *y* to confirm. We should now see the confirmation message:

    ```
    Removed 20000 duplicate tasks from the store.
    ```

Because our tasks are stored in a relational database, we could also connect to that database and use SQL to further interact with our tasks before we decide to republish them.

11. Now that we've removed the duplicate tasks, let's restore them to the queue. To do this, we'll publish tasks to the "tasks" exchange. Once again, Taskrabbit will ask us to confirm our command, and will display a summary of the published tasks.

    Let's start by publishing the "add" tasks first:

    ```
    taskr fill tasks --task-name tasks.add
    Publish tasks to the tasks exchange? [y/N]: y
    ───────────────────────
    │ Task           │ Count │

    │ tasks.add      │ 10000 │
    └───────────────────────
    ```

12. In the RabbitMQ management interface, we should now see 10,000 tasks in the arithmetic queue.

13. We can publish all remaining stored tasks:

    ```
    taskr
    Publish tasks to the tasks exchange? [y/N]: y
    ───────────────────────
    │ Task           │ Count │

    │ tasks.multiply │ 10000 │
    └───────────────────────
    ```

Congratulations! You've successfully used Taskrabbit to pull tasks from a queue, remove duplicates, and re-publish them.

Finally, we can exit our app container, and shut down our remaining containers:

```
exit
docker-compose down
```

CHAPTER

TWO

TASK STORES

Task stores provide an interface and persistance layer for `StoredTask` objects.

2.1 `StoredTask`

2.2 Base `TaskStore`

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search