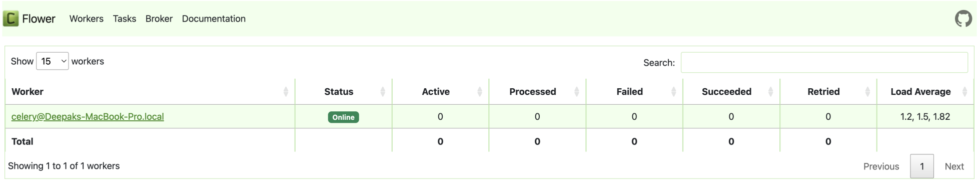
Chapter - 8 : Celery Management using Flower

Flower is an open source web application for monitoring and managing celery clusters. It provides real time information about the status of celery workers and tasks

Once the flower is installed, we can connect the same to the message broker to get the information about the celery workeres and tasks

```
celery --broker=amqp://guest:guest@localhost:5672// flower
```

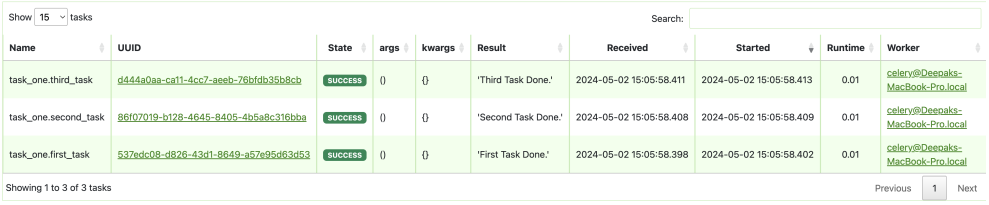


Worker	Status	Active	Processed	Failed	Succeeded	Retried	Load Average
celery@Deepaks-MacBook-Pro.local	Online	0	0	0	0	0	1.2, 1.5, 1.82
Total		**0**	**0**	**0**	**0**	**0**	

Showing 1 to 1 of 1 workers

Flower Dashboard

Once we run the tasks, then all tasks will be visible in the flower



Name	UUID	State	args	kwargs	Result	Received	Started	Runtime	Worker
task_one.third_task	d444a0aa-ca11-4cc7-aeeb-76bfdb35b8cb	SUCCESS	()	{}	'Third Task Done.'	2024-05-02 15:05:58.411	2024-05-02 15:05:58.413	0.01	celery@Deepaks-MacBook-Pro.local
task_one.second_task	86f07019-b128-4645-8405-4b5a8c316bba	SUCCESS	()	{}	'Second Task Done.'	2024-05-02 15:05:58.408	2024-05-02 15:05:58.409	0.01	celery@Deepaks-MacBook-Pro.local
task_one.first_task	537edc08-d826-43d1-8649-a57e95d63d53	SUCCESS	()	{}	'First Task Done.'	2024-05-02 15:05:58.398	2024-05-02 15:05:58.402	0.01	celery@Deepaks-MacBook-Pro.local

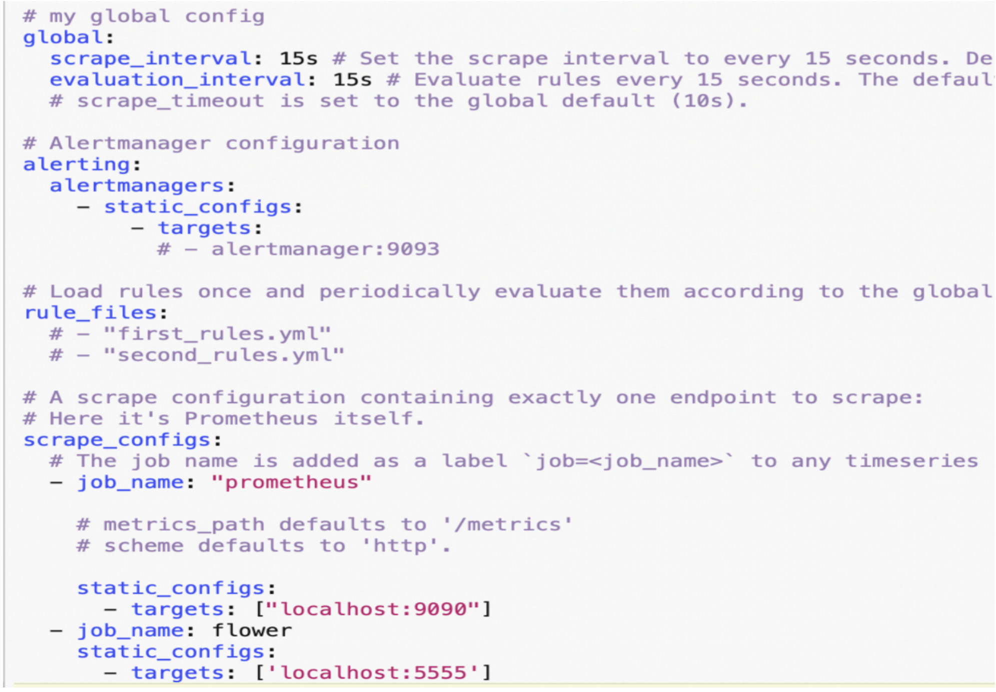
Showing 1 to 3 of 3 tasks

Tasks in Flower

Using Prometheus

`Prometheus` *is a time series database, we can store the flower events in this database to be able to revist the same later*

Here is the config file which is needed for the Prometheus



```
# my global config
global:
  scrape_interval: 15s # Set the scrape interval to every 15 seconds. De
  evaluation_interval: 15s # Evaluate rules every 15 seconds. The defaul
  # scrape_timeout is set to the global default (10s).

# Alertmanager configuration
alerting:
  alertmanagers:
    - static_configs:
        - targets:
          # - alertmanager:9093

# Load rules once and periodically evaluate them according to the global
rule_files:
  # - "first_rules.yml"
  # - "second_rules.yml"

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries
  - job_name: "prometheus"

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

    static_configs:
      - targets: ["localhost:9090"]
  - job_name: flower
    static_configs:
      - targets: ['localhost:5555']
```

Prometheus Config File

Using Grafana

We can use grafana using the docker image. Grafana can be used to better visualize the data from the Prometheus

```
**docker run -d -p 3000:3000 --name=grafana grafana/grafana-enterprise**
```

if we're using docker then the Prometheus URL will be

```
http://host.docker.internal:9090/
```