CELERY

Distributed Task Queue for Python Applications

Traitement Asynchrone | Scheduling | Scalabilité Horizontale

Pourquoi utiliser Celery ?

Le Problème : Blocage

Lorsqu'un utilisateur attend la fin d'une tâche lourde (ex: envoi d'email, traitement d'image), l'application est bloquée et l'UX dégradée.

La Solution : Celery

Déporter le travail vers des "Workers" en arrière-plan. La réponse HTTP est immédiate, le travail est fait "plus tard".

Caractéristiques Clés

Asynchronisme

Exécution hors du cycle de requête HTTP principal.

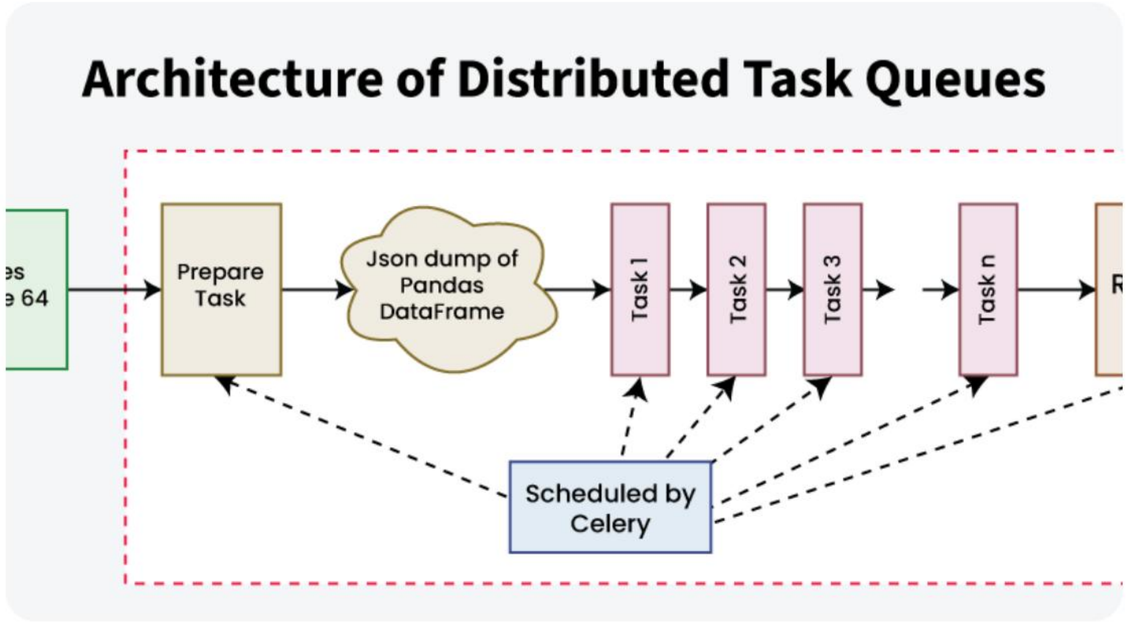
Scheduling

Tâches planifiées (Celery Beat) comme un Cron distribué.

Résilience

Gestion automatique des échecs et tentatives (Retries).

L'Architecture Celery



- **Producers :** Vos apps (Django, Flask) qui créent les tâches.
- **Broker :** L'intermédiaire (Redis, RabbitMQ) qui stocke la file.
- **Workers :** Les serveurs qui consomment et exécutent les tâches.
- **Result Backend :** Où le résultat final est stocké.

Les Piliers de l'Écosystème

RabbitMQ
Flexible messaging and streaming

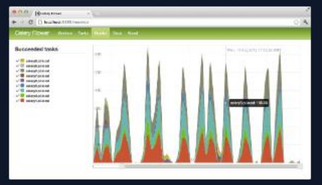
RabbitMQ

Le broker recommandé pour la robustesse en production.

Redis

Redis

Le broker le plus rapide et facile à configurer.



Flower

Outil de monitoring temps réel pour vos workers.

Cas d'usage concrets

Type de tâche	Exemple	Impact UX
Notifications	Envoi d'e-mails de bienvenue ou SMS.	Instantanné pour l'utilisateur.
Média	Redimensionnement d'images, transcodage vidéo.	Traitement transparent en tâche de fond.
Data Processing	Génération de rapports PDF, imports CSV massifs.	Évite le timeout du serveur web.
Maintenance	Nettoyage quotidien de la base de données.	Automatisation via Celery Beat.

Scalabilité Horizontale

WORKERS

Plus de charge ? Plus de serveurs !

Celery permet d'ajouter des workers sur n'importe quel serveur distant connecté au broker.

Vous pouvez dédier des serveurs spécifiques aux tâches CPU-intensives et d'autres aux tâches I/O.

Workflow d'une Tâche

Déclenchement

L'application appelle .delay()

Mise en file

Message envoyé au Broker

Réception

Un worker libre récupère la tâche

Fin d'exécution

Résultat écrit dans le Backend

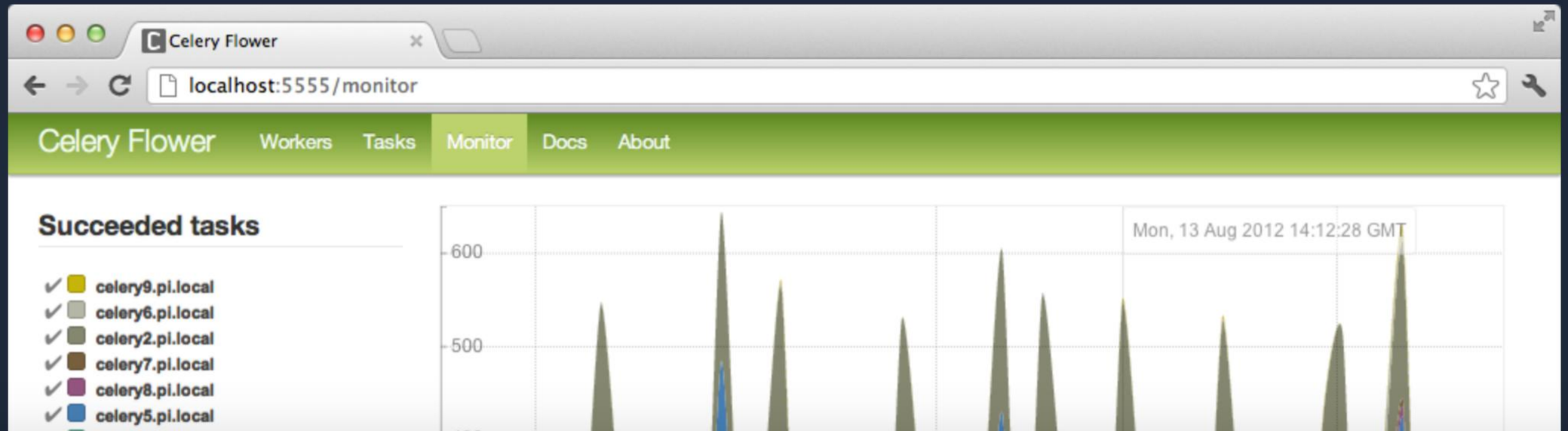
Gain de Performance (Réponse Web)

**Celery libère le thread de l'application web immédiatement après avoir mis le message dans le broker.*

Flower : Le Cockpit

Flower est l'outil indispensable pour superviser votre infrastructure Celery en temps réel.

- Statut des Workers (Up/Down)
- Statistiques de réussite/échec
- Inspection des files d'attente
- Temps d'exécution moyen



"Celery allows your application to focus on giving users answers, while the heavy work is done by the experts in the background."

— Adage de l'architecture distribuée

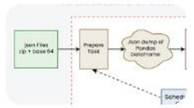
Questions ?

Prêt à asynchroniser vos applications ?

github.com/celery/celery

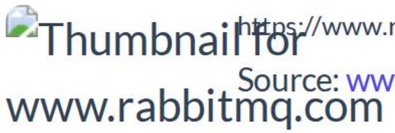
docs.celeryq.dev

Image Sources



https://media.geeksforgeeks.org/wp-content/uploads/20240912133124/Architecture-of-Distributed-Task-Queues.png

Source: www.geeksforgeeks.org

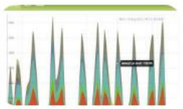


https://www.rabbitmq.com/img/rabbitmq-social-media-card.svg

Source: www.rabbitmq.com

https://1000logos.net/wp-content/uploads/2020/08/Redis-Logo.jpg

Source: 1000logos.net



https://docs.celeryq.dev/en/4.0/_images/monitor.png

Source: docs.celeryq.dev