Teach Me Skills

Занятие 27.
celery + celery beat

1

Background tasks

Для чего нужны бэкграунд задачи

- Выполнение cpu-bound вычислений
- Выполнение долгих задач (рассылка email и т д)
- Выполнение периодических задач по расписанию
- Быстрый ответ конечному пользователю
- Отсутсвие блокировки http потока
- Отделение от контекста http запросов

Инструменты

- **celery**
- **rq** (redis queue) - простой и легковесный
- **apscheduler** - библиотека для планирования задач
- **django background tasks**

Возможности celery

- Выполнять задания асинхронно или синхронно
- Выполнять периодические задания
- Выполнять отложенные задания
- Распределенное выполнение (может быть запущен на N серверах)
- В пределах одного worker'а возможно конкурентное выполнение нескольких задач(одновременно)
- Выполнять задание повторно, если вылез exception
- Ограничивать количество заданий в единицу времени (rate limit, для задания или глобально)
- Routing заданий (какому worker'у что делать)
- Несложно мониторить выполнение заданий
- Выполнять подзадания
- Присылать отчеты об exception'ах на email
- Проверять выполнилось ли задание (удобно для построения Ajax приложений, где юзер ждет факта завершения)

Схема работы celery

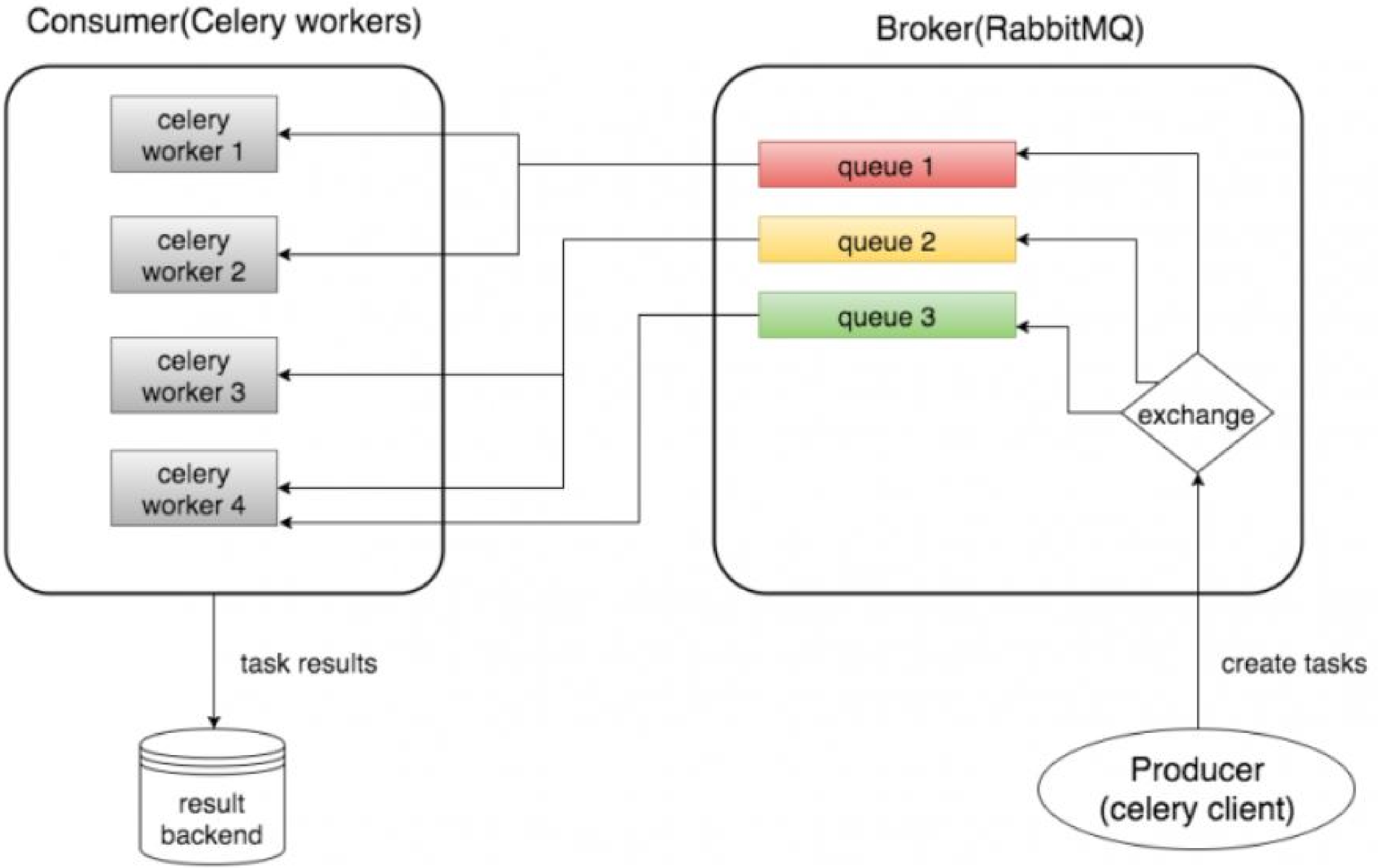
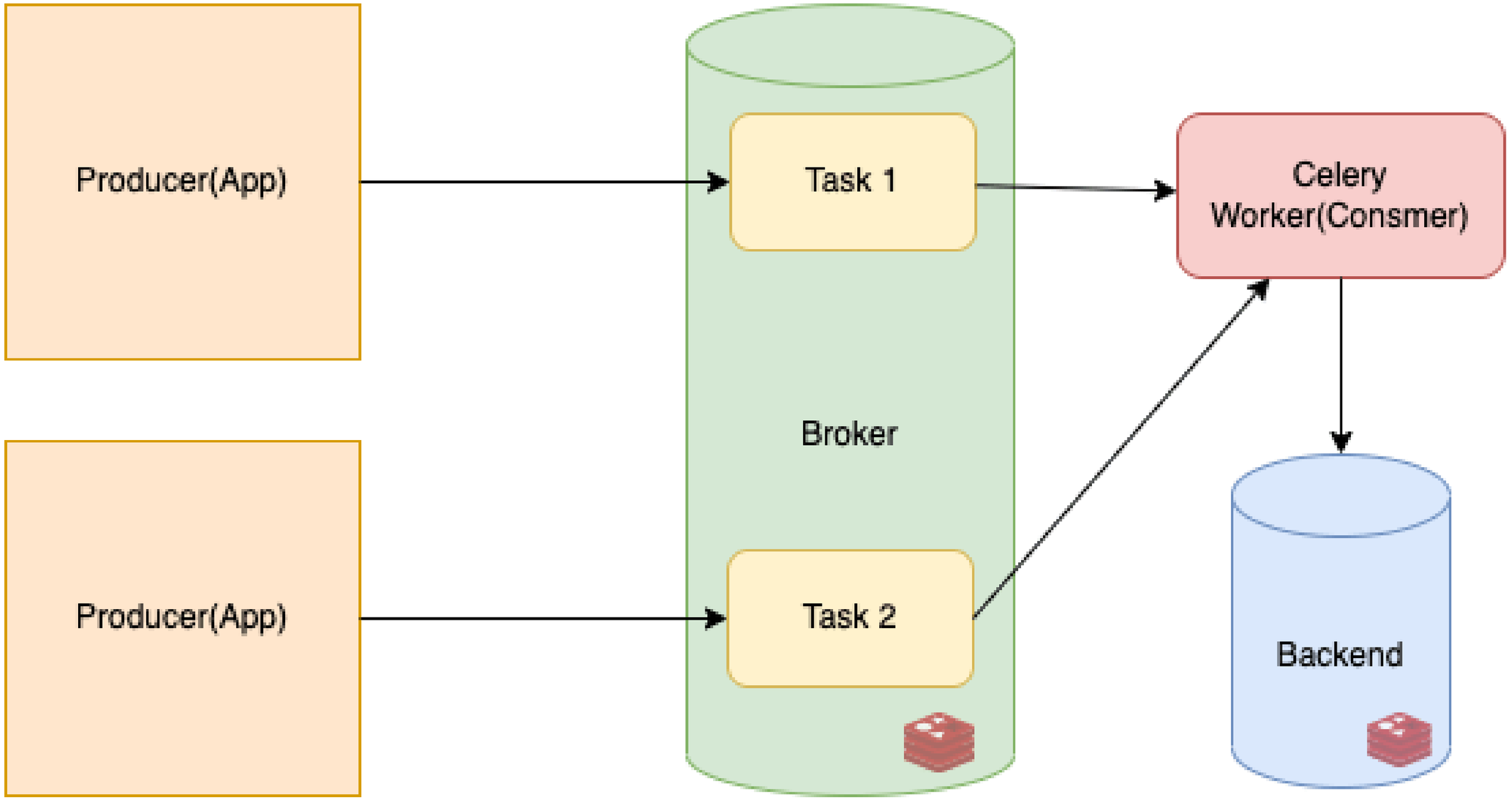


Схема работы celery



Celery beat

- позволяет запускать задачи по расписанию в определенное время или с определенной периодичностью
- автоматизация задач
- управление временем выполнения (распределение нагрузки)
- гибкость и расширяемость

Celery vs django background tasks

- **Django Background Tasks** - это простой плагин для Django, который выполняет задачи в рамках основного процесса Django. Он использует базу данных Django для хранения и планирования задач. С другой стороны, Celery - это отдельный фреймворк для асинхронной обработки задач, который использует распределенную очередь задач и рабочие процессы для выполнения задач.
- **Масштабируемость и производительность:** Celery предлагает высокую масштабируемость, позволяя распределять задачи на несколько рабочих процессов или серверов. Это особенно полезно при обработке больших объемов задач или выполнении длительных операций. Django Background Tasks, в свою очередь, выполняет задачи в рамках основного процесса Django, что может ограничивать производительность и масштабируемость при работе с большими объемами задач

Celery vs django background tasks

- **Отказоустойчивость**: Celery обладает встроенной отказоустойчивостью, позволяя обрабатывать ошибки и автоматически повторять выполнение задач в случае сбоев. Он также предоставляет механизмы мониторинга и управления выполнением задач. Django Background Tasks не имеет таких встроенных механизмов отказоустойчивости и требует дополнительной настройки для обработки ошибок и повторного выполнения задач.
- **Функциональность**: Celery предлагает богатый набор функциональных возможностей, таких как планирование задач, мониторинг выполнения, результаты задач, параллельная обработка и многое другое. Django Background Tasks имеет более ограниченный набор функций и ориентирован на базовую асинхронную обработку задач в рамках Django.

2

Брокеры сообщений

Брокер сообщений

Брокер сообщений - это промежуточное программное обеспечение или сервис, который служит для передачи сообщений между различными компонентами системы или приложениями.

В контексте Celery и других систем асинхронной обработки задач, брокер сообщений играет важную роль в установлении связи между отправителем задачи и ее исполнителем.

Основная функция брокера сообщений в Celery и аналогичных системах заключается в приеме, хранении и доставке сообщений о задачах от отправителя (поставщика задачи) к исполнителю (рабочему процессу или рабочему узлу). Брокер сообщений служит в качестве посредника, который обеспечивает безопасную и надежную передачу сообщений.

Брокер сообщений

1. **RabbitMQ** является мощным брокером сообщений, реализующим протокол AMQP (Advanced Message Queuing Protocol). Он обеспечивает надежную доставку сообщений, обработку очередей и управление обменами сообщениями.
2. **Redis** - это база данных, которая также может использоваться в качестве брокера сообщений для систем асинхронной обработки задач. Он предоставляет высокую производительность, низкую задержку и поддерживает различные структуры данных, которые могут быть полезны при работе с очередями задач.
3. **Apache Kafka** является распределенной платформой для потоковых данных, которая также может использоваться в качестве брокера сообщений. Он предоставляет высокую масштабируемость, отказоустойчивость и обработку данных в реальном времени.
4. **Amazon Simple Queue Service (SQS)**: Amazon SQS является управляемой службой очередей сообщений, предоставляемой Amazon Web Services (AWS). Он обеспечивает масштабируемость, надежность и интеграцию с другими сервисами AWS.

4

Домашнее задание

Домашнее задание

- Настроить celery в своем проекте
- Добавить задачу, которая по окончанию дня логгирует, какие задачи были завершены сегодня для каждого пользователя в файл. Имя файла должно содержать id задачи, name и updated_at

Спасибо за внимание!