The dos and don’ts of task queues

EuroPython 2019

Petr Stehlík
@petrstehlik

$ whoami

Petr Stehlík

Python developer @ Kiwi.com

Finance tribe

Outline

1. Task queues
2. The story
3. Examples vs. reality
4. Final setup
5. How we do it in Kiwi.com
6. Lessons learned
7. Q&A

Task queues

What is a task queue

“parallel execution of discrete tasks without blocking”

- Not just Celery
- Major parts
  - Queue
  - Task – unit of work
  - Producer
  - Consumer

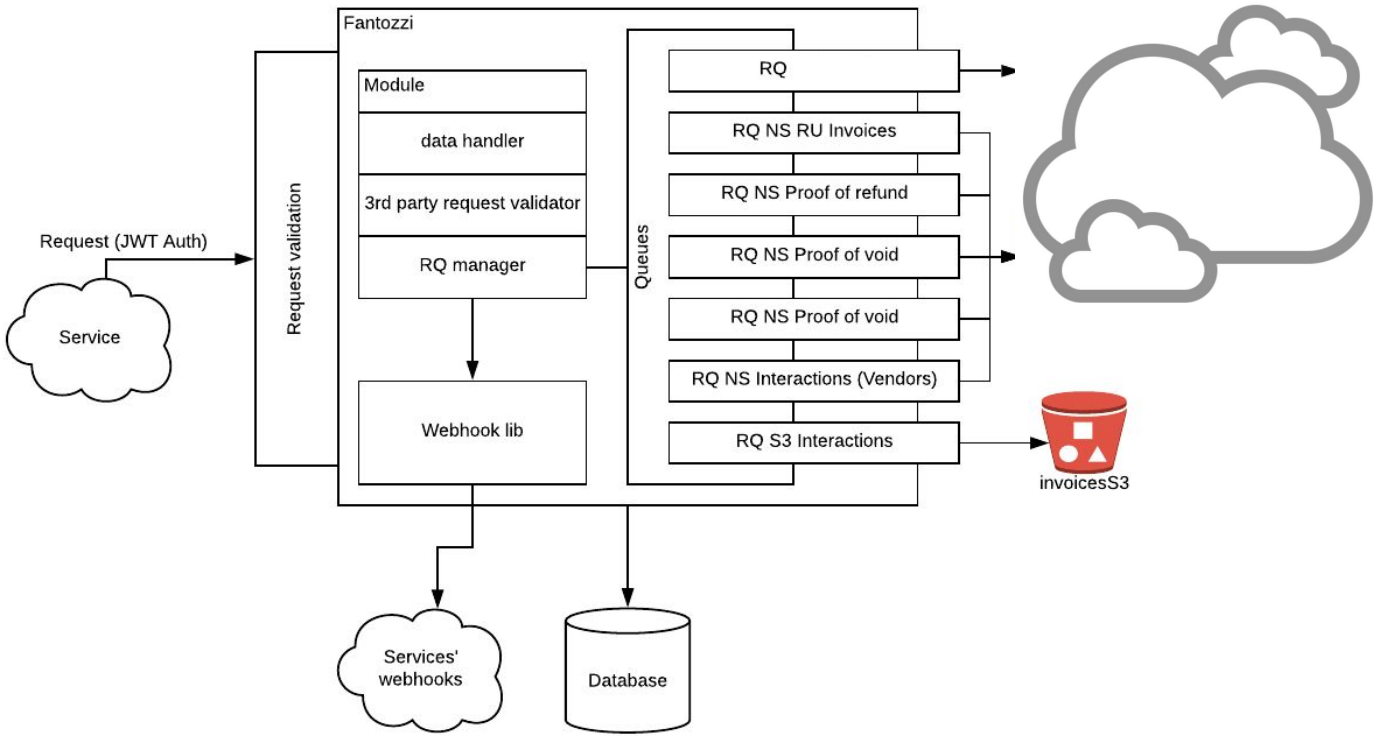
Source: DENÍK/Michal Kovář

For what is a task queue

- Decouple long-running task from a synchronous call
- Perform something periodically
- Break down software to more isolated pieces (when microservice is too big)
- Minimize wait time, latency and/or response time
- Increase throughput of the system

The story

The story


Fantozzi
Request (JWT Auth)
Service
Request validation
Module
data handler
3rd party request validator
RQ manager
Webhook lib
Queues
RQ
RQ NS RU Invoices
RQ NS Proof of refund
RQ NS Proof of void
RQ NS Proof of void
RQ NS Interactions (Vendors)
RQ S3 Interactions
invoicesS3
Services' webhooks
Database

The story

“New is always better.”

The story

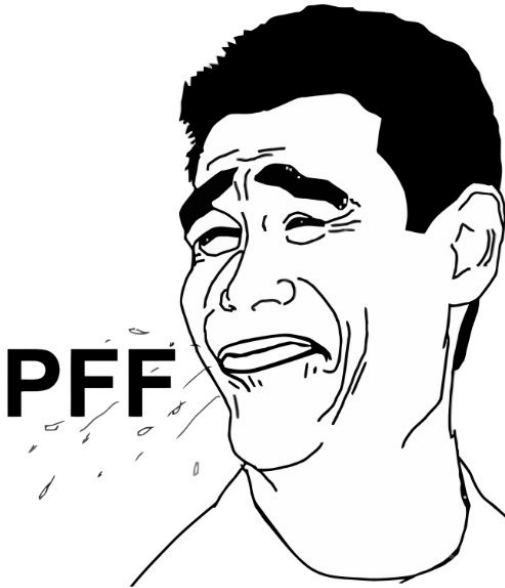
“Think outside the box.”

The story

“I know everything I need.”

The story

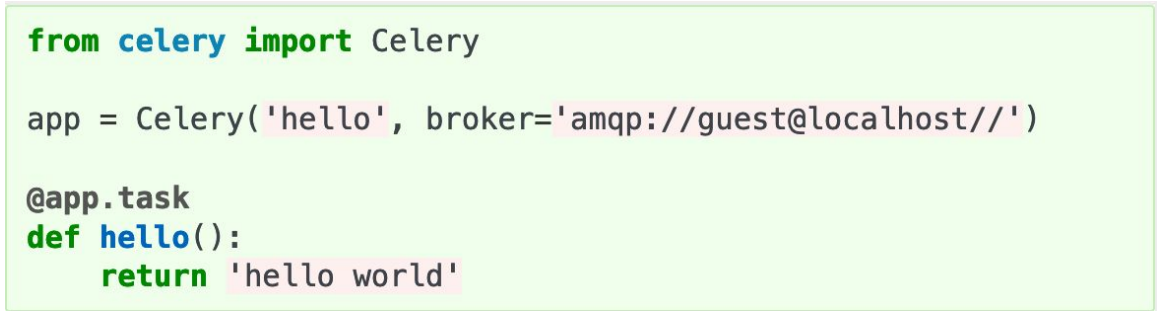
“I can do it better.”



Examples vs. reality

why it all happened

Example



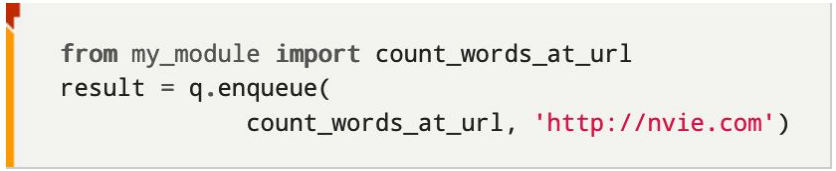
```python
from celery import Celery

app = Celery('hello', broker='amqp://guest@localhost//')

@app.task
def hello():
    return 'hello world'
```

```python
from redis import Redis
from rq import Queue

q = Queue(connection=Redis())
```



```python
from my_module import count_words_at_url
result = q.enqueue(
             count_words_at_url, 'http://nvie.com')
```

Reality

RQ

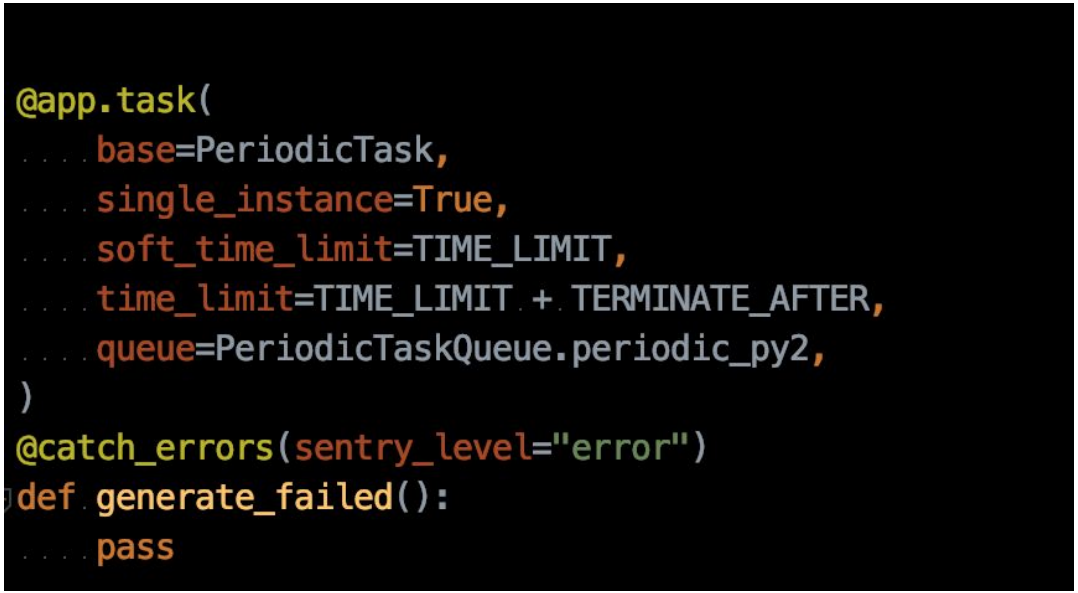
```python
@classmethod
def check_func(cls, res: requests.Response) -> Tuple[bool, str]:
    """Function checking job success based on return data."""
    raise NotImplementedError

@classmethod
def callback(cls, data_dict: dict) -> None:
    job = Job.fetch(data_dict.get("job_id"), connection=QMan.redis_connection)
    result_success, msg = cls.check_func(job.result)
    if result_success:
        return
    if not result_success and data_dict.get("call_count") < cls.call_count:
        log.warning(cls.name + " ,job failed.", reason=msg)
        job.kwargs.update({"data_dict": data_dict})
        cls.enqueue_func(job.func, *job.args, **job.kwargs)
        return
    return

@classmethod
def enqueue_func(cls, func: Callable, *args, **kwargs) -> str:
    """Add task and callback to queue."""
    job = cls.queue.enqueue(func, *args, **kwargs)
    data_dict = kwargs.pop("data_dict", {"job_id": job.id, "call_count": 0})
    data_dict["call_count"] += 1
    cls.queue.enqueue(cls.callback, data_dict)
    return job.id

@classmethod
def work(cls, *args, **kwargs):
    raise NotImplementedError
```

Reality



```python
@app.task(
    base=PeriodicTask,
    single_instance=True,
    soft_time_limit=TIME_LIMIT,
    time_limit=TIME_LIMIT + TERMINATE_AFTER,
    queue=PeriodicTaskQueue.periodic_py2,
)
@catch_errors(sentry_level="error")
def generate_failed():
    pass
```

18 Files 244 14

Final setup

Final setup

- Python + PostgreSQL
- **Flask**
- Connexion
- **Celery**
- Redis on AWS
- Multiple deploy targets
- Logz.io & Datadog
- Sentry
- PagerDuty

How we do it in Kiwi.com

In finance tribe

Kiwi.com | Finance Tribe toolset

- Python + PostgreSQL
- Flask/AioHttp
- Connexion
- Celery
- Redis on AWS
- Multiple deploy targets
- Logz.io & Datadog
- Sentry
- PagerDuty

Kiwi.com | Finance Tribe toolset

- Python
  - New projects always 3.6+
  - Old projects transitioning from 2.7 to 3.6
  - Monolith -> microservice architecture
- Flask/AioHttp
  - Our go-to framework
  - Boilerplates
  - Quick scaffolding
- Connexion
  - OpenAPI 3
  - Token-based authentication & authorization

```
connexion_app = App(__package__)

flask_app = connexion_app.app
flask_app.config.from_object(settings_object)
flask_app.config.update(**kwargs)
connexion_app.add_api(
    "schema.yaml", validate_responses=True, strict_validation=True,
)
connexion_app.add_error_handler(Error, sentry_error_handler)

db.init_app(flask_app)
db.app = flask_app

sentry.init_app(flask_app, dsn=flask_app.config["SENTRY_DSN"])

add_cli_commands(flask_app)

setup_logging(flask_app)
setup_datadog(flask_app)
setup_tracer(flask_app, "finance")
```

Kiwi.com | Finance Tribe toolset

- Celery
  - Follow the best practices (next section)
- Redis on AWS
  - Reliability
  - Easy to deploy

Kiwi.com toolset | Finance Tribe

- Multiple deploy targets
  - HTTP API
  - Workers
  - Etc.
  - Internal tool for deploying from Gitlab CI
- Logz.io & Datadog
  - Extensive logging
- Sentry
  - When something goes wrong
- PagerDuty
  - When something goes really wrong

```
deploy-production:
  extends: .crane
  when: manual
  variables:
    CRANE_ARGS: $CRANE_PRODUCTION_ARGS
  environment:
    name: production
    url: 'https://fantozzi.skypicker.com/ui'

deploy-workers:
  extends: .crane
  when: manual
  variables:
    CRANE_ARGS: $CRANE_WORKERS_ARGS
  environment:
    name: workers
    url: 'https://fantozzi.skypicker.com/ui'
```

Lessons learned

Lessons learned

Use Redis or AMQP broker (never a database)

```
app = Celery("exampleApp", broker_url="pyamqp://guest@localhost//")


$> pip install celery[redis]

app = Celery("exampleApp", broker_url="redis://127.0.0.1:6379/2")
```

Lessons learned

Pass simple objects to the tasks

Lessons learned

Do not wait for tasks inside tasks

Lessons learned

Set retry limit

```
@app.task(retry_kwargs={'max_retries': 5})
```

Lessons learned

Use `autoretry_for`

```
@app.task(autoretry_for=(NetworkError,), retry_kwargs={'max_retries': 5})
```

Lessons learned

Use `retry_backoff=True` and `retry_jitter=True`

```
@celery_app.task(
    autoretry_for=(NetworkError,),
    retry_backoff=True,  # disabled by default
    retry_jitter=True,  # enabled by default
    retry_kwargs={"max_retries": 5},
)
```

Lessons learned

Set hard and soft time limits

```
@celery_app.task(
    soft_time_limit=30,
    time_limit=60,
    autoretry_for=(NetworkError,),
    retry_backoff=True,  # disabled by default
    retry_jitter=True,  # enabled by default
    retry_kwargs={"max_retries": 5},
)
```

Lessons learned

Use `bind` for a bit of extra oomph (logs, handling, etc.)

```python
@celery_app.task(bind=True)
def get_user(self, user_id: str) -> object:
    """Get a user from external service identified by their ID."""
    try:
        res = get_user(user_id)
        statsd.increment("get_user", tags=["status:success"])
    except (requests.RequestException, ConnectionError) as e:
        statsd.increment("get_user", tags=["status:error"])
        log.error("get_user", message=e, user_id=user_id)
        raise self.retry(exc=e, max_retries=5, retry_jitter=True, retry_backoff=True)

    return res.json()
```

Lessons learned

Use separate queues for demanding tasks (set priorities)

```
app.conf.task_default_queue = 'default'
app.conf.task_queues = (
    Queue('default', routing_key='default.#'),
    Queue('fast', routing_key='fast.#'),
    Queue('slow', routing_key='slow.#'),
)

...

get_user.apply_async(args=["john.kiwi"], queue='fast')
```

Lessons learned

Prefer idempotency and atomicity

"Idempotence is the property of certain operations in mathematics and computer science, that can be applied multiple times without changing the result beyond the initial application."

- Wikipedia

“Atomic operation appears to the rest of the system to occur instantaneously. Atomicity is a guarantee of isolation from concurrent processes.

- Wikipedia

Lessons learned

- Use Redis or AMQP (RabbitMQ) broker (never a database)
- Pass simple objects to the tasks
- Do not wait for tasks inside tasks
- Set retry limit
- Use `autoretry_for`
- Use `retry_backoff=True` and `retry_jitter=True`
- Set hard and soft time limits
- Use `bind` for a bit of extra oomph in tasks (logging, handling, etc.)
- Use separate queues for demanding tasks (set priorities)
- Prefer idempotency and atomicity

Things to consider

- Sharing codebase between producer and consumer (producer must know everything about consumer and vica versa)
- Use celery to its full potential -> read celery’s docs
- Scalability of 3rd party APIs

Join our Wednesday party at Europython and win flight vouchers

More info @ meet.kiwi.com

Meet us at the booth #45

KIWI.COM

Any questions?

You can find me at

@petrstehlik & **petr.stehlik@kiwi.com**