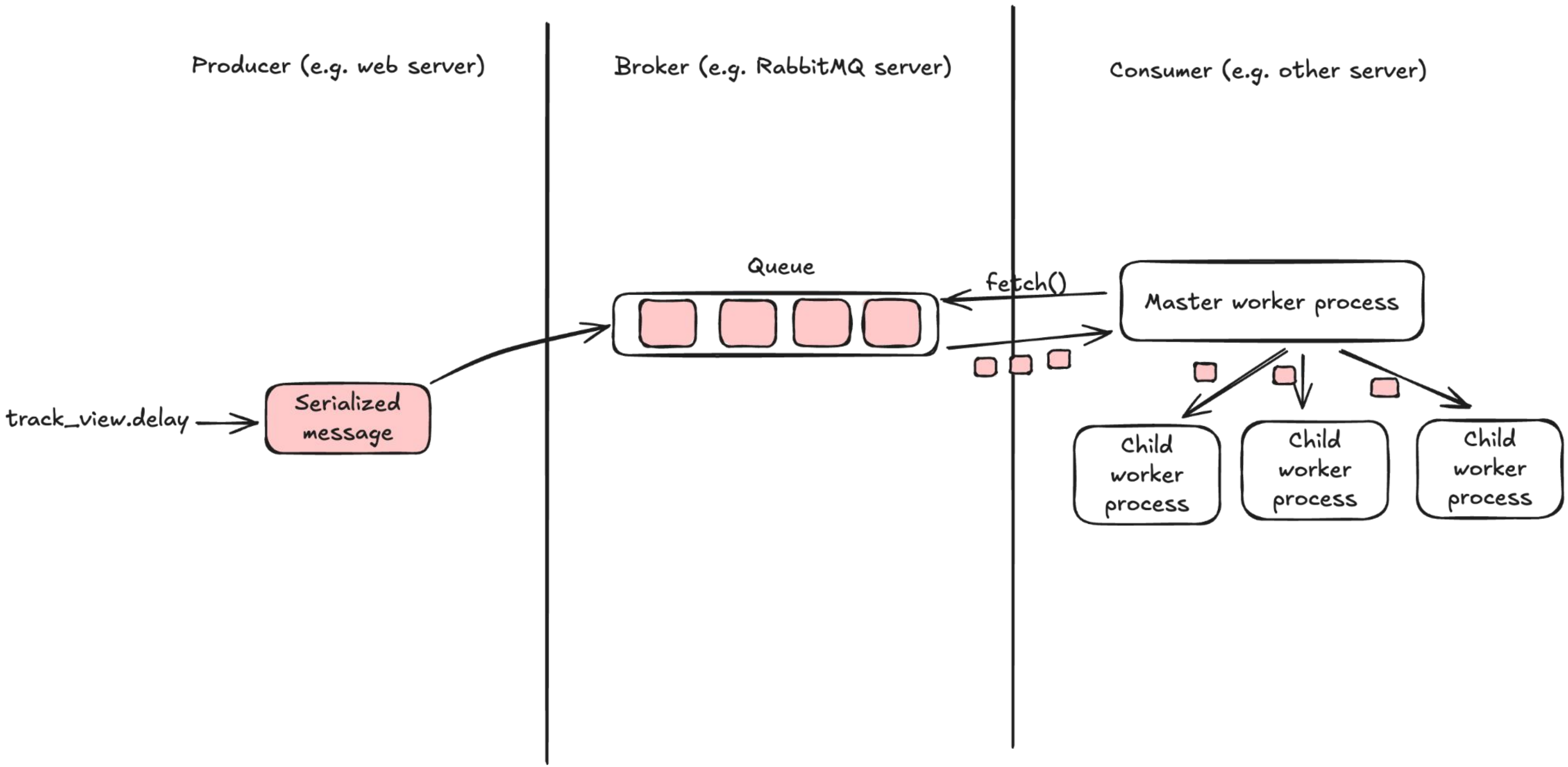
Taskmaster

Solving Deployment Headaches Caused By Long-Running Celery Jobs

Tomislav Maričević | https://tmarice.dev | @tmrcv | github.com/tmarice

Agenda

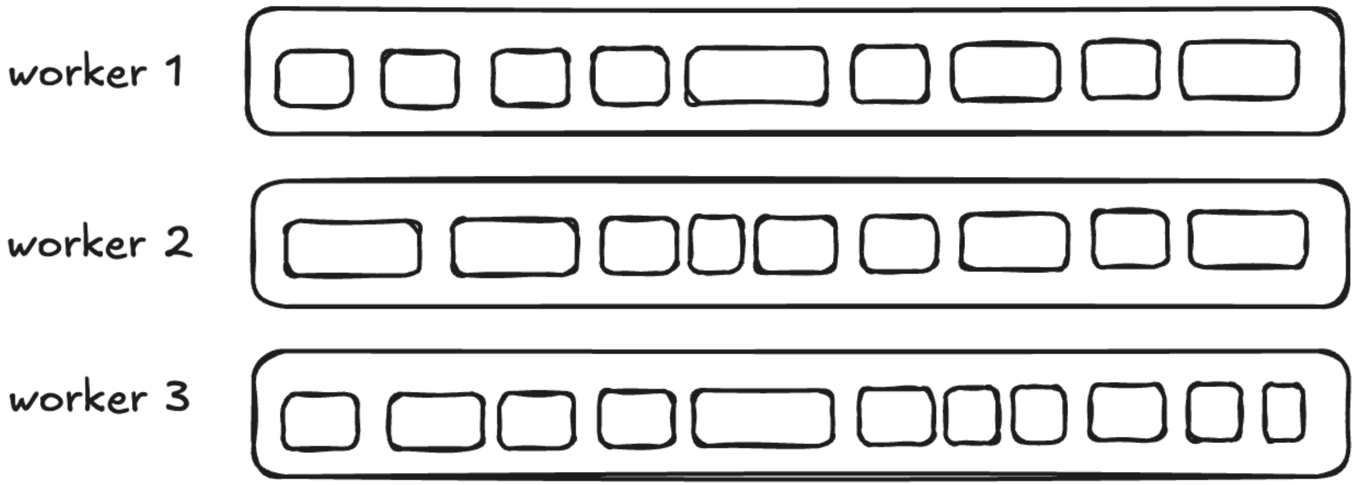
```python
# views.py

from django.views import View
from .tasks import track_view


class VideoTrackingView(View):
    ...

    def get(self, request):
        ...
        track_view.delay(user_id=request.user.id, video_id=request.GET['video_id'])


# tasks.py

from celery import shared_task
from .models import VideoView


@shared_task
def track_view(user_id, video_id):
    VideoView.objects.create(video_id=video_id, user_id=user_id)
```


Producer (e.g. web server)
Broker (e.g. RabbitMQ server)
Consumer (e.g. other server)
Queue
fetch()
Master worker process
track_view.delay
Serialized message
Child worker process
Child worker process
Child worker process

The Problem

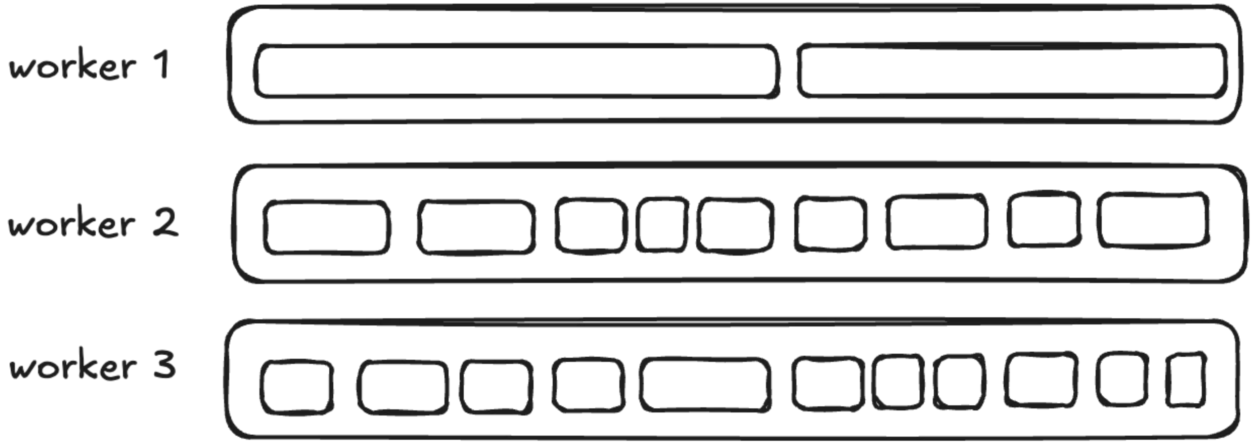
- Stack:
  - Django
  - Postgres
  - Celery
  - RabbitMQ
  - AWS EC2
  - Supervisor
- Celery
  - Prefetch multiplier = 4x
  - Late acknowledgment = False
  - No task timeouts

The Problem - The Simple Life



```
ssh server
sudo supervisorctl stop all
git pull
sudo supervisorctl start all
```

The Problem - Life Gets Complicated

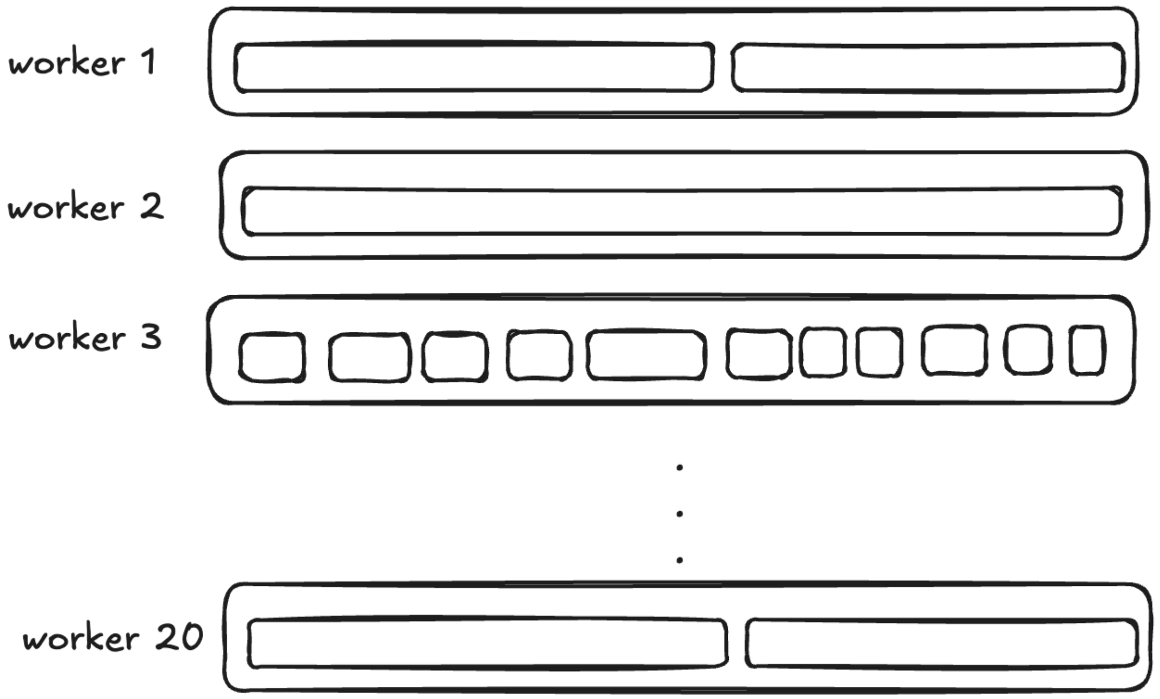


```
ssh server
celery inspect active
sudo supervisorctl stop all
git pull
sudo supervisorctl start all
```

Supervisor

- stopwaitsecs = 600
- killasgroup = true

The Problem - Life's No Fun Anymore



The Problem - Recap

- Many Celery workers on multiple servers
- Task loss is not acceptable
- Long tasks run often
- Deployment is stressful
  - No DB migrations → you need to know which workers have to be restarted
  - DB migrations → try to find known times in day when chance of long tasks is low
  - Even with celery inspect active, there’s a small possibility a task starts right after
  - Explaining the process to new hires is not possible (because there is no process)
  - Time from PR approval to deploy is very high - if you miss your chance, try again tomorrow
  - Deployment errors happen often

Goals

- The deployment process can be manual, but very straightforward.
- Multiple developers should be able to deploy every day.
- Deployment shouldn’t be (very) stressful.
- We should not drastically reduce development speed or code readability.
- We should not end up maintaining an overly complex tech stack.

Alternative 1:
Remove Cold Shutdown

Pros

- No changes in Python code
- No loss of work during deployment

Cons

- Supervisor doesn’t support this
- Migrations are problematic

Alternative 2:
Organize Tasks With a Daily/Weekly Deployment Window

Pros

- No code changes required

Cons

- Not scalable as the number of tasks grows
- Limits the number of daily deploys
- Might be harder to roll back changes or do follow-up deployments

Alternative 3:
Make All Workers Stoppable In Reasonable Time

Pros

- Very clear deployment process
- Eliminates migration problems

Cons

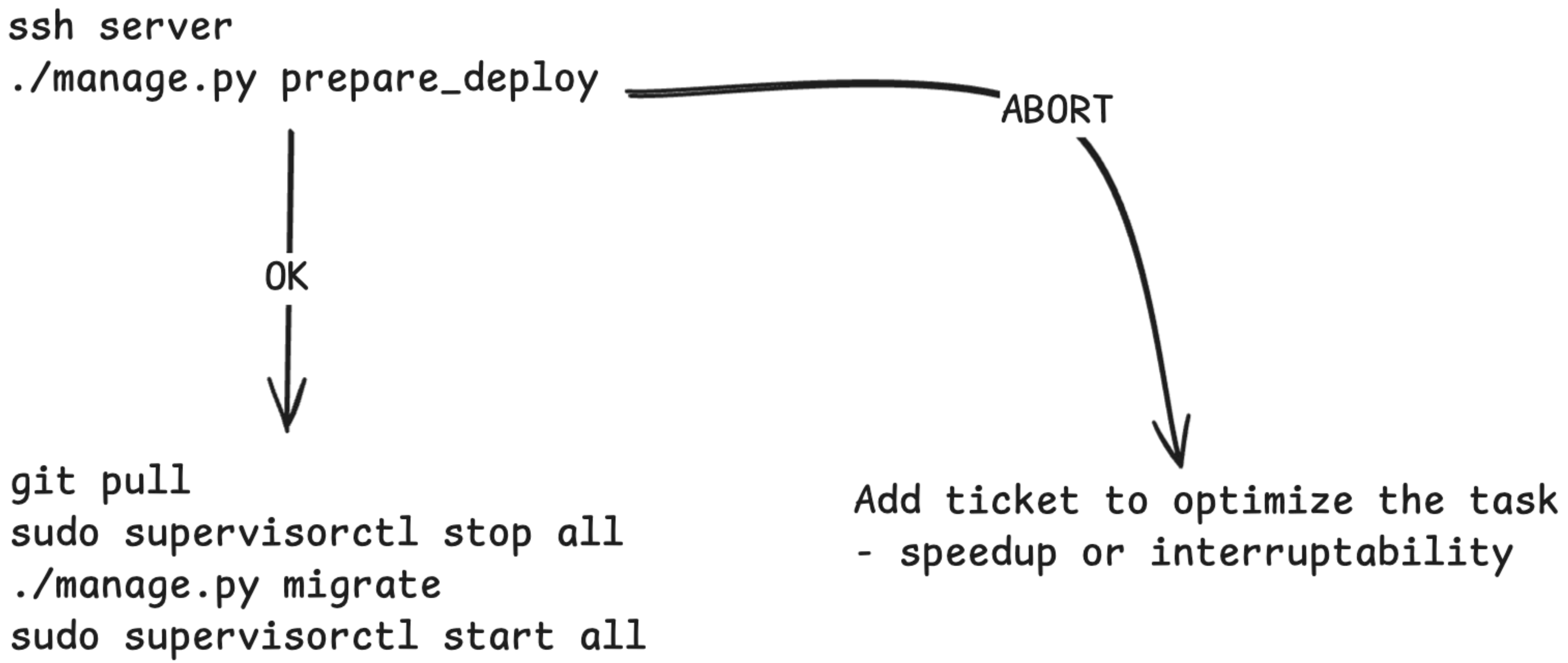
- Work will be lost after worker terminates
- Would have to refactor a large number of tasks

Solution - All Workers Stoppable In Reasonable Time

All tasks are either:

- Uninterruptible and very short
  - < 1 min duration
  - Anything with 3rd party API access - e.g. email sending
- Interruptible
  - We have to be able to interrupt the task at ANY point and be able to simply re-run it without anything bad happening
  - Considerations: DB, ES, Redis, other Celery tasks!
  - acks_late=True

Chosen Solution



```
ssh server
./manage.py prepare_deploy
```

```
git pull
sudo supervisorctl stop all
./manage.py migrate
sudo supervisorctl start all
```

Add ticket to optimize the task
- speedup or interruptability

Chosen Solution - Deployment Script

Utilizes Celery control interface

https://docs.celeryq.dev/en/stable/reference/celery.app.control.html

1. Instruct all Celery workers to stop fetching new tasks
2. Every 20s, check if all worker are idle or are they running only tasks marked with acks_late=True
3. If 10 minutes pass without this condition being true
   a. Instruct workers to continue consuming
   b. Output the list of non-acks_late tasks still running after 10 minutes
   c. Tell dev to abort deployment
4. If condition is true, tell dev to proceed

The Good, The Bad

Pros

- Deployments are stress-free
- Allows gradual improvement

Cons

- Someone needs to be the tasks police
- A lot of older code needs to be refactored
- Hard to reason about interruptibility if tasks delay other tasks

Task Guidelines

Short

- Optimize ORM queries
- Convert into a chain of shorter tasks

Interruptible

- acks_late=True
- Gather data in Python, persist to DB at the end – short transaction
- Avoid delaying other tasks
- For really long tasks, cache intermediate results – durable execution

Celery alternatives with durable execution

- https://temporal.io/
- https://www.dbos.dev/
- https://hatchet.run/

- https://github.com/RealOrangeOne/django-tasks

Thank you!

Q & A!

Tomislav Maričević | https://tmarice.dev | @tmrcv | github.com/tmarice