Celery Python task queue

CONCEPTS

The caller only publishes a message; a separate worker process runs the task. No worker, no work.

Your app task.delay()
↓ publish
Broker RabbitMQ or Redis
↓ consume
Worker runs the task
↓ store
Result backend optional, for .get()

WORKER

celery -A proj worker -l INFO
Start a worker

-c *<n>*
Number of pool processes

-Q *<q1>,<q2>*
Consume only these queues

DEFINE AND CALL

@app.task
Declare a task

@app.task(bind=True)
Get self for retry and request

add.delay(2, 3)
Queue with plain arguments

add.apply_async((2, 3), countdown=10)
Queue with options

apply_async(queue="*<q>*")
Send to a named queue

add.s(2, 3)
Signature: a task call as data

res.get(timeout=*<s>*)
Block for the result

res.ready() / res.state
Check for completion

CANVAS

chain(a.s(), b.s())
Pipe each result into the next

a.s() | b.s() | c.s()
Chain with the pipe operator

CANVAS (cont.)

group(a.s(i) for i in r)
Run many tasks in parallel

chord(group)(callback.s())
Run a callback after the group

A chain passes each return value as the first argument of the next task; use .si() to ignore it.

RETRIES AND ERRORS

self.retry(exc=e, countdown=*<s>*)
Retry the task later

autoretry_for=(*<Exc>*,)
Retry automatically on these errors

max_retries=*<n>*
Give up after n retries

acks_late=True
Acknowledge after the task finishes

ROUTING AND CONFIG

task_routes = {"*<task>*": {"queue": "*<q>*"}}
Route tasks to queues

broker_url = "redis://localhost:6379/0"
Set the broker

result_backend = "redis://..."
Set the result backend

PERIODIC TASKS

celery -A proj beat -l INFO
Start the scheduler

beat_schedule = {...}
Declare the schedule

crontab(hour=7, minute=30)
Cron-style schedule

Run exactly one beat per schedule, or every task fires multiple times.

INSPECT AND CONTROL

celery -A proj status
Workers that answer

celery -A proj inspect active
Tasks running now