# Celery Cheat Sheet

Task Queue · Broker · Beat · Canvas · Retry

worksheetdownload.com

## SETUP & BROKER

```
pip install celery redis
# celery.py
from celery import Celery
app = Celery('myapp',
broker='redis://localhost:6379/0',
backend='redis://localhost:6379/1')
celery -A myapp worker -l info
```

## DEFINING TASKS

```
@app.task(bind=True, max_retries=3)
def send_email(self, to_addr, subject):
try:
mailer.send(to_addr, subject)
except Exception as exc:
raise self.retry(exc=exc, countdown=60)
# Simple task:
@app.task
def add(x, y): return x + y
```

## CALLING TASKS

```
add.delay(4, 4)          # fire and
forget
add.apply_async((4,4), countdown=30) #
delayed
add.apply_async((4,4),
eta=datetime.utcnow()+timedelta(hours=1))
result = add.delay(4,4)
result.get(timeout=10)   # blocks until
done
result.status  # PENDING / STARTED /
SUCCESS / FAILURE
```

## RESULTS & STATE

```
res = add.apply_async((2,3))
res.id             # task UUID
res.ready()        # True if done
res.successful()   # True if SUCCESS
res.result         # return value
res.traceback      # if FAILURE
AsyncResult('uuid').get()  # by task ID
```

## PERIODIC TASKS (BEAT)

```
from celery.schedules import crontab
app.conf.beat_schedule = {
'daily-report': {
'task': 'tasks.send_report',
'schedule': crontab(hour=8, minute=0),
},
'every-30s': {
'task': 'tasks.heartbeat',
'schedule': 30.0,
},
}
celery -A myapp beat -l info
```

## RETRIES

```
@app.task(bind=True,
autoretry_for=(Exception,),
retry_kwargs={'max_retries': 5},
retry_backoff=True,
retry_backoff_max=700,
retry_jitter=True)
def fetch_data(self, url): ...
self.retry(exc=exc,
countdown=2**self.request.retries)
```

## CANVAS: CHAIN & GROUP

```
from celery import chain, group, chord
# Chain (sequential):
result = chain(task1.s(arg), task2.s(),
task3.s())()
# Group (parallel):
job = group(add.s(i, i) for i in
range(10))
result = job.apply_async()
# Chord (parallel then callback):
chord(group(tasks))(callback.s())
```

## ROUTING & QUEUES

```
@app.task(queue='high_priority')
def urgent_task(): ...
app.conf.task_routes = {
'tasks.urgent_*': {'queue':
'high_priority'},
'tasks.email_*': {'queue': 'email'},
}
celery worker -Q high_priority,email
```

## CONFIGURATION

```
app.conf.update(
task_serializer='json',
accept_content=['json'],
result_expires=3600,
task_time_limit=300,      # hard time
limit
task_soft_time_limit=240, # soft (raises
SoftTimeLimitExceeded)
worker_max_tasks_per_child=1000,
)
```

## SIGNALS

```
from celery.signals import task_success,
task_failure
@task_success.connect
def on_success(result, **kwargs):
metrics.incr('task.success')
@task_failure.connect
def on_failure(exception, **kwargs):
alert.send(str(exception))
```

Signals: prerun, postrun, retry, revoke

## COMMON GOTCHAS

Don't pass ORM objects to tasks pass IDs instead
result.get() inside another task causes deadlock
Pickle serializer security risk  use JSON
Task imports must be accessible in worker environment
Beat scheduler needs single instance (use RedBeat for HA)