# Redis and Celery task durability

## Redis HA and durability

**1. Using Redis Cluster URL Without Node Info**

When you provide just a Redis **Cluster URL** (e.g., redis://my-redis-cluster:6379), and it works, it usually means:

* You’re using **a service name** (e.g., via Kubernetes/OpenShift DNS like my-redis-cluster) that resolves to a **load balancer or a set of pods**.
* Redis **internally redirects** clients to the appropriate cluster nodes for key management.

However:

🔥 **Celery by default does not support Redis Cluster fully.**  
Redis Cluster lacks some key queue-related commands like KEYS, BRPOP, which Celery uses by default.

So if you're not using [celery-redis-cluster](https://github.com/celery/celery-redis-cluster), you're potentially in **undefined/unsupported** behavior.

**✅ Option A: Safe Usage with celery-redis-cluster Without Node Info**

You **can still use** celery-redis-cluster with just the cluster DNS name if it's round-robin or load-balanced.

Here’s how:

bash

CopyEdit

pip install celery-redis-cluster

Then in celery\_app.py:

python

CopyEdit

from celery import Celery

celery\_app = Celery("myapp")

celery\_app.conf.update(

broker\_transport="redis\_cluster",

broker\_url="redis://my-redis-cluster:6379", # Kubernetes/OpenShift DNS

broker\_transport\_options={

"cluster": {

"startup\_nodes": [

{"host": "my-redis-cluster", "port": "6379"} # cluster URL here

],

"read\_from\_replicas": False,

"max\_connections": 100

}

},

result\_backend="redis://my-redis-cluster:6379/0",

task\_default\_delivery\_mode="persistent",

task\_acks\_late=True,

task\_reject\_on\_worker\_lost=True,

broker\_connection\_retry\_on\_startup=True,

broker\_heartbeat=30,

worker\_max\_tasks\_per\_child=100,

)

This is **safe** if your cluster exposes at least **one master node** reliably via the cluster URL.

**Redis Cluster with Replication**

* **Ensure every master has at least one replica**:

bash

CopyEdit

redis-cli -c -h <node-host> -p <port> cluster info

redis-cli -c -h <node-host> -p <port> cluster nodes

Look for master and slave role entries — each master should have one or more connected replicas.

* Set these in Redis config (redis.conf) to enable automatic failover:

conf

CopyEdit

cluster-enabled yes

cluster-node-timeout 5000

appendonly yes # Enables AOF for durability

* **AOF (Append-Only File)** helps **persist writes to disk**, ensuring data durability even after pod restarts.
* Set the AOF fsync policy to balance safety and performance:

conf

CopyEdit

appendfsync everysec

* Celery doesn’t handle Redis failover — your Redis client must.
* celery-redis-cluster handles Redis Cluster node discovery and retry.

If you're not using celery-redis-cluster, **Celery will not failover between Redis masters** correctly in a Redis Cluster.

## Celery Durability

**Making Celery Handle Worker Crashes Gracefully**

Use these settings for **reliability** and **durability**:

| **Setting** | **Purpose** |
| --- | --- |
| task\_acks\_late=True | Ensures task is acknowledged *after* execution (not before) |
| task\_reject\_on\_worker\_lost=True | If worker crashes mid-task, task is re-queued |
| task\_default\_delivery\_mode='persistent' | Tasks are stored on disk in Redis if possible |
| worker\_max\_tasks\_per\_child=100 | Avoid memory leaks by restarting workers after N tasks |
| broker\_connection\_retry\_on\_startup=True | Retries connection on startup |
| broker\_transport\_options['visibility\_timeout'] | Requeue tasks if worker doesn’t ack in this time (used in crashes/timeouts). |
| retry\_on\_timeout | Retry if Redis times out when publishing/consuming tasks. |
| broker\_heartbeat=30 | Ensures connection liveness monitoring |

**What You Can’t Do Without Node Info**

If you don’t provide startup nodes and aren’t using celery-redis-cluster, Redis cluster client discovery may fail when:

* Nodes move around.
* The single entrypoint goes down.
* Load-balancer/DNS hides MOVED responses or lacks session stickiness.

So the safest solution is to **either:**

1. Use celery-redis-cluster with a single DNS name as the only startup\_node, or
2. Ask your infra team for **stable startup node info** (just 1-2 nodes is enough) and configure startup\_nodes.

**Summary: HA and Durable Celery with Cluster URL**

| **Aspect** | **Solution** |
| --- | --- |
| Redis Cluster | Use celery-redis-cluster transport |
| No node info | Use the DNS name (my-redis-cluster) in startup\_nodes |
| Worker crash handling | Use task\_acks\_late, task\_reject\_on\_worker\_lost, and retries |
| Durable tasks | Use task\_default\_delivery\_mode='persistent' |
| Redis persistence | Enable AOF (appendonly yes) in Redis |