



This document describes the current stable version of Celery (4.2). For development docs, [go here](#).

Debugging

Debugging Tasks Remotely (using pdb)

Star 11,909

Please help support this community project with a donation:



Previous topic

[Optimizing](#)

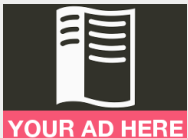
Next topic

[Concurrency](#)

This Page

[Show Source](#)

Quick search



Reach over 7 million devs each month when you advertise with Read the Docs.

Sponsored · Ads served ethically

Basics

celery.contrib.rdb is an extended version of **pdb** that enables remote debugging of processes that doesn't have terminal access.

Example usage:

```
from celery import task
from celery.contrib import rdb

@task()
def add(x, y):
    result = x + y
    rdb.set_trace() # <- set break-point
    return result
```

set_trace() sets a break-point at the current location and creates a socket you can telnet into to remotely debug your task.

The debugger may be started by multiple processes at the same time, so rather than using a fixed port the debugger will search for an available port, starting from the base port (6900 by default). The base port can be changed using the environment variable **CELERY_RDB_PORT**.

By default the debugger will only be available from the local host, to enable access from the outside you have to set the environment variable **CELERY_RDB_HOST**.

When the worker encounters your break-point it'll log the following information:

```
[INFO/MainProcess] Received task:
tasks.add[d7261c71-4962-47e5-b342-2448bedd20e8]
[WARNING/PoolWorker-1] Remote Debugger:6900:
Please telnet 127.0.0.1 6900. Type `exit` in session to continue.
[2011-01-18 14:25:44,119: WARNING/PoolWorker-1] Remote Debugger:6900:
Waiting for client...
```

If you telnet the port specified you'll be presented with a **pdb** shell:

```
$ telnet localhost 6900
Connected to localhost.
Escape character is '^]'.
> /opt/dev/demoapp/tasks.py(128)add()
-> return result
(Pdb)
```

Enter **help** to get a list of available commands, It may be a good idea to read the [Python Debugger Manual](#) if you have never used **pdb** before.

To demonstrate, we'll read the value of the **result** variable, change it and continue execution of the task:

```
(Pdb) result
4
(Pdb) result = 'hello from rdb'
(Pdb) continue
Connection closed by foreign host.
```

The result of our vandalism can be seen in the worker logs:

```
[2011-01-18 14:35:36,599: INFO/MainProcess] Task
tasks.add[d7261c71-4962-47e5-b342-2448bedd20e8] succeeded
in 61.481s: 'hello from rdb'
```

Tips

Enabling the break-point signal

If the environment variable **CELERY_RDBSIG** is set, the worker will open up an rdb instance whenever the *SIGUSR2* signal is sent. This is the case for both main and worker processes.

For example starting the worker with:

```
$ CELERY_RDBSIG=1 celery worker -l info
```

You can start an rdb session for any of the worker processes by executing:

```
$ kill -USR2 <pid>
```