

Group 94:

# Report for Assignment 1

## Project chosen

Name: Celery

URL: <https://github.com/iwtwm/celery>

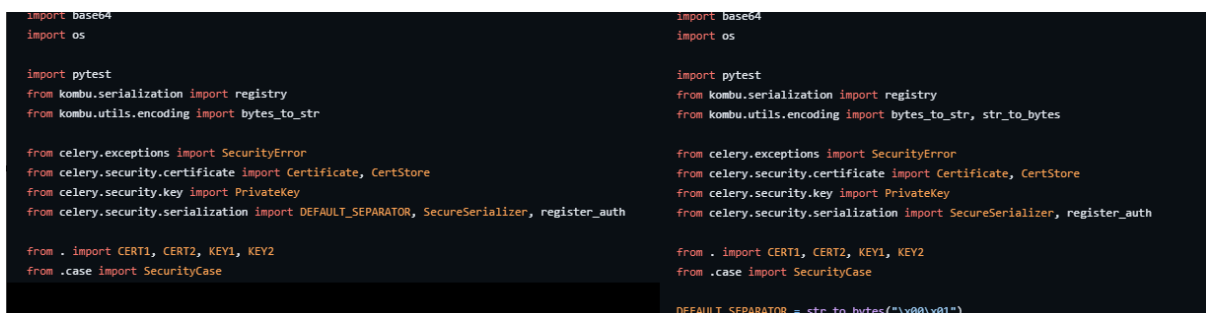
Number of lines of code and the tool used to count it: total 63856 nloc using Lizard and "lizard -l python" in command line

Programming language: Python

## Coverage measurement

### Existing tool

The coverage measurement was done using coverage.py, and the tests were based on the pytest framework and ran using pytest and the plugins pytest-cov. The tests required multiple modules to be installed like celery and pytest-celery among others. Despite installing all of the required modules one of the tests had an import error that would result in a faulty coverage run results so we changed two lines as a quick fix to solve the error. The file in question was test\_serialization.py.



```
import base64
import os

import pytest
from kombu.serialization import registry
from kombu.utils.encoding import bytes_to_str

from celery.exceptions import SecurityError
from celery.security.certificate import Certificate, CertStore
from celery.security.key import PrivateKey
from celery.security.serialization import DEFAULT_SEPARATOR, SecureSerializer, register_auth

from . import CERT1, CERT2, KEY1, KEY2
from .case import SecurityCase

import base64
import os

import pytest
from kombu.serialization import registry
from kombu.utils.encoding import bytes_to_str, str_to_bytes

from celery.exceptions import SecurityError
from celery.security.certificate import Certificate, CertStore
from celery.security.key import PrivateKey
from celery.security.serialization import SecureSerializer, register_auth

from . import CERT1, CERT2, KEY1, KEY2
from .case import SecurityCase

DEFAULT_SEPARATOR = str_to_bytes("\x00\x01")
```

Changes before and after in test\_serialization.py

To run the measurement of the code coverage of the program the following command-line command is used:

```
coverage run --rcfile=covconfig.coveragerc -m pytest
```

The covconfig.coveragerc had the configuration of the coverage run:

```
1 [run]
2 branch = True
3 omit = t
4 source = .
5
6 [report]
7 show_missing = True
8
9 [html]
10 directory = coverage_html
```

The config file sets the source, and indicates that the branch coverage should also be tested and that the missing lines should be shown.

The next following command makes coverage.py generate a html report of the code coverage results in the htmlcov folder:

```
coverage html -i
```

Coverage report: 84%

Files Functions Classes

coverage.py v7.5.3, created at 2024-06-14 13:00 +0200

filter

hide covered

| File                             | function                        | statements | missing | excluded | branches | partial | coverage |
|----------------------------------|---------------------------------|------------|---------|----------|----------|---------|----------|
| celery/_init_.py                 | debug_import                    | 0          | 0       | 4        | 0        | 0       | 100%     |
| celery/state.py                  | get_current_app                 | 0          | 0       | 2        | 0        | 0       | 100%     |
| celery/state.py                  | _app_or_default_trace           | 0          | 0       | 16       | 0        | 0       | 100%     |
| celery/app/base.py               | Celery.on_init                  | 0          | 0       | 0        | 0        | 0       | 100%     |
| celery/app/defaults.py           | find_deprecated_settings        | 0          | 0       | 0        | 0        | 0       | 100%     |
| celery/app/task.py               | Task.on_bound                   | 0          | 0       | 0        | 0        | 0       | 100%     |
| celery/app/task.py               | Task.shadow_name                | 0          | 0       | 0        | 0        | 0       | 100%     |
| celery/app/task.py               | Task.before_start               | 0          | 0       | 0        | 0        | 0       | 100%     |
| celery/app/task.py               | Task.on_success                 | 0          | 0       | 0        | 0        | 0       | 100%     |
| celery/app/task.py               | Task.on_retry                   | 0          | 0       | 0        | 0        | 0       | 100%     |
| celery/app/task.py               | Task.on_failure                 | 0          | 0       | 0        | 0        | 0       | 100%     |
| celery/app/task.py               | Task.after_return               | 0          | 0       | 0        | 0        | 0       | 100%     |
| celery/apps/_init_.py            | (no function)                   | 0          | 0       | 0        | 0        | 0       | 100%     |
| celery/apps/worker.py            | install_rdb_handler             | 0          | 0       | 3        | 0        | 0       | 100%     |
| celery/apps/worker.py            | install_rdb_handler.rdb_handler | 0          | 0       | 7        | 0        | 0       | 100%     |
| celery/backends/_init_.py        | (no function)                   | 0          | 0       | 0        | 0        | 0       | 100%     |
| celery/backends/base.py          | Backend.cleanup                 | 0          | 0       | 0        | 0        | 0       | 100%     |
| celery/backends/base.py          | Backend.process_cleanup         | 0          | 0       | 0        | 0        | 0       | 100%     |
| celery/bin/_init_.py             | (no function)                   | 0          | 0       | 0        | 0        | 0       | 100%     |
| celery/bin/list.py               | list                            | 0          | 0       | 0        | 0        | 0       | 100%     |
| celery/bin/logtool.py            | logtool                         | 0          | 0       | 0        | 0        | 0       | 100%     |
| celery/bin/shell.py              | _ipython_pre_10                 | 0          | 0       | 5        | 0        | 0       | 100%     |
| celery/bin/shell.py              | _ipython_terminal               | 0          | 0       | 2        | 0        | 0       | 100%     |
| celery/bin/shell.py              | _ipython_D10                    | 0          | 0       | 2        | 0        | 0       | 100%     |
| celery/bin/shell.py              | _no_ipython                     | 0          | 0       | 1        | 0        | 0       | 100%     |
| celery/bin/upgrade.py            | upgrade                         | 0          | 0       | 0        | 0        | 0       | 100%     |
| celery/bootsteps.py              | Step.create                     | 0          | 0       | 0        | 0        | 0       | 100%     |
| celery/canvas.py                 | StampingVisitor.on_signature    | 0          | 0       | 0        | 0        | 0       | 100%     |
| celery/concurrency/asyncpool.py  | AsyncPool.on_shrink             | 0          | 0       | 0        | 0        | 0       | 100%     |
| celery/contrib/_init_.py         | (no function)                   | 0          | 0       | 0        | 0        | 0       | 100%     |
| celery/contrib/django/_init_.py  | (no function)                   | 0          | 0       | 0        | 0        | 0       | 100%     |
| celery/contrib/testing/_init_.py | (no function)                   | 0          | 0       | 0        | 0        | 0       | 100%     |
| celery/events/cursesmon.py       | CursesMonitor._init_            | 0          | 0       | 14       | 0        | 0       | 100%     |
| celery/events/cursesmon.py       | CursesMonitor.format_row        | 0          | 0       | 25       | 0        | 0       | 100%     |
| celery/events/cursesmon.py       | CursesMonitor.screen_width      | 0          | 0       | 2        | 0        | 0       | 100%     |

Coverage results after opening index.html in the generated htmlcov folder

### Your own coverage tool

Rose Hollander

Function 1: celery/app/amqp.py      AMQP.\_handle\_conf\_update

```

_handle_conf_update() in amqp.py
1 - """Sending/Receiving Messages (Kombu integration)."""
2 - import numbers
3 - from collections import namedtuple
4 - from collections.abc import Mapping
5 - from datetime import timedelta
6 - from weakref import WeakValueDictionary
7 -
8 - from kombu import Connection, Consumer, Exchange, Producer, Queue, pools
9 - from kombu.common import Broadcast
10 - from kombu.utils.functional import maybe_list
11 - from kombu.utils.objects import cached_property
12 -
13 - from celery import signals
14 - from celery.utils.nodenames import anon_nodename
15 - from celery.utils.saferepr import saferepr
16 - from celery.utils.text import indent as textindent
17 - from celery.utils.time import maybe_make_aware
18 -
19 - from . import routes as _routes
20 -
21 - __all__ = ('AMQP', 'Queues', 'task_message')
22 -
611 - def _handle_conf_update(self, *args, **kwargs):
612 -     if ('task_routes' in kwargs or 'task_routes' in args):
613 -         self.flush_routes()
614 -         self.router = self.Router()
615 -     return

```

```

2 + import numbers
3 + from collections import namedtuple
4 + from collections.abc import Mapping
5 + from datetime import timedelta
6 + from weakref import WeakValueDictionary
7 +
8 + from kombu import Connection, Consumer, Exchange, Producer, Queue, pools
9 + from kombu.common import Broadcast
10 + from kombu.utils.functional import maybe_list
11 + from kombu.utils.objects import cached_property
12 +
13 + from celery import signals
14 + from celery.utils.nodenames import anon_nodename
15 + from celery.utils.saferepr import saferepr
16 + from celery.utils.text import indent as textindent
17 + from celery.utils.time import maybe_make_aware
18 +
19 + from . import routes as _routes
20 +
21 + from branch_dictionary import branch_coverage
22 +
23 + __all__ = ('AMQP', 'Queues', 'task_message')
614 + def _handle_conf_update(self, *args, **kwargs):
615 +     if ('task_routes' in kwargs or 'task_routes' in args):
616 +         branch_coverage["_handle_conf_update1"] = True
617 +         self.flush_routes()
618 +         self.router = self.Router()
619 +     else:
620 +         branch_coverage["_handle_conf_update2"] = True
621 +     return

```

```

-- laptop: /Documents/uni/sep/celery python3 branch_tracker.py
===== test session starts =====
platform linux -- Python 3.10.12, pytest-8.2.2, pluggy-1.5.0
rootdir: /Documents/uni/sep/celery
configfile: pyproject.toml
testpaths: t/unit/
plugins: docker-tools-3.1.3, mock-3.14.0, subtests-0.12.1, anyio-4.4.0, cov-5.0.0, flaky-3.8.1, celery-1.0.0
collected 3350 items

t/unit/app/test_amqp.py ..... [ 13]
t/unit/app/test_annotations.py ..... [ 13]
t/unit/app/test_app.py ..... [ 48]
t/unit/app/test_backends.py ..... [ 58]
t/unit/app/test_beat.py ..... [ 78]
t/unit/app/test_builtins.py ..... [ 78]
t/unit/app/test_celery.py ..... [ 78]
t/unit/app/test_control_cli.py ..... [ 98]
t/unit/app/test_defaults.py ..... [ 98]
t/unit/app/test_exceptions.py ..... [ 98]
t/unit/app/test_loaders.py ..... [108]
t/unit/app/test_log.py ..... [128]
t/unit/app/test_preload_cli.py EE [128]
t/unit/app/test_registry.py ..... [128]
t/unit/app/test_routes.py ..... [128]
t/unit/app/test_schedules.py ..... [168]

-- Docs: https://docs.pytest.org/en/stable/how-to/capture-warnings.html
===== short test summary info =====
ERROR t/unit/app/test_preload_cli.py::test_preload_options[subcommand_with_params0]
ERROR t/unit/app/test_preload_cli.py::test_preload_options[subcommand_with_params1]
ERROR t/unit/bin/test_beat.py::test_cli
ERROR t/unit/bin/test_beat.py::test_cli quiet
ERROR t/unit/bin/test_control.py::test_custom_remote_command[inspect-custom_cmd0]
ERROR t/unit/bin/test_control.py::test_custom_remote_command[control-custom_cmd1]
ERROR t/unit/bin/test_control.py::test_unrecognized_remote_command[inspect-this_command_does_not_exist]
ERROR t/unit/bin/test_control.py::test_unrecognized_remote_command[control-this_command_does_not_exist]
ERROR t/unit/bin/test_control.py::test_unrecognized_remote_command[inspect-custom_control_cmd]
ERROR t/unit/bin/test_control.py::test_unrecognized_remote_command[control-custom_inspect_cmd]
ERROR t/unit/bin/test_control.py::test_listing_remote_commands[inspect-\n custom_inspect_cmd x\\s+Ask the workers to reply with x\\s+\n]
ERROR t/unit/bin/test_control.py::test_listing_remote_commands[control-\n custom_control_cmd a b\\s+Ask the workers to reply with a and b\\s+\n]
ERROR t/unit/bin/test_daemonization.py::test_daemon_options_from_config[worker]
ERROR t/unit/bin/test_daemonization.py::test_daemon_options_from_config[beat]
ERROR t/unit/bin/test_daemonization.py::test_daemon_options_from_config[events]
ERROR t/unit/bin/test_worker.py::test_cli
ERROR t/unit/bin/test_worker.py::test_cli skip_checks
===== 3322 passed, 8 skipped, 3 xfailed, 58 warnings, 17 errors, 28910 subtests passed in 117.18s (0:01:57) =====

BRANCH COVERAGE:

Branch Hits:

_handle_conf_update1 is covered: False
_handle_conf_update2 is covered: False
worker_main1 is covered: False
worker_main2 is covered: True
worker_main3 is covered: False
worker_main4 is covered: True

Branch Coverage Percentage:

_handle_conf_update: 0 out of 2 branches covered (0.00%)
worker_main: 2 out of 4 branches covered (50.00%)

```

Function 2: celery/app/base.py

Celery.worker\_main

```

worker_main() in base.py
1  """Actual App instance implementation"""
2  +
3  + import inspect
4  + import os
5  + import sys
6  + import threading
7  + import warnings
8  + from collections import UserDict, defaultdict, deque
9  + from datetime import datetime
10 + from datetime import timezone as datetime_timezone
11 + from operator import attrgetter
12 +
13 + from click.exceptions import Exit
14 + from dateutil.parser import isoparse
15 + from kombu import pools
16 + from kombu.clocks import ImportClock
17 + from kombu.common import eid_from
18 + from kombu.utils.compat import register_after_fork
19 + from kombu.utils.objects import cached_property
20 + from kombu.utils.uid import uuid
21 + from vine import starprmise
22 +
23 + from celery import platforms, signals
24 + from celery.state import _announce_app_finalized, _deregister_app, _register_app, _set_current_app, _task_stack,
25 +   connect_on_app_finalize, get_current_app, get_current_worker_task, set_default_app
26 + from celery.exceptions import AlreadyRegistered, ImproperlyConfigured
27 + from celery.loaders import get_loader_cls
28 + from celery.local import PromiseProxy, maybe_evaluate
29 + from celery.utils import abstract
30 + from celery.utils.collections import AttributeDictMixin
31 + from celery.utils.dispatch import Signal
32 + from celery.utils.functional import first, head_from_fun, maybe_list
33 + from celery.utils.imports import get_task_name, instantiate, symbol_by_name
34 + from celery.utils.log import get_logger
35 + from celery.utils.objects import FallBackContext, wra_lookup
36 + from celery.utils.time import maybe_make_aware, timezone, to_utc
37 +
38 + # Load all builtin tasks
39 + from . import backends, builtins # noqa
40 + from .annotations import prepare as prepare_annotations
41 + from .autoretry import add_autoretry_behaviour
42 + from .defaults import DEFAULT_SECURITY_DIGEST, find_deprecated_settings
43 + from .registry import TaskRegistry
44 + from .utils import (AppPicker, Settings, new_key_to_old, old_key_to_new, unpickle_app, unpickle_app_v2, appstr,
45 +   bugreport, detect_settings)
46 +
47 + all = {'(Celery)'}

```

```

381 -
382 - def worker_main(self, argv=None):
383 -     """Run :program:`celery worker` using 'argv'."""
384 -     Uses :data:`sys.argv` if 'argv' is not specified.
385 -     """
386 -     if argv is None:
387 -         argv = sys.argv
388 -
389 -     if 'worker' not in argv:
390 -         raise ValueError(
391 -             "The worker sub-command must be specified in argv.\n"
392 -             "Use 'app.start()' to programmatically start other commands."
393 -         )
394 -
395 -     self.start(argv=argv)
396 -
397 -

```

```

celery.py
1  """Actual App instance implementation"""
2  +
3  + import inspect
4  + import os
5  + import sys
6  + import threading
7  + import warnings
8  + from collections import UserDict, defaultdict, deque
9  + from datetime import datetime
10 + from datetime import timezone as datetime_timezone
11 + from operator import attrgetter
12 +
13 + from click.exceptions import Exit
14 + from dateutil.parser import isoparse
15 + from kombu import pools
16 + from kombu.clocks import ImportClock
17 + from kombu.common import eid_from
18 + from kombu.utils.compat import register_after_fork
19 + from kombu.utils.objects import cached_property
20 + from kombu.utils.uid import uuid
21 + from vine import starprmise
22 +
23 + from celery import platforms, signals
24 + from celery.state import _announce_app_finalized, _deregister_app, _register_app, _set_current_app, _task_stack,
25 +   connect_on_app_finalize, get_current_app, get_current_worker_task, set_default_app
26 + from celery.exceptions import AlreadyRegistered, ImproperlyConfigured
27 + from celery.loaders import get_loader_cls
28 + from celery.local import PromiseProxy, maybe_evaluate
29 + from celery.utils import abstract
30 + from celery.utils.collections import AttributeDictMixin
31 + from celery.utils.dispatch import Signal
32 + from celery.utils.functional import first, head_from_fun, maybe_list
33 + from celery.utils.imports import get_task_name, instantiate, symbol_by_name
34 + from celery.utils.log import get_logger
35 + from celery.utils.objects import FallBackContext, wra_lookup
36 + from celery.utils.time import maybe_make_aware, timezone, to_utc
37 +
38 + # Load all builtin tasks
39 + from . import backends, builtins # noqa
40 + from .annotations import prepare as prepare_annotations
41 + from .autoretry import add_autoretry_behaviour
42 + from .defaults import DEFAULT_SECURITY_DIGEST, find_deprecated_settings
43 + from .registry import TaskRegistry
44 + from .utils import (AppPicker, Settings, new_key_to_old, old_key_to_new, unpickle_app, unpickle_app_v2, appstr,
45 +   bugreport, detect_settings)
46 +
47 + all = {'(Celery)'}

```

```

385 + def worker_main(self, argv=None):
386 +     """Run :program:`celery worker` using 'argv'."""
387 +     Uses :data:`sys.argv` if 'argv' is not specified.
388 +     """
389 +     if argv is None:
390 +         argv = sys.argv
391 +
392 +     if 'worker' not in argv:
393 +         raise ValueError(
394 +             "The worker sub-command must be specified in argv.\n"
395 +             "Use 'app.start()' to programmatically start other commands."
396 +         )
397 +
398 +     self.start(argv=argv)
399 +
400 +

```

```

- laptop: /Documents/uni/sep/celerys python3 branch_tracker.py
===== test session starts =====
platform linux -- Python 3.10.12, pytest-8.2.2, pluggy-1.5.0
rootdir: /Documents/uni/sep/celery
configfile: pyproject.toml
testpaths: t/unit/
plugins: docker-tools-3.1.3, mock-3.14.0, subtests-0.12.1, anyio-4.4.0, cov-5.0.0, flaky-3.8.1, celery-1.0.0
collected 3350 items

t/unit/app/test_app.py ..... [ 1%]
t/unit/app/test_annotations.py ..... [ 1%]
t/unit/app/test_app.py ..... [ 4%]
t/unit/app/test_backends.py ..... [ 5%]
t/unit/app/test_beat.py ..... [ 7%]
t/unit/app/test_builtins.py ..... [ 7%]
t/unit/app/test_celery.py ..... [ 7%]
t/unit/app/test_control.py ..... [ 9%]
t/unit/app/test_defaults.py ..... [ 9%]
t/unit/app/test_exceptions.py ..... [ 9%]
t/unit/app/test_loaders.py ..... [ 10%]
t/unit/app/test_log.py ..... [ 12%]
t/unit/app/test_preload_cli.py EE [ 12%]
t/unit/app/test_registry.py ..... [ 12%]
t/unit/app/test_routes.py ..... [ 12%]
t/unit/app/test_schedules.py ..... [ 13%]

-- Docs: https://docs.pytest.org/en/stable/how-to/capture-warnings.html
===== short test summary info =====
ERROR t/unit/app/test_preload_cli.py::test_preload_options[subcommand_with_params0]
ERROR t/unit/app/test_preload_cli.py::test_preload_options[subcommand_with_params1]
ERROR t/unit/bin/test_beat.py::test_cli
ERROR t/unit/bin/test_beat.py::test_cli quiet
ERROR t/unit/bin/test_control.py::test_custom_remote_command[inspect-custom_cmd0]
ERROR t/unit/bin/test_control.py::test_custom_remote_command[control-custom_cmd1]
ERROR t/unit/bin/test_control.py::test_unrecognized_remote_command[inspect-this_command_does_not_exist]
ERROR t/unit/bin/test_control.py::test_unrecognized_remote_command[control-this_command_does_not_exist]
ERROR t/unit/bin/test_control.py::test_unrecognized_remote_command[inspect-custom_control_cmd]
ERROR t/unit/bin/test_control.py::test_unrecognized_remote_command[control-custom_inspect_cmd]
ERROR t/unit/bin/test_control.py::test_listing_remote_commands[inspect-\n custom_inspect_cmd x\\s+Ask the workers to reply with x\\.\n]
ERROR t/unit/bin/test_control.py::test_listing_remote_commands[control-\n custom_control_cmd a b\\s+Ask the workers to reply with a and b\\.\n]
ERROR t/unit/bin/test_daemonization.py::test_daemon_options_from_config[worker]
ERROR t/unit/bin/test_daemonization.py::test_daemon_options_from_config[beat]
ERROR t/unit/bin/test_daemonization.py::test_daemon_options_from_config[events]
ERROR t/unit/bin/test_worker.py::test_cli
ERROR t/unit/bin/test_worker.py::test_cli skip checks
===== 3322 passed, 8 skipped, 3 xfailed, 58 warnings, 17 errors, 28910 subtests passed in 117.18s (0:01:57) =====

BRANCH COVERAGE:

Branch Hits:
_handle_conf_update1 is covered: False
_handle_conf_update2 is covered: False
worker_main1 is covered: False
worker_main2 is covered: True
worker_main3 is covered: False
worker_main4 is covered: True

Branch Coverage Percentage:
_handle_conf_update: 0 out of 2 branches covered (0.00%)
worker_main: 2 out of 4 branches covered (50.00%)

```


celery / branch\_dictionary.py

 iwtwm completed branch coverage tracker

Code


Blame

14 lines (12 loc) · 448 Bytes

 Code 55% faster with GitHub Copilot

```
1  ∨  branch_coverage = {
2      "_handle_conf_update1": False,      # if 'task_routes' in kwargs or 'task_routes' in args
3      "_handle_conf_update2": False,      # else branch
4
5      "worker_main1": False,              # if argv is None
6      "worker_main2": False,              # else branch
7      "worker_main3": False,              # 'worker' not in argv
8      "worker_main4": False               # else branch
9  }
10
11  ∨  branch_totals = {
12      "_handle_conf_update": 2,
13      "worker_main": 4
14  }
```


celery / branch\_tracker.py

 iwtwm completed branch coverage tracker

Code

Blame

41 lines (33 loc) · 1.28 KB

 Code 55% faster with GitHub Copilot

```
1  import os
2  from branch_dictionary import branch_coverage, branch_totals
3
4  # Function to print branch coverage
5  def print_branch_coverage():
6      print("\nBRANCH COVERAGE:\n")
7
8      covered_branches = {
9          "_handle_conf_update": 0,
10         "worker_main": 0
11     }
12
13     print("Branch Hits:\n")
14     s = ""
15     for key, value in branch_coverage.items():
16         s += f"{key} is covered: {value}\n"
17     print(s)
18
19     for key, value in branch_coverage.items():
20         if value:
21             if key.startswith("worker_main"):
22                 covered_branches["worker_main"] += 1
23             elif key.startswith("_handle_conf_update"):
24                 covered_branches["_handle_conf_update"] += 1
25
26     print("Branch Coverage Percentage:\n")
27     for key, value in branch_totals.items():
28         covered = covered_branches[key]
29         coverage_percentage = (covered / value) * 100
30         print(f"{key}: {covered} out of {value} branches covered ({coverage_percentage:.2f}%)")
31
32     # Function to run all tests and print coverage
33     def run_coverage():
34         import pytest
35         pytest.main([os.path.join('t', 'unit', 'app', 'test_amqp.py')])
36         pytest.main([os.path.join('t', 'unit', 'app', 'test_app.py')])
37
38         print_branch_coverage()
39
40     if __name__ == '__main__':
41         run_coverage()
```

Buse Basavci

Function 1: celery/app/control.py    Inspect.\_prepare

✓ 1,579 celery/app/control.py

```
... @@ -1,779 +1,800 @@
1 - """Worker Remote Control Client.
2 -
3 - Client for worker remote control commands.
4 - Server implementation is in :mod:`celery.worker.control`.
5 - There are two types of remote control commands:
6 -
7 - * Inspect commands: Does not have side effects, will usually just return some value
8 - found in the worker, like the list of currently registered tasks, the list of active tasks, etc.
9 - Commands are accessible via :class:`Inspect` class.
10 -
11 - * Control commands: Performs side effects, like adding a new queue to consume from.
12 - Commands are accessible via :class:`Control` class.
13 - """
14 - import warnings
15 -
16 - from billiard.common import TERM_SIGNAME
17 - from kombu.matcher import match
18 - from kombu.pidbox import Mailbox
19 - from kombu.utils.compat import register_after_fork
20 - from kombu.utils.functional import lazy
21 - from kombu.utils.objects import cached_property
22 -
23 - from celery.exceptions import DuplicateNodeNameWarning
24 - from celery.utils.log import get_logger
25 - from celery.utils.text import pluralize
26 -
27 - __all__ = ('Inspect', 'Control', 'flatten_reply')
28 -
29 - logger = get_logger(__name__)
```

```
class Inspect:
    """API for inspecting workers.

    This class provides proxy for accessing Inspect API of workers. The API is
    defined in :py:mod:`celery.worker.control`
    """

    app = None

    def __init__(self, destination=None, timeout=1.0, callback=None,
                 connection=None, app=None, limit=None, pattern=None,
                 matcher=None):
        self.app = app or self.app
        self.destination = destination
        self.timeout = timeout
        self.callback = callback
        self.connection = connection
        self.limit = limit
        self.pattern = pattern
        self.matcher = matcher

    def _prepare(self, reply):
        if reply:
            by_node = flatten_reply(reply)
            if (self.destination and
                not isinstance(self.destination, (list, tuple))):
                return by_node.get(self.destination)
            if self.pattern:
                pattern = self.pattern
                matcher = self.matcher
                return {node: reply for node, reply in by_node.items()
                        if match(node, pattern, matcher)}
            return by_node
```

```
1 + """Worker Remote Control Client.
2 +
3 + Client for worker remote control commands.
4 + Server implementation is in :mod:`celery.worker.control`.
5 + There are two types of remote control commands:
6 +
7 + * Inspect commands: Does not have side effects, will usually just return some value
8 + found in the worker, like the list of currently registered tasks, the list of active tasks, etc.
9 + Commands are accessible via :class:`Inspect` class.
10 +
11 + * Control commands: Performs side effects, like adding a new queue to consume from.
12 + Commands are accessible via :class:`Control` class.
13 + """
14 + import warnings
15 +
16 + from billiard.common import TERM_SIGNAME
17 + from kombu.matcher import match
18 + from kombu.pidbox import Mailbox
19 + from kombu.utils.compat import register_after_fork
20 + from kombu.utils.functional import lazy
21 + from kombu.utils.objects import cached_property
22 +
23 + from celery.exceptions import DuplicateNodeNameWarning
24 + from celery.utils.log import get_logger
25 + from celery.utils.text import pluralize
26 +
27 + from branch_dictionary import branch_coverage
28 +
29 + __all__ = ('Inspect', 'Control', 'flatten_reply')
```

```
72 +
73 + class Inspect:
74 +     """API for inspecting workers.
75 +
76 +     This class provides proxy for accessing Inspect API of workers. The API is
77 +     defined in :py:mod:`celery.worker.control`
78 +     """
79 +
80 +     app = None
81 +
82 +     def __init__(self, destination=None, timeout=1.0, callback=None,
83 +                  connection=None, app=None, limit=None, pattern=None,
84 +                  matcher=None):
85 +         self.app = app or self.app
86 +         self.destination = destination
87 +         self.timeout = timeout
88 +         self.callback = callback
89 +         self.connection = connection
90 +         self.limit = limit
91 +         self.pattern = pattern
92 +         self.matcher = matcher
93 +
94 +     def _prepare(self, reply):
95 +         if reply:
96 +             branch_coverage["inspect_prepare1"] = True
97 +             by_node = flatten_reply(reply)
98 +             if (self.destination and
99 +                 not isinstance(self.destination, (list, tuple))):
100 +                 branch_coverage["inspect_prepare3"] = True
101 +                 return by_node.get(self.destination)
102 +             else:
103 +                 branch_coverage["inspect_prepare4"] = True
104 +
105 +             if self.pattern:
106 +                 branch_coverage["inspect_prepare5"] = True
107 +                 pattern = self.pattern
108 +                 matcher = self.matcher
109 +
110 +                 def match_with_coverage(node, pattern, matcher):
111 +                     if match(node, pattern, matcher):
112 +                         branch_coverage["inspect_prepare7"] = True
113 +                         return True
114 +                     else:
115 +                         branch_coverage["inspect_prepare8"] = True
116 +                         return False
117 +
118 +                 return {node: reply for node, reply in by_node.items()
119 +                         if match_with_coverage(node, pattern, matcher)}
120 +             else:
121 +                 branch_coverage["inspect_prepare9"] = True
122 +                 return by_node
123 +             else:
124 +                 branch_coverage["inspect_prepare2"] = True
```

```

_handle_conf_update1 is covered: False
_handle_conf_update2 is covered: False
worker_main1 is covered: False
worker_main2 is covered: False
worker_main3 is covered: False
worker_main4 is covered: False
Inspect._prepare1 is covered: True
Inspect._prepare2 is covered: True
Inspect._prepare3 is covered: True
Inspect._prepare4 is covered: True
Inspect._prepare5 is covered: False
Inspect._prepare6 is covered: True
Inspect._prepare7 is covered: False
Inspect._prepare8 is covered: False
Worker.on_start1 is covered: False
Worker.on_start2 is covered: True
Worker.on_start3 is covered: True
Worker.on_start4 is covered: False
Worker.on_start5 is covered: True
Worker.on_start6 is covered: False
Worker.on_start7 is covered: False
Worker.on_start8 is covered: True
Worker.on_start9 is covered: True
Worker.on_start10 is covered: False
Worker.on_start11 is covered: False
Worker.on_start12 is covered: True

Branch Coverage Percentage:

_handle_conf_update: 0 out of 2 branches covered (0.00%)
worker_main: 0 out of 4 branches covered (0.00%)
Inspect._prepare: 5 out of 8 branches covered (62.50%)
Worker.on_start: 6 out of 12 branches covered (50.00%)

```

## Function 2: celery/apps/worker.py Worker.on\_start

```

799  ...  @@ -1,389 +1,410 @@
...  ...
1  - """Worker command-line program.
2  -
3  - This module is the 'program-version' of :mod:'celery.worker'.
4  -
5  - It does everything necessary to run that module
6  - as an actual application, like installing signal handlers,
7  - platform tweaks, and so on.
8  - """
9  - import logging
10 - import os
11 - import platform as platform
12 - import sys
13 - from datetime import datetime
14 - from functools import partial
15 -
16 - from billiard.common import RMAP_SIGTERM
17 - from billiard.process import current_process
18 - from kombu.utils.encoding import safe_str
19 -
20 - from celery import VERSION_BANNER, platforms, signals
21 - from celery.app import trace
22 - from celery.loaders.app import AppLoader
23 - from celery.platforms import EX_FAILURE, EX_OK, check_privileges
24 - from celery.utils import static, term
25 - from celery.utils.debug import cry
26 - from celery.utils.imports import qualname
27 - from celery.utils.log import get_logger, in_sighandler, set_in_sighandler
28 - from celery.utils.text import pluralize
29 - from celery.worker import WorkController
30 -
31 - __all__ = ('Worker',)
32 -
33 - logger = get_logger(__name__)
34 - is_jython = sys.platform.startswith('java')
35 - is_pypy = hasattr(sys, 'pypy_version_info')

```

```

121 - def on_start(self):
122 -     app = self.app
123 -     super().on_start()
124 -
125 -     # this signal can be used to, for example, change queues after
126 -     # the -Q option has been applied.
127 -     signals.celeryd_after_setup.send(
128 -         sender=self.hostname, instance=self, conf=app.conf,
129 -     )
130 -
131 -     if self.purge:
132 -         self.purge_messages()
133 -
134 -     if not self.quiet:
135 -         self.emit_banner()
136 -
137 -     self.set_process_status('-active-')
138 -     self.install_platform_tweaks(self)
139 -     if not self._custom_logging and self.redirect_stdouts:
140 -         app.log.redirect_stdouts(self.redirect_stdouts_level)
141 -
142 -     # TODO: Remove the following code in Celery 6.0
143 -     # This qualifies as a hack for issue #6366.
144 -     warn_deprecated = True
145 -     config_source = app._config_source
146 -     if isinstance(config_source, str):
147 -         # Don't raise the warning when the settings originate from
148 -         # django.conf.settings
149 -         warn_deprecated = config_source.lower() not in [
150 -             'django.conf.settings',
151 -         ]
152 -
153 -     if warn_deprecated:
154 -         if app.conf.maybe_warn_deprecated_settings():
155 -             logger.warning(
156 -                 "Please run 'celery upgrade settings path/to/settings.py' "
157 -                 "to avoid these warnings and to allow a smoother upgrade "
158 -                 "to Celery 6.0."
159 -             )
160 -

```



```

1 + """Worker command-line program.
2 +
3 + This module is the 'program-version' of :mod:`celery.worker`.
4 +
5 + It does everything necessary to run that module
6 + as an actual application, like installing signal handlers,
7 + platform tweaks, and so on.
8 + """
9 + import logging
10 + import os
11 + import platform as _platform
12 + import sys
13 + from datetime import datetime
14 + from functools import partial
15 +
16 + from billiard.common import REMAP_SIGTERM
17 + from billiard.process import current_process
18 + from kombu.utils.encoding import safe_str
19 +
20 + from celery import VERSION_BANNER, platforms, signals
21 + from celery.app import trace
22 + from celery.loaders.app import AppLoader
23 + from celery.platforms import EX_FAILURE, EX_OK, check_privileges
24 + from celery.utils import static, term
25 + from celery.utils.debug import cry
26 + from celery.utils.imports import qualname
27 + from celery.utils.log import get_logger, in_sighandler, set_in_sighandler
28 + from celery.utils.text import pluralize
29 + from celery.worker import WorkController
30 +
31 + from branch_dictionary import branch_coverage
32 +
33 + __all__ = ('Worker',)
34 +
35 + logger = get_logger(__name__)
36 + is_jython = sys.platform.startswith('java')
37 + is_pypy = hasattr(sys, 'pypy_version_info')
38 +

```

```

#test 2
def on_start(self):
    app = self.app
    super().on_start()

    # this signal can be used to, for example, change queues after
    # the -Q option has been applied.
    signals.celeryd_after_setup.send(
        sender=self.hostname, instance=self, conf=app.conf,
    )

    if self.purge:
        branch_coverage["Worker.on_start1"] = True
        self.purge_messages()
    else:
        branch_coverage["Worker.on_start2"] = True

    if not self.quiet:
        branch_coverage["Worker.on_start3"] = True
        self.emit_banner()
    else:
        branch_coverage["Worker.on_start4"] = True

    self.set_process_status('-active-')
    self.install_platform_tweaks(self)
    if not self._custom_logging and self.redirect_stdouts:
        branch_coverage["Worker.on_start5"] = True
        app.log.redirect_stdouts(self.redirect_stdouts_level)
    else:
        branch_coverage["Worker.on_start6"] = True

    # TODO: Remove the following code in Celery 6.0
    # This qualifies as a hack for issue #6366.
    warn_deprecated = True
    config_source = app._config_source
    if isinstance(config_source, str):
        branch_coverage["Worker.on_start7"] = True
        # Don't raise the warning when the settings originate from
        # django.conf.settings
        warn_deprecated = config_source.lower() not in [
            'django.conf.settings',
        ]
    else:
        branch_coverage["Worker.on_start8"] = True

    if warn_deprecated:
        branch_coverage["Worker.on_start9"] = True
        if app.conf.maybe_warn_deprecated_settings():
            branch_coverage["Worker.on_start11"] = True
            logger.warning(
                "Please run 'celery upgrade settings path/to/settings.py' "
                "to avoid these warnings and to allow a smoother upgrade "
                "to Celery 6.0."
            )
        else:
            branch_coverage["Worker.on_start12"] = True
    else:
        branch_coverage["Worker.on_start10"] = True

```

```
_handle_conf_update1 is covered: False
_handle_conf_update2 is covered: False
worker_main1 is covered: False
worker_main2 is covered: False
worker_main3 is covered: False
worker_main4 is covered: False
Inspect._prepare1 is covered: True
Inspect._prepare2 is covered: True
Inspect._prepare3 is covered: True
Inspect._prepare4 is covered: True
Inspect._prepare5 is covered: False
Inspect._prepare6 is covered: True
Inspect._prepare7 is covered: False
Inspect._prepare8 is covered: False
Worker.on_start1 is covered: False
Worker.on_start2 is covered: True
Worker.on_start3 is covered: True
Worker.on_start4 is covered: False
Worker.on_start5 is covered: True
Worker.on_start6 is covered: False
Worker.on_start7 is covered: False
Worker.on_start8 is covered: True
Worker.on_start9 is covered: True
Worker.on_start10 is covered: False
Worker.on_start11 is covered: False
Worker.on_start12 is covered: True

Branch Coverage Percentage:

_handle_conf_update: 0 out of 2 branches covered (0.00%)
worker_main: 0 out of 4 branches covered (0.00%)
Inspect._prepare: 5 out of 8 branches covered (62.50%)
Worker.on_start: 6 out of 12 branches covered (50.00%)
```

```

...  ...  @@ -0,0 +1,46 @@
1 + import os
2 + from branch_dictionary import branch_coverage, branch_totals
3 +
4 + # Function to print branch coverage
5 + def print_branch_coverage():
6 +     print("\nBRANCH COVERAGE:\n")
7 +
8 +     covered_branches = {
9 +         "_handle_conf_update": 0,
10 +         "worker_main": 0,
11 +         "Inspect._prepare": 0,
12 +         "Worker.on_start": 0
13 +     }
14 +
15 +     print("Branch Hits:\n")
16 +     s = ""
17 +     for key, value in branch_coverage.items():
18 +         s += f"{key} is covered: {value}\n"
19 +     print(s)
20 +
21 +     for key, value in branch_coverage.items():
22 +         if value:
23 +             if key.startswith("worker_main"):
24 +                 covered_branches["worker_main"] += 1
25 +             elif key.startswith("_handle_conf_update"):
26 +                 covered_branches["_handle_conf_update"] += 1
27 +             elif key.startswith("Inspect._prepare"):
28 +                 covered_branches["Inspect._prepare"] += 1
29 +             elif key.startswith("Worker.on_start"):
30 +                 covered_branches["Worker.on_start"] += 1
31 +
32 +     print("Branch Coverage Percentage:\n")
33 +     for key, value in branch_totals.items():
34 +         covered = covered_branches[key]
35 +         coverage_percentage = (covered / value) * 100
36 +         print(f"{key}: {covered} out of {value} branches covered ({coverage_percentage:.2f}%)")
37 +
38 + # Function to run all tests and print coverage
39 + def run_coverage():
40 +     import pytest
41 +     pytest.main(['t/unit/apps/test_worker.py'])
42 +
43 +     print_branch_coverage()
44 +
45 + if __name__ == '__main__':
46 +     run_coverage()

```

```
38 branch_dictionary.py
... @@ -0,0 +1,38 @@
1 + branch_coverage = {
2 +     "_handle_conf_update1": False,      # if 'task_routes' in kwargs or 'task_routes' in args
3 +     "_handle_conf_update2": False,      # else branch
4 +
5 +     "worker_main1": False,              # if argv is None
6 +     "worker_main2": False,              # else branch
7 +     "worker_main3": False,              # 'worker' not in argv
8 +     "worker_main4": False,              # else branch
9 +
10 +     "Inspect._prepare1": False,
11 +     "Inspect._prepare2": False,
12 +     "Inspect._prepare3": False,
13 +     "Inspect._prepare4": False,
14 +     "Inspect._prepare5": False,
15 +     "Inspect._prepare6": False,
16 +     "Inspect._prepare7": False,
17 +     "Inspect._prepare8": False,
18 +
19 +     "Worker.on_start1": False,
20 +     "Worker.on_start2": False,
21 +     "Worker.on_start3": False,
22 +     "Worker.on_start4": False,
23 +     "Worker.on_start5": False,
24 +     "Worker.on_start6": False,
25 +     "Worker.on_start7": False,
26 +     "Worker.on_start8": False,
27 +     "Worker.on_start9": False,
28 +     "Worker.on_start10": False,
29 +     "Worker.on_start11": False,
30 +     "Worker.on_start12": False
31 + }
32 +
33 + branch_totals = {
34 +     "_handle_conf_update": 2,
35 +     "worker_main": 4,
36 +     "Inspect._prepare": 8,
37 +     "Worker.on_start": 12
38 + }
```

Jacqueline Bouwman  
Function 1: TermLogger.info

```

93 -
94 - $ # lists also works with named workers
95 - $ celery multi start foo bar baz xuzzy -c 3 -c:foo,bar,baz 10
96 - celery worker -n foo@myhost -c 10
97 - celery worker -n bar@myhost -c 10
98 - celery worker -n baz@myhost -c 10
99 - celery worker -n xuzzy@myhost -c 3
100 -
101 - import os
102 - import signal
103 - import sys
104 - from functools import wraps
105 -
106 - import click
107 - from kombu.utils.objects import cached_property
108 -
109 - from celery import VERSION_BANNER
110 - from celery.apps.multi import Cluster, MultiParser, NamespacedOptionParser
111 - from celery.bin.base import CeleryCommand, handle_preload_options
112 - from celery.platforms import EX_FAILURE, EX_OK, signals
113 - from celery.utils import term
114 - from celery.utils.text import pluralize
115 -

```

```

170 - class TermLogger:
171 -
172 -     splash_text = 'celery multi v{version}'
173 -     splash_context = {'version': VERSION_BANNER}
174 -
175 -     #: Final exit code.
176 -     retcode = 0
177 -
178 -     def setup_terminal(self, stdout, stderr,
179 -                        nosplash=False, quiet=False, verbose=False,
180 -                        no_color=False, **kwargs):
181 -         self.stdout = stdout or sys.stdout
182 -         self.stderr = stderr or sys.stderr
183 -         self.nosplash = nosplash
184 -         self.quiet = quiet
185 -         self.verbose = verbose
186 -         self.no_color = no_color
187 -
188 -     def ok(self, m, newline=True, file=None):
189 -         self.say(m, newline=newline, file=file)
190 -         return EX_OK
191 -
192 -     def say(self, m, newline=True, file=None):
193 -         print(m, file=file or self.stdout, end='\n' if newline else ' ')
194 -
195 -     def carp(self, m, newline=True, file=None):
196 -         return self.say(m, newline, file or self.stderr)
197 -
198 -     def error(self, msg=None):
199 -         if msg:
200 -             self.carp(msg)
201 -             self.usage()
202 -             return EX_FAILURE
203 -
204 -     def info(self, msg, newline=True):
205 -         if self.verbose:
206 -             self.note(msg, newline=newline)

```

```

99 + celery worker -n xuzzy@myhost -c 3
100 +
101 + import os
102 + import signal
103 + import sys
104 + from functools import wraps
105 +
106 + import click
107 + from kombu.utils.objects import cached_property
108 +
109 + from celery import VERSION_BANNER
110 + from celery.apps.multi import Cluster, MultiParser, NamespacedOptionParser
111 + from celery.bin.base import CeleryCommand, handle_preload_options
112 + from celery.platforms import EX_FAILURE, EX_OK, signals
113 + from celery.utils import term
114 + from celery.utils.text import pluralize
115 +
116 + from branch_dictionary import branch_coverage
117 +
118 + __all__ = ('MultiTool',)
119 +

```

```

171 +
172 + class TermLogger:
173 +
174 +     splash_text = 'celery multi v{version}'
175 +     splash_context = {'version': VERSION_BANNER}
176 +
177 +     #: Final exit code.
178 +     retcode = 0
179 +
180 +     def setup_terminal(self, stdout, stderr,
181 +                        nosplash=False, quiet=False, verbose=False,
182 +                        no_color=False, **kwargs):
183 +         self.stdout = stdout or sys.stdout
184 +         self.stderr = stderr or sys.stderr
185 +         self.nosplash = nosplash
186 +         self.quiet = quiet
187 +         self.verbose = verbose
188 +         self.no_color = no_color
189 +
190 +     def ok(self, m, newline=True, file=None):
191 +         self.say(m, newline=newline, file=file)
192 +         return EX_OK
193 +
194 +     def say(self, m, newline=True, file=None):
195 +         print(m, file=file or self.stdout, end='\n' if newline else '')
196 +
197 +     def carp(self, m, newline=True, file=None):
198 +         return self.say(m, newline, file or self.stderr)
199 +
200 +     def error(self, msg=None):
201 +         if msg:
202 +             self.carp(msg)
203 +             self.usage()
204 +             return EX_FAILURE
205 +
206 +     @test 1
207 +     def info(self, msg, newline=True):
208 +         if self.verbose:
209 +             branch_coverage["TermLogger.info1"] = True
210 +             self.note(msg, newline=newline)
211 +         else:
212 +             branch_coverage["TermLogger.info2"] = True
213 +

```

```

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! Interrupted: 2 errors during collection !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
----- 2 errors in 0.24s -----

BRANCH COVERAGE:

Branch Hits:
handle_conf_update1 is covered: False
handle_conf_update2 is covered: False
worker_main1 is covered: False
worker_main2 is covered: False
worker_main3 is covered: False
worker_main4 is covered: False
werkLogger.info1 is covered: False
werkLogger.info2 is covered: False
CeleryOption.get_default1 is covered: False
CeleryOption.get_default2 is covered: False

Branch Coverage Percentage:
handle_conf_update: 0 out of 2 branches covered (0.00%)
worker_main: 0 out of 4 branches covered (0.00%)
werkLogger.info: 0 out of 2 branches covered (0.00%)
CeleryOption.get_default: 0 out of 2 branches covered (0.00%)

```

## Function 2: CeleryOption.get\_default

```

617 celery/bin/assep
... @@ -1,385 +1,312 @@
1 - """Click customizations for celery."""
2 - import json
3 - import numbers
4 - from collections import OrderedDict
5 - from functools import update_wrapper
6 - from pprint import pformat
7 - from typing import Any
8 -
9 - import click
10 - from click import Context, ParamType
11 - from kombu.utils.objects import cached_property
12 -
13 - from celery.state import get_current_app
14 - from celery.signals import user_preload_options
15 - from celery.utils import text
16 - from celery.utils.log import mlevel
17 - from celery.utils.time import maybe_iso8601
18 -
19 - try:
20 -     from pygments import highlight
21 -     from pygments.formatters import Terminal256Formatter
22 -     from pygments.lexers import PythonLexer
23 - except ImportError:
24 -     def highlight(s, *args, **kwargs):
25 -         """Place holder function in case pygments is missing."""
26 -         return s
27 -     LEXER = None
28 -     FORMATTER = None
29 - else:
30 -     LEXER = PythonLexer()
31 -     FORMATTER = Terminal256Formatter()
32 -
33 -
34 -
35 -
36 -
37 -
38 -
39 -
40 -
41 -
42 -
43 -
44 -
45 -
46 -
47 -
48 -
49 -
50 -
51 -
52 -
53 -
54 -
55 -
56 -
57 -
58 -
59 -
60 -
61 -
62 -
63 -
64 -
65 -
66 -
67 -
68 -
69 -
70 -
71 -
72 -
73 -
74 -
75 -
76 -
77 -
78 -
79 -
80 -
81 -
82 -
83 -
84 -
85 -
86 -
87 -
88 -
89 -
90 -
91 -
92 -
93 -
94 -
95 -
96 -
97 -
98 -
99 -
100 -
101 -
102 -
103 -
104 -
105 -
106 -
107 -
108 -
109 -
110 -
111 -
112 -
113 -
114 -
115 -
116 -
117 -
118 -
119 -
120 -
121 -
122 -
123 -
124 -
125 -
126 -
127 -
128 -
129 -
130 -
131 -
132 -
133 -
134 -
135 -
136 -
137 -
138 -
139 -
140 -
141 -
142 -
143 -
144 -
145 -
146 -
147 -
148 -
149 -
150 -
151 -
152 -
153 -
154 -
155 -
156 -
157 -
158 -
159 -
160 -
161 -
162 -
163 -
164 -
165 -
166 -
167 -
168 -
169 -
170 -
171 -
172 -
173 -
174 -
175 -
176 -
177 -
178 -
179 -
180 -
181 -
182 -
183 -
184 -
185 -
186 -
187 -
188 -
189 -
190 -
191 -
192 -
193 -
194 -
195 -
196 -
197 -
198 -
199 -
200 -
201 -
202 -
203 -
204 -
205 -
206 -
207 -
208 -
209 -
210 -
211 -
212 -
213 -
214 -
215 -
216 -
217 -
218 -
219 -
220 -
221 -
222 -
223 -
224 -
225 -
226 -
227 -
228 -
229 -
230 -
231 -
232 -
233 -
234 -
235 -
236 -
237 -
238 -
239 -
240 -
241 -
242 -
243 -
244 -
245 -
246 -
247 -
248 -
249 -
250 -
251 -
252 -
253 -
254 -
255 -
256 -
257 -
258 -
259 -
260 -
261 -
262 -
263 -
264 -
265 -
266 -
267 -
268 -
269 -
270 -
271 -
272 -
273 -
274 -
275 -
276 -
277 -
278 -
279 -
280 -
281 -
282 -
283 -
284 -
285 -
286 -
287 -
288 -
289 -
290 -
291 -
292 -
293 -
294 -
295 -
296 -
297 -
298 -
299 -
300 -
301 -
302 -
303 -
304 -
305 -
306 -
307 -
308 -
309 -
310 -
311 -
312 -
313 -
314 -
315 -
316 -
317 -
318 -
319 -
320 -
321 -
322 -
323 -
324 -
325 -
326 -
327 -
328 -
329 -
330 -
331 -
332 -
333 -
334 -
335 -
336 -
337 -
338 -
339 -
340 -
341 -
342 -
343 -
344 -
345 -
346 -
347 -
348 -
349 -
350 -
351 -
352 -
353 -
354 -
355 -
356 -
357 -
358 -
359 -
360 -
361 -
362 -
363 -
364 -
365 -
366 -
367 -
368 -
369 -
370 -
371 -
372 -
373 -
374 -
375 -
376 -
377 -
378 -
379 -
380 -
381 -
382 -
383 -
384 -
385 -
386 -
387 -
388 -
389 -
390 -
391 -
392 -
393 -
394 -
395 -
396 -
397 -
398 -
399 -
400 -
401 -
402 -
403 -
404 -
405 -
406 -
407 -
408 -
409 -
410 -
411 -
412 -
413 -
414 -
415 -
416 -
417 -
418 -
419 -
420 -
421 -
422 -
423 -
424 -
425 -
426 -
427 -
428 -
429 -
430 -
431 -
432 -
433 -
434 -
435 -
436 -
437 -
438 -
439 -
440 -
441 -
442 -
443 -
444 -
445 -
446 -
447 -
448 -
449 -
450 -
451 -
452 -
453 -
454 -
455 -
456 -
457 -
458 -
459 -
460 -
461 -
462 -
463 -
464 -
465 -
466 -
467 -
468 -
469 -
470 -
471 -
472 -
473 -
474 -
475 -
476 -
477 -
478 -
479 -
480 -
481 -
482 -
483 -
484 -
485 -
486 -
487 -
488 -
489 -
490 -
491 -
492 -
493 -
494 -
495 -
496 -
497 -
498 -
499 -
500 -
501 -
502 -
503 -
504 -
505 -
506 -
507 -
508 -
509 -
510 -
511 -
512 -
513 -
514 -
515 -
516 -
517 -
518 -
519 -
520 -
521 -
522 -
523 -
524 -
525 -
526 -
527 -
528 -
529 -
530 -
531 -
532 -
533 -
534 -
535 -
536 -
537 -
538 -
539 -
540 -
541 -
542 -
543 -
544 -
545 -
546 -
547 -
548 -
549 -
550 -
551 -
552 -
553 -
554 -
555 -
556 -
557 -
558 -
559 -
560 -
561 -
562 -
563 -
564 -
565 -
566 -
567 -
568 -
569 -
570 -
571 -
572 -
573 -
574 -
575 -
576 -
577 -
578 -
579 -
580 -
581 -
582 -
583 -
584 -
585 -
586 -
587 -
588 -
589 -
590 -
591 -
592 -
593 -
594 -
595 -
596 -
597 -
598 -
599 -
600 -
601 -
602 -
603 -
604 -
605 -
606 -
607 -
608 -
609 -
610 -
611 -
612 -
613 -
614 -
615 -
616 -
617 -
618 -
619 -
620 -
621 -
622 -
623 -
624 -
625 -
626 -
627 -
628 -
629 -
630 -
631 -
632 -
633 -
634 -
635 -
636 -
637 -
638 -
639 -
640 -
641 -
642 -
643 -
644 -
645 -
646 -
647 -
648 -
649 -
650 -
651 -
652 -
653 -
654 -
655 -
656 -
657 -
658 -
659 -
660 -
661 -
662 -
663 -
664 -
665 -
666 -
667 -
668 -
669 -
670 -
671 -
672 -
673 -
674 -
675 -
676 -
677 -
678 -
679 -
680 -
681 -
682 -
683 -
684 -
685 -
686 -
687 -
688 -
689 -
690 -
691 -
692 -
693 -
694 -
695 -
696 -
697 -
698 -
699 -
700 -
701 -
702 -
703 -
704 -
705 -
706 -
707 -
708 -
709 -
710 -
711 -
712 -
713 -
714 -
715 -
716 -
717 -
718 -
719 -
720 -
721 -
722 -
723 -
724 -
725 -
726 -
727 -
728 -
729 -
730 -
731 -
732 -
733 -
734 -
735 -
736 -
737 -
738 -
739 -
740 -
741 -
742 -
743 -
744 -
745 -
746 -
747 -
748 -
749 -
750 -
751 -
752 -
753 -
754 -
755 -
756 -
757 -
758 -
759 -
760 -
761 -
762 -
763 -
764 -
765 -
766 -
767 -
768 -
769 -
770 -
771 -
772 -
773 -
774 -
775 -
776 -
777 -
778 -
779 -
780 -
781 -
782 -
783 -
784 -
785 -
786 -
787 -
788 -
789 -
790 -
791 -
792 -
793 -
794 -
795 -
796 -
797 -
798 -
799 -
800 -
801 -
802 -
803 -
804 -
805 -
806 -
807 -
808 -
809 -
810 -
811 -
812 -
813 -
814 -
815 -
816 -
817 -
818 -
819 -
820 -
821 -
822 -
823 -
824 -
825 -
826 -
827 -
828 -
829 -
830 -
831 -
832 -
833 -
834 -
835 -
836 -
837 -
838 -
839 -
840 -
841 -
842 -
843 -
844 -
845 -
846 -
847 -
848 -
849 -
850 -
851 -
852 -
853 -
854 -
855 -
856 -
857 -
858 -
859 -
860 -
861 -
862 -
863 -
864 -
865 -
866 -
867 -
868 -
869 -
870 -
871 -
872 -
873 -
874 -
875 -
876 -
877 -
878 -
879 -
880 -
881 -
882 -
883 -
884 -
885 -
886 -
887 -
888 -
889 -
890 -
891 -
892 -
893 -
894 -
895 -
896 -
897 -
898 -
899 -
900 -
901 -
902 -
903 -
904 -
905 -
906 -
907 -
908 -
909 -
910 -
911 -
912 -
913 -
914 -
915 -
916 -
917 -
918 -
919 -
920 -
921 -
922 -
923 -
924 -
925 -
926 -
927 -
928 -
929 -
930 -
931 -
932 -
933 -
934 -
935 -
936 -
937 -
938 -
939 -
940 -
941 -
942 -
943 -
944 -
945 -
946 -
947 -
948 -
949 -
950 -
951 -
952 -
953 -
954 -
955 -
956 -
957 -
958 -
959 -
960 -
961 -
962 -
963 -
964 -
965 -
966 -
967 -
968 -
969 -
970 -
971 -
972 -
973 -
974 -
975 -
976 -
977 -
978 -
979 -
980 -
981 -
982 -
983 -
984 -
985 -
986 -
987 -
988 -
989 -
990 -
991 -
992 -
993 -
994 -
995 -
996 -
997 -
998 -
999 -
1000 -

```

```

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! Interrupted: 2 errors during collection !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
----- 2 errors in 0.24s -----

BRANCH COVERAGE:

Branch Hits:
handle_conf_update1 is covered: False
handle_conf_update2 is covered: False
worker_main1 is covered: False
worker_main2 is covered: False
worker_main3 is covered: False
worker_main4 is covered: False
werkLogger.info1 is covered: False
werkLogger.info2 is covered: False
CeleryOption.get_default1 is covered: False
CeleryOption.get_default2 is covered: False

Branch Coverage Percentage:
handle_conf_update: 0 out of 2 branches covered (0.00%)
worker_main: 0 out of 4 branches covered (0.00%)
werkLogger.info: 0 out of 2 branches covered (0.00%)
CeleryOption.get_default: 0 out of 2 branches covered (0.00%)

```

```
22 branch_directionary.py
... @@ -0,0 +1,22 @@
1 + branch_coverage = {
2 +     "_handle_conf_update1": False,      # if 'task_routes' in kwargs or 'task_routes' in args
3 +     "_handle_conf_update2": False,      # else branch
4 +
5 +     "worker_main1": False,              # if argv is None
6 +     "worker_main2": False,              # else branch
7 +     "worker_main3": False,              # 'worker' not in argv
8 +     "worker_main4": False,              # else branch
9 +
10 +     "TermLogger.info1": False,
11 +     "TermLogger.info2": False,
12 +
13 +     "CeleryOption.get_default1": False,
14 +     "CeleryOption.get_default2": False
15 + }
16 +
17 + branch_totals = {
18 +     "_handle_conf_update": 2,
19 +     "worker_main": 4,
20 +     "TermLogger.info": 2,
21 +     "CeleryOption.get_default": 2
22 + }
```

```
46 branch_tracker.py
... @@ -0,0 +1,46 @@
1 + import os
2 + from branch_dictionary import branch_coverage, branch_totals
3 +
4 + # Function to print branch coverage
5 + def print_branch_coverage():
6 +     print("\nBRANCH COVERAGE:\n")
7 +
8 +     covered_branches = {
9 +         "_handle_conf_update": 0,
10 +         "worker_main": 0,
11 +         "TermLogger.info": 0,
12 +         "CeleryOption.get_default": 0
13 +     }
14 +
15 +     print("Branch Hits:\n")
16 +     s = ""
17 +     for key, value in branch_coverage.items():
18 +         s += f"{key} is covered: {value}\n"
19 +     print(s)
20 +
21 +     for key, value in branch_coverage.items():
22 +         if value:
23 +             if key.startswith("worker_main"):
24 +                 covered_branches["worker_main"] += 1
25 +             elif key.startswith("_handle_conf_update"):
26 +                 covered_branches["_handle_conf_update"] += 1
27 +             elif key.startswith("TermLogger.info"):
28 +                 covered_branches["TermLogger.info"] += 1
29 +             elif key.startswith("CeleryOption.get_default"):
30 +                 covered_branches["CeleryOption.get_default"] += 1
31 +
32 +     print("Branch Coverage Percentage:\n")
33 +     for key, value in branch_totals.items():
34 +         covered = covered_branches[key]
35 +         coverage_percentage = (covered / value) * 100
36 +         print(f"{key}: {covered} out of {value} branches covered ({coverage_percentage:.2f}%)")
37 +
38 + # Function to run all tests and print coverage
39 + def run_coverage():
40 +     import pytest
41 +     pytest.main(["t/unit/app/test_celery.py"])
42 +
43 +     print_branch_coverage()
44 +
45 + if __name__ == '__main__':
46 +     run_coverage()
```

<Group member name>

<Function 1 name>

<Show a patch (diff) or a link to a commit made in your forked repository that shows the instrumented code to gather coverage measurements>

<Provide a screenshot of the coverage results output by the instrumentation>



<Function 2 name>

<Provide the same kind of information provided for Function 1>

## Coverage improvement

### Individual tests

Rose Hollander

Function 1 Tests: celery/app/amqp.py

AMQP.\_handle\_conf\_update

```
... @@ -1,407 +1,422 @@
1 - from datetime import datetime, timedelta, timezone
2 - from unittest.mock import Mock, patch
3 -
4 - import pytest
5 - from kombu import Exchange, Queue
6 -
7 - from celery import uuid
8 - from celery.app.amqp import Queues, utf8dict
9 - from celery.utils.time import to_utc
10 -
11 -
12 - class test_TaskConsumer:
13 -
14 -     def test_accept_content(self, app):
15 -         with app.pool.acquire(block=True) as con:
16 -             app.conf.accept_content = ['application/json']
17 -             assert app.amqp.TaskConsumer(con).accept == {
18 -                 'application/json',
19 -             }
20 -             assert app.amqp.TaskConsumer(con, accept=['json']).accept == {
21 -                 'application/json',
22 -             }
23 -
```

```
1 + from datetime import datetime, timedelta, timezone
2 + from unittest.mock import Mock, patch
3 +
4 + import pytest
5 + from kombu import Exchange, Queue
6 +
7 + from celery import uuid
8 + from celery.app.amqp import Queues, utf8dict, AMQP
9 + from celery.utils.time import to_utc
10 +
11 +
12 + class test_handle_conf_update:
13 +
14 +     def test_conf_update_task_routes(self, app):
15 +         with patch.object(AMQP, 'flush_routes') as mock_flush_routes, patch.object(AMQP, 'Router') as mock_router:
16 +             app.amqp._handle_conf_update("task_routes")
17 +             mock_flush_routes.assert_called_once()
18 +             mock_router.assert_called_once()
19 +
20 +     def test_conf_update_no_args(self, app):
21 +         with patch.object(AMQP, 'flush_routes') as mock_flush_routes, patch.object(AMQP, 'Router') as mock_router:
22 +             app.amqp._handle_conf_update()
23 +             mock_flush_routes.assert_not_called()
24 +             mock_router.assert_not_called()
25 +
26 +
27 + class test_TaskConsumer:
28 +
29 +     def test_accept_content(self, app):
30 +         with app.pool.acquire(block=True) as con:
31 +             app.conf.accept_content = ['application/json']
32 +             assert app.amqp.TaskConsumer(con).accept == {
33 +                 'application/json',
34 +             }
35 +             assert app.amqp.TaskConsumer(con, accept=['json']).accept == {
36 +                 'application/json',
37 +             }
38 -
```

Coverage report: 84%

Files Functions Classes

| File               | function                 | statements | missing | excluded | branches | partial | coverage |
|--------------------|--------------------------|------------|---------|----------|----------|---------|----------|
| celery/app/amqp.py | AMQP._handle_conf_update | 4          | 4       | 0        | 2        | 0       | 0%       |
| celery/app/base.py | Celery.worker main       | 5          | 2       | 0        | 4        | 2       | 56%      |

Here it went from 0% to 100%, there were no tests so I created two new ones: one for branch where "task\_routes" is in the arguments and one where it isn't.

Function 2 Tests: celery/app/base.py Celery.worker\_main

testing worker\_main in test\_app.py

```
594 -  
595 - @patch('celery.bin.celery.celery')  
596 - def test_worker_main(self, mocked_celery):  
597 -     self.app.worker_main(argv=['worker', '--help'])  
598 -  
599 -     mocked_celery.main.assert_called_with(  
600 -         args=['worker', '--help'], standalone_mode=False)  
601 -  
602 - def test_config_from_envvar(self, monkeypatch):  
603 -     monkeypatch.setenv("CELERYTEST_CONFIG_OBJECT", 't.unit.app.test_app')  
604 -     self.app.config_from_envvar('CELERYTEST_CONFIG_OBJECT')  
605 -     assert self.app.conf.THIS_IS_A_KEY == 'this is a value'  
606 -
```


```
595 + @patch('celery.bin.celery.celery')  
596 + def test_worker_main(self, mocked_celery):  
597 +     self.app.worker_main(argv=['worker', '--help'])  
598 +  
599 +     mocked_celery.main.assert_called_with(  
600 +         args=['worker', '--help'], standalone_mode=False)  
601 +  
602 + @patch('celery.bin.celery.celery')  
603 + def test_worker_main_no_args(self, mocked_celery):  
604 +     with pytest.raises(ValueError):  
605 +         self.app.worker_main(argv=None)  
606 +  
607 + @patch('celery.bin.celery.celery')  
608 + def test_worker_main_invalid_arg(self, mocked_celery):  
609 +     with pytest.raises(ValueError):  
610 +         self.app.worker_main(argv='test')  
611 +  
612 + def test_config_from_envvar(self, monkeypatch):  
613 +     monkeypatch.setenv("CELERYTEST_CONFIG_OBJECT", 't.unit.app.test_app')  
614 +     self.app.config_from_envvar('CELERYTEST_CONFIG_OBJECT')  
615 +     assert self.app.conf.THIS_IS_A_KEY == 'this is a value'
```

Coverage report: 84%

Files

Functions

Classes

| File               | function                | statements  | missing | excluded | branches | partial | coverage |
|--------------------|-------------------------|--|---------|----------|----------|---------|----------|
| celery/app/amqp.py | AMQP_handle_conf_update | 4  | 4       | 0        | 2        | 0       | 0%       |
| celery/app/base.py | Celery.worker main      | 5  | 2       | 0        | 4        | 2       | 56%      |

Coverage result changed to 100%, because I added two new tests for the missing branches where value errors are raised (when arg is None for first branch and no "worker" in the argument list in the third branch).

Jacqueline Bouwman

Function 1: TermLogger.info

```
25 t/unit/app/test_termlogger.py
... @@ -0,0 +1,25 @@
1 + import pytest
2 + import celery
3 + from unittest.mock import Mock
4 +
5 + from celery.bin.multi import TermLogger
6 +
7 + class test_TermLogger:
8 +
9 +     @pytest.fixture
10 +     def term_logger(self):
11 +         logger = TermLogger()
12 +         logger.note = Mock()
13 +         return logger
14 +
15 +     def test_info_verbose_true(self, term_logger):
16 +         term_logger.verbose = True
17 +         msg = "Test message"
18 +         term_logger.info(msg)
19 +         term_logger.note.assert_called_once_with(msg, newline=True)
20 +
21 +     def test_info_verbose_false(self, term_logger):
22 +         term_logger.verbose = False
23 +         msg = "Test message"
24 +         term_logger.info(msg)
25 +         term_logger.note.assert_not_called()
```

|                     |                 |   |   |   |   |   |    |
|---------------------|-----------------|---|---|---|---|---|----|
| celery/bin/multi.py | TermLogger.info | 2 | 2 | 0 | 2 | 0 | 0% |
|---------------------|-----------------|---|---|---|---|---|----|

Went from 0 to 100% by adding two tests: one that tests when it is verbose (the if branch) and when it isn't (the invisible else branch).

Function 2: CeleryOption.get\_default

|                    |  |                          |     |   |
|--------------------|--|--------------------------|-----|---|
| 1                  | - import pytest  | test_celery              | 3/3 | ^ |
| 2                  | -  |                          |     |   |
| 3                  | - import celery  |                          |     |   |
| 4                  | -  |                          |     |   |
| 5                  | -  |                          |     |   |
| 6                  | - def test_version():  |                          |     |   |
| 7                  | -     assert celery.VERSION  |                          |     |   |
| 8                  | -     assert len(celery.VERSION) >= 3  |                          |     |   |
| 9                  | -     celery.VERSION = (0, 3, 0)   |                          |     |   |
| 10                 | -     assert celery.__version__.count('.') >= 2  |                          |     |   |
| 11                 | -  |                          |     |   |
| 12                 | -  |                          |     |   |
| 13                 | - @pytest.mark.parametrize('attr', [   |                          |     |   |
| 14                 | -     '__author__', '__contact__', '__homepage__', '__docformat__',  |                          |     |   |
| 15                 | - ])   |                          |     |   |
| 16                 | - def test_meta(attr):   |                          |     |   |
| 17                 | -     assert getattr(celery, attr, None)   |                          |     |   |
| 1                  | + import pytest  |                          |     |   |
| 2                  | + import celery  |                          |     |   |
| 3                  | + from unittest.mock import Mock, patch  |                          |     |   |
| 4                  | + import click   |                          |     |   |
| 5                  | +  |                          |     |   |
| 6                  | + # Assuming the CeleryOption class is defined in a module named celery_option   |                          |     |   |
| 7                  | + from celery.bin.base import CeleryOption   |                          |     |   |
| 8                  | +  |                          |     |   |
| 9                  | + def test_version():  |                          |     |   |
| 10                 | +     assert celery.VERSION  |                          |     |   |
| 11                 | +     assert len(celery.VERSION) >= 3  |                          |     |   |
| 12                 | +     celery.VERSION = (0, 3, 0)   |                          |     |   |
| 13                 | +     assert celery.__version__.count('.') >= 2  |                          |     |   |
| 14                 | +  |                          |     |   |
| 15                 | +  |                          |     |   |
| 16                 | + @pytest.mark.parametrize('attr', [   |                          |     |   |
| 17                 | +     '__author__', '__contact__', '__homepage__', '__docformat__',  |                          |     |   |
| 18                 | + ])   |                          |     |   |
| 19                 | + def test_meta(attr):   |                          |     |   |
| 20                 | +     assert getattr(celery, attr, None)   |                          |     |   |
| 21                 | +  |                          |     |   |
| 22                 | + class test_CeleryOption:   |                          |     |   |
| 23                 | +  |                          |     |   |
| 24                 | +     @pytest.fixture  |                          |     |   |
| 25                 | +     def mock_ctx(self):  |                          |     |   |
| 26                 | +         ctx = Mock()   |                          |     |   |
| 27                 | +         ctx.obj = {'default_key': 'default_value'}   |                          |     |   |
| 28                 | +         return ctx   |                          |     |   |
| 29                 | +  |                          |     |   |
| 30                 | +     def test_get_default_with_default_value_from_context(self, mock_ctx):  |                          |     |   |
| 31                 | +         option = CeleryOption(param_decls=['--test'], default_value_from_context='default_key')                            |                          |     |   |
| 32                 | +  |                          |     |   |
| 33                 | +         with patch.object(click.Option, 'get_default', return_value='default_value_from_super') as mock_super_get_default: |                          |     |   |
| 34                 | +             default_value = option.get_default(mock_ctx)   |                          |     |   |
| 35                 | +             assert option.default == 'default_value'   |                          |     |   |
| 36                 | +             assert default_value == 'default_value_from_super'   |                          |     |   |
| 37                 | +             mock_super_get_default.assert_called_once_with(mock_ctx)   |                          |     |   |
| 38                 | +  |                          |     |   |
| 39                 | +     def test_get_default_without_default_value_from_context(self, mock_ctx):   |                          |     |   |
| 40                 | +         option = CeleryOption(param_decls=['--test'])  |                          |     |   |
| 41                 | +  |                          |     |   |
| 42                 | +         with patch.object(click.Option, 'get_default', return_value='default_value_from_super') as mock_super_get_default: |                          |     |   |
| 43                 | +             default_value = option.get_default(mock_ctx)   |                          |     |   |
| 44                 | +             assert option.default != 'default_value'   |                          |     |   |
| 45                 | +             assert default_value == 'default_value_from_super'   |                          |     |   |
| 46                 | +             mock_super_get_default.assert_called_once_with(mock_ctx)   |                          |     |   |
|                    |  |                          |     |   |
| celery/bin/base.py |  | CeleryOption.get_default | 3   | 3 |
|                    |  |                          | 0   | 2 |
|                    |  |                          | 0   | 0 |
|                    |  |                          | 0%  |   |

Went from 0 to 100% by adding two tests: one that tests with the default value (the if branch) and without (the invisible else branch).

Jonathan Davidson

<Test 1>

<Show a patch (diff) or a link to a commit made in your forked repository that shows the new/enhanced test>

```
Enhanced Test_Testy
main
J Davidson committed 1 minute ago
Showing 1 changed file with 26 additions and 1 deletion
Whitebox Greenbox Snap
test_rtti/test_test.py
@@ 1,6-1,6 @@
1 1 import pytest
2
3 2 from clazy.utils.test import assert, ensure_raises, indent, pretty, truncate
4 3
5 4 from clazy.utils.test import assert, assert_raises, indent, 300, pretty, truncate, fill_paragraph
6 5
7 6 NAMETEST = ""
8 7
9 8 the quick brown
10
11 @@ 79,1-79,26 @@ def test_assert_raises():
79 79
80 80 def test_pretty():
81 81     assert pretty("a", "b", "c")
82
83 82 + @pytest.mark.parametrize("l, sep, expected": [
84 83 +     ((["a", "b", "c"], "a", "b", "c"), "abc"),
85 84 +     ((["a", "b", "c"], "\n", "a", "b", "c"), "a\nb\nc"),
86 85 +     ((["a", "b", "c"], "\n", "a", "b", "c"), "a\nb\nc"),
87 86 +     ((["a", "b", "c"], "\n", "a", "b", "c"), "a\nb\nc"),
88 87 +     ((["a", "b", "c"], "\n", "a", "b", "c"), "a\nb\nc"),
89 88 +     ((["a", "b", "c"], "\n", "a", "b", "c"), "a\nb\nc"),
90 89 +     ((["a", "b", "c"], "\n", "a", "b", "c"), "a\nb\nc"),
91 90 +     ((["a", "b", "c"], "\n", "a", "b", "c"), "a\nb\nc"),
92 91 + ]
93 92 + )
94 93 def test_join(l, sep, expected):
95 94     print(f"Testing {l}")
96 95     assert join(l, sep) == expected
97
98 96 + @pytest.mark.parametrize("width, sep, expected": [
99 97 +     ("Line from editor will meet, connector joining all", 30, "a", "Line from editor will meet, connector joining all"),
100 98 +     ("Line from editor will meet, connector joining all", 30, "a", "Line from editor will meet, connector joining all"),
101 99 +     ("First paragraph (indented paragraph)", 30, "a", "First paragraph (indented paragraph)",
102 100 +     ("First paragraph (indented paragraph)", 30, "a", "First paragraph (indented paragraph)",
103 101 +     ("", 30, "a", ""))
104 102 + ]
105 103 + )
106 104 def test_fill_paragraph(width, sep, expected):
107 105     assert fill_paragraph(i, width, sep) == expected
108 106
```

These are both enhanced functions in 1 commit.

<Provide a screenshot of the old coverage results (the same as you already showed above)>

| File                              | function               | statements | missing | excluded | branches ▼ | partial | coverage |
|-----------------------------------|------------------------|------------|---------|----------|------------|---------|----------|
| celery/result.py                  | ResultSet.join_native  | 16         | 0       | 0        | 16         | 1       | 97%      |
| celery/contrib/testing/manager.py | ManagerMixin.join      | 16         | 16      | 0        | 8          | 0       | 0%       |
| t/unit/tasks/test_chord.py        | TSR.join               | 5          | 0       | 0        | 6          | 1       | 91%      |
| t/unit/tasks/test_chord.py        | TSR_failed_join_report | 3          | 0       | 0        | 4          | 1       | 86%      |
| celery/utils/text.py              | join                   | 1          | 1       | 0        | 2          | 0       | 0%       |

```

82
83 @pytest.mark.parametrize('l, sep, expected', [
84     ('a', 'b', 'c'], '\n', 'a\nb\nc'),
85     ('a', 'b', 'c'], ', ', 'a, b, c'),
86     ('a', '', 'c'], '\n', 'a\nc'),
87     ('', '', ''), '\n', ''),
88     ([], '\n', ''),
89     ('', 'b', '', 'd'], '\n', 'b\nd'),
90     ('a'], '\n', 'a'),
91 ])
92
93 def test_join(l, sep, expected):
94     assert join(l, sep) == expected

```

```

jojo@DESKTOP-ROQAJDU: /mnt/c/Users/Jonat/Documents/GitHub/webtech-lab99/celery$ pytest -k "test_join" t/unit/utils/test_text.py
===== test session starts =====
platform linux -- Python 3.10.12, pytest-8.2.2, pluggy-1.5.0
rootdir: /mnt/c/Users/Jonat/Documents/GitHub/webtech-lab99/celery
configfile: pyproject.toml
plugins: celery-1.0.0, click-1.1.0, docker-tools-3.1.3, order-1.2.1, rerunfailures-14.0, subtests-0.12.1, timeout-2.3.1, cov-5.0.0
collected 20 items / 13 deselected / 7 selected

t/unit/utils/test_text.py .....
===== 7 passed, 13 deselected in 0.05s =====

```

<State the coverage improvement with a number and elaborate on why the coverage is improved>

0 → 100%, at first there were no tests covering whether the join() function would join the characters, where I added coverage on a list of characters, empty strings and strings including newlines.

<Test 2>

|                                      |                                 |   |   |   |   |   |    |
|--------------------------------------|---------------------------------|---|---|---|---|---|----|
| <a href="#">celery/utils/text.py</a> | <a href="#">fill_paragraphs</a> | 1 | 1 | 0 | 2 | 0 | 0% |
|--------------------------------------|---------------------------------|---|---|---|---|---|----|

<Provide the same kind of information provided for Test 1>

```

97 @pytest.mark.parametrize("s, width, sep, expected", [
98     ("Lorem ipsum dolor sit amet, consectetur adipiscing elit.", 20, "\n", "Lorem ipsum dolor\nsit amet,\nconsectetur\nadipiscing elit."),
99     ("Lorem ipsum dolor sit amet, consectetur adipiscing elit.", 15, "\n", "Lorem ipsum\ndolor sit amet,\nconsectetur\nadipiscing\nelit."),
100     ("First paragraph.\n\nSecond paragraph.", 30, "\n", "First paragraph.\n\nSecond paragraph."),
101     ("First paragraph.\n\nSecond paragraph.", 20, "\n", "First paragraph.\n\nSecond paragraph."),
102     ("", 10, "\n", ""),
103 ])
104 def test_fill_paragraphs(s, width, sep, expected):
105     assert fill_paragraphs(s, width, sep) == expected
106

```

```

jojo@DESKTOP-ROQAJDU: /mnt/c/Users/Jonat/Documents/GitHub/webtech-lab99/celery$ pytest -s -k "test_fill_paragraphs" t/unit/utils/test_text.py
===== test session starts =====
platform linux -- Python 3.10.12, pytest-8.2.2, pluggy-1.5.0
rootdir: /mnt/c/Users/Jonat/Documents/GitHub/webtech-lab99/celery
configfile: pyproject.toml
plugins: celery-1.0.0, click-1.1.0, docker-tools-3.1.3, order-1.2.1, rerunfailures-14.0, subtests-0.12.1, timeout-2.3.1, cov-5.0.0
collected 25 items / 20 deselected / 5 selected

t/unit/utils/test_text.py .....
===== 5 passed, 20 deselected in 0.05s =====

```

I created a test in order to assert the different possible situations, so lorem ipsum with differing widths, 10-30, newlines within the text and empty strings. This should cover the possible inputs.

### ### Overall

<Provide a screenshot of the old coverage results by running an existing tool (the same as you already showed above)>

<Provide a screenshot of the new coverage results by running the existing tool using all test modifications made by the group>

### ## Statement of individual contributions

<Write what each group member did>

Jonathan Davidson - Added coverage to join() from celery/utils/text.py and to fill\_paragraphs() from celery/utils/text.py which both introduced character based tests, which could be created by creating several test lines using pytest.mark.parametrize. Both had 0% coverage and ultimately had 100% coverage.

Buse Basavci - I worked on two specific functions: Inspect.\_prepare in celery/app/control.py and Worker.on\_start in celery/apps/worker.py. I added instrumentation to measure branch coverage, developed new test cases, and improved existing ones to ensure that at least 80% of the conditional branches in these functions are covered by tests. This process involved analysing the functions, identifying untested branches, and creating comprehensive tests to achieve the desired coverage.

Jacqueline Bouwman - Created a new test file (test\_termlogger.py) for the TermLogger function, and also added test cases for the test\_celery.py file, where I tested the CeleryOption.get\_default. 'Branch\_dictionary.py' and 'branch\_tracker.py' have been created as the test coverage tool and added instrumented code to 'multi.py' and 'base.py'. Both coverage for Celery and TermLogger went from 0% to 100%.

Rose Hollander: found the github project and made a fork, also did the line of code with lizard and made the initial coverage report, added tests for two functions (worker\_main and \_handle\_conf\_update).