Group 94:

Report for Assignment 1

Project chosen

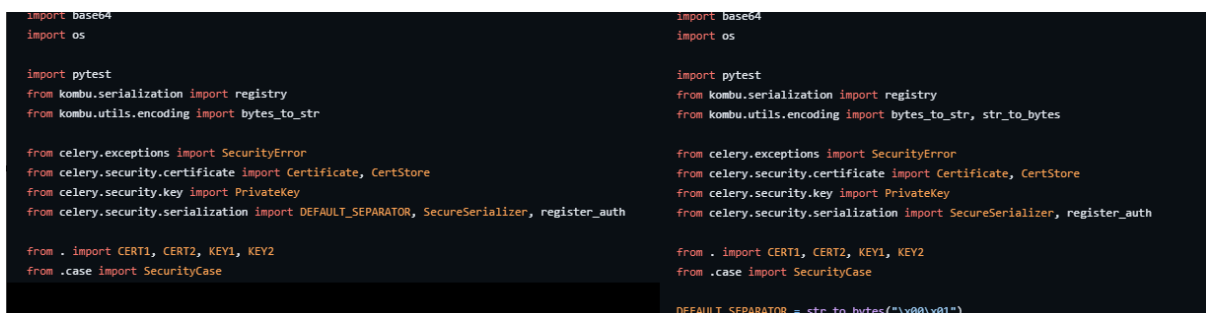Name: Celery

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.

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 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
```

```
r       -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 .........................................                                              [  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 ..............................................................s.............      [ 16%]

-- 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%)
```

Function 2: celery/app/base.py   Celery.worker_main

Left panel (base.py - original):

```
1      """Actual App instance implementation."""
2    - import inspect
3    - import os
4    - import sys
5    - import threading
6    - import warnings
7    - from collections import UserDict, defaultdict, deque
8    - from datetime import datetime
9    - from datetime import timezone as datetime_timezone
10   - from operator import attrgetter
11   -
12   - from click.exceptions import Exit
13   - from dateutil.parser import isoparse
14   - from kombu import pools
15   - from kombu.clocks import LamportClock
16   - from kombu.common import oid_from
17   - from kombu.utils.compat import register_after_fork
18   - from kombu.utils.objects import cached_property
19   - from kombu.utils.uuid import uuid
20   - from vine import starpromise
21   -
22   - from celery import platforms, signals
23   - from celery._state import (_announce_app_finalized, _deregister_app, _register_app, _set_current_app, _task_stack,
24   -                            connect_on_app_finalize, get_current_app, get_current_worker_task, set_default_app)
25   - from celery.exceptions import AlwaysEagerIgnored, ImproperlyConfigured
26   - from celery.loaders import get_loader_cls
27   - from celery.local import PromiseProxy, maybe_evaluate
28   - from celery.utils import abstract
29   - from celery.utils.collections import AttributeDictMixin
30   - from celery.utils.dispatch import Signal
31   - from celery.utils.functional import first, head_from_fun, maybe_list
32   - from celery.utils.imports import gen_task_name, instantiate, symbol_by_name
33   - from celery.utils.log import get_logger
34   - from celery.utils.objects import FallbackContext, mro_lookup
35   - from celery.utils.time import maybe_make_aware, timezone, to_utc
36   -
37   - # Load all builtin tasks
38   - from . import backends, builtins  # noqa
39   - from .annotations import prepare as prepare_annotations
40   - from .autoretry import add_autoretry_behaviour
41   - from .defaults import DEFAULT_SECURITY_DIGEST, find_deprecated_settings
42   - from .registry import TaskRegistry
43   - from .utils import (AppPickler, Settings, _new_key_to_old, _old_key_to_new, _unpickle_app, _unpickle_app_v2, appstr,
44   -                     bugreport, detect_settings)
45   -
46   - __all__ = ('Celery',)
```

Right panel (base.py - modified):

```
1      """Actual App instance implementation."""
2    + import inspect
3    + import os
4    + import sys
5    + import threading
6    + import warnings
7    + from collections import UserDict, defaultdict, deque
8    + from datetime import datetime
9    + from datetime import timezone as datetime_timezone
10   + from operator import attrgetter
11   +
12   + from click.exceptions import Exit
13   + from dateutil.parser import isoparse
14   + from kombu import pools
15   + from kombu.clocks import LamportClock
16   + from kombu.common import oid_from
17   + from kombu.utils.compat import register_after_fork
18   + from kombu.utils.objects import cached_property
19   + from kombu.utils.uuid import uuid
20   + from vine import starpromise
21   +
22   + from celery import platforms, signals
23   + from celery._state import (_announce_app_finalized, _deregister_app, _register_app, _set_current_app, _task_stack,
24   +                            connect_on_app_finalize, get_current_app, get_current_worker_task, set_default_app)
25   + from celery.exceptions import AlwaysEagerIgnored, ImproperlyConfigured
26   + from celery.loaders import get_loader_cls
27   + from celery.local import PromiseProxy, maybe_evaluate
28   + from celery.utils import abstract
29   + from celery.utils.collections import AttributeDictMixin
30   + from celery.utils.dispatch import Signal
31   + from celery.utils.functional import first, head_from_fun, maybe_list
32   + from celery.utils.imports import gen_task_name, instantiate, symbol_by_name
33   + from celery.utils.log import get_logger
34   + from celery.utils.objects import FallbackContext, mro_lookup
35   + from celery.utils.time import maybe_make_aware, timezone, to_utc
36   +
37   + # Load all builtin tasks
38   + from . import backends, builtins  # noqa
39   + from .annotations import prepare as prepare_annotations
40   + from .autoretry import add_autoretry_behaviour
41   + from .defaults import DEFAULT_SECURITY_DIGEST, find_deprecated_settings
42   + from .registry import TaskRegistry
43   + from .utils import (AppPickler, Settings, _new_key_to_old, _old_key_to_new, _unpickle_app, _unpickle_app_v2, appstr,
44   +                     bugreport, detect_settings)
45   +
46   + from branch_dictionary import branch_coverage
47   +
48   + __all__ = ('Celery',)
```

Lower-left panel (original worker_main):

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

Lower-right panel (modified worker_main):

```
385  +     def worker_main(self, argv=None):
386  +         """Run :program:`celery worker` using `argv`.
387  +
388  +         Uses :data:`sys.argv` if `argv` is not specified.
389  +         """
390  +         if argv is None:
391  +             branch_coverage["worker_main1"] = True
392  +             argv = sys.argv
393  +         else:
394  +             branch_coverage["worker_main2"] = True
395  +
396  +         if 'worker' not in argv:
397  +             branch_coverage["worker_main3"] = True
398  +             raise ValueError(
399  +                 "The worker sub-command must be specified in argv.\n"
400  +                 "Use app.start() to programmatically start other commands."
401  +             )
402  +         else:
403  +             branch_coverage["worker_main4"] = True
404  +
405  +         self.start(argv=argv)
406  +
```

Terminal output:

```
r       -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 .............................................................              [  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 ..........................................5...............           [ 16%]

-- 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%)
```

iwtwm completed branch coverage tracker

Code Blame   14 lines (12 loc) · 448 Bytes   Code 55% faster with GitHub Copilot

```python
branch_coverage = {
    "_handle_conf_update1": False,     # if 'task_routes' in kwargs or 'task_routes' in args
    "_handle_conf_update2": False,     # else branch

    "worker_main1": False,     # if argv is None
    "worker_main2": False,     # else branch
    "worker_main3": False,     # 'worker' not in argv
    "worker_main4": False      # else branch
}

branch_totals = {
    "_handle_conf_update": 2,
    "worker_main": 4
}
```

iwtwm completed branch coverage tracker

| Code | Blame | 41 lines (33 loc) · 1.28 KB | Code 55% faster with GitHub Copilot |

```python
import os
from branch_dictionary import branch_coverage, branch_totals

# Function to print branch coverage
def print_branch_coverage():
    print("\nBRANCH COVERAGE:\n")

    covered_branches = {
        "_handle_conf_update": 0,
        "worker_main": 0
    }

    print("Branch Hits:\n")
    s = ""
    for key, value in branch_coverage.items():
        s += f"{key} is covered: {value}\n"
    print(s)

    for key, value in branch_coverage.items():
        if value:
            if key.startswith("worker_main"):
                covered_branches["worker_main"] += 1
            elif key.startswith("_handle_conf_update"):
                covered_branches["_handle_conf_update"] += 1

    print("Branch Coverage Percentage:\n")
    for key, value in branch_totals.items():
        covered = covered_branches[key]
        coverage_percentage = (covered / value) * 100
        print(f"{key}: {covered} out of {value} branches covered ({coverage_percentage:.2f}%)")

# Function to run all tests and print coverage
def run_coverage():
    import pytest
    pytest.main([os.path.join('t', 'unit', 'app', 'test_amqp.py')])
    pytest.main([os.path.join('t', 'unit', 'app', 'test_app.py')])

    print_branch_coverage()

if __name__ == '__main__':
    run_coverage()
```

Buse Basavci
Function 1: celery/app/control.py   Inspect._prepare

```
@@ -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._prepare6"] = 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

```python
"""Worker command-line program.

This module is the 'program-version' of :mod:`celery.worker`.

It does everything necessary to run that module
as an actual application, like installing signal handlers,
platform tweaks, and so on.
"""
import logging
import os
import platform as _platform
import sys
from datetime import datetime
from functools import partial

from billiard.common import REMAP_SIGTERM
from billiard.process import current_process
from kombu.utils.encoding import safe_str

from celery import VERSION_BANNER, platforms, signals
from celery.app import trace
from celery.loaders.app import AppLoader
from celery.platforms import EX_FAILURE, EX_OK, check_privileges
from celery.utils import static, term
from celery.utils.debug import cry
from celery.utils.imports import qualname
from celery.utils.log import get_logger, in_sighandler, set_in_sighandler
from celery.utils.text import pluralize
from celery.worker import WorkController

from branch_dictionary import branch_coverage

__all__ = ('Worker',)

logger = get_logger(__name__)
is_jython = sys.platform.startswith('java')
is_pypy = hasattr(sys, 'pypy_version_info')
```

```python
#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%)
```

```python
@@ -0,0 +1,46 @@
import os
from branch_dictionary import branch_coverage, branch_totals

# Function to print branch coverage
def print_branch_coverage():
    print("\nBRANCH COVERAGE:\n")

    covered_branches = {
        "_handle_conf_update": 0,
        "worker_main": 0,
        "Inspect._prepare": 0,
        "Worker.on_start": 0
    }

    print("Branch Hits:\n")
    s = ""
    for key, value in branch_coverage.items():
        s += f"{key} is covered: {value}\n"
    print(s)

    for key, value in branch_coverage.items():
        if value:
            if key.startswith("worker_main"):
                covered_branches["worker_main"] += 1
            elif key.startswith("_handle_conf_update"):
                covered_branches["_handle_conf_update"] += 1
            elif key.startswith("Inspect._prepare"):
                covered_branches["Inspect._prepare"] += 1
            elif key.startswith("Worker.on_start"):
                covered_branches["Worker.on_start"] += 1

    print("Branch Coverage Percentage:\n")
    for key, value in branch_totals.items():
        covered = covered_branches[key]
        coverage_percentage = (covered / value) * 100
        print(f"{key}: {covered} out of {value} branches covered ({coverage_percentage:.2f}%)")

# Function to run all tests and print coverage
def run_coverage():
    import pytest
    pytest.main(['t/unit/apps/test_worker.py'])

    print_branch_coverage()

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

Function 2: CeleryOption.get_default

```python
branch_coverage = {
    "_handle_conf_update1": False,      # if 'task_routes' in kwargs or 'task_routes' in args
    "_handle_conf_update2": False,      # else branch

    "worker_main1": False,      # if argv is None
    "worker_main2": False,      # else branch
    "worker_main3": False,      # 'worker' not in argv
    "worker_main4": False,       # else branch

    "TermLogger.info1": False,
    "TermLogger.info2": False,

    "CeleryOption.get_default1": False,
    "CeleryOption.get_default2": False
}

branch_totals = {
    "_handle_conf_update": 2,
    "worker_main": 4,
    "TermLogger.info": 2,
    "CeleryOption.get_default": 2
}
```

```
v 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 na
me>
<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

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

```
@@ -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
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% |

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>

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
```

```
PROBLEMS    OUTPUT    TERMINAL    DEBUG CONSOLE    PORTS

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>

celery/utils/text.py	fill_paragraphs	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       |
```

```
PROBLEMS    OUTPUT    TERMINAL    DEBUG CONSOLE    PORTS

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)>

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	8	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_010	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/asynpool.py	AsynPool.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%

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

Coverage report: 84%

Files | Functions | Classes

coverage.py v7.5.3, created at 2024-06-27 23:52 +0200

File	statements ▲	missing	excluded	branches	partial	coverage
celery/apps/__init__.py	0	0	0	0	0	100%
celery/backends/__init__.py	0	0	0	0	0	100%
celery/bin/__init__.py	0	0	0	0	0	100%
celery/contrib/__init__.py	0	0	0	0	0	100%
celery/contrib/django/__init__.py	0	0	0	0	0	100%
celery/contrib/testing/__init__.py	0	0	0	0	0	100%
celery/fixups/__init__.py	0	0	0	0	0	100%
t/__init__.py	0	0	0	0	0	100%
t/integration/__init__.py	0	0	0	0	0	100%
t/smoke/__init__.py	0	0	0	0	0	100%
t/smoke/operations/__init__.py	0	0	0	0	0	100%
t/smoke/tests/__init__.py	0	0	0	0	0	100%
t/smoke/tests/failover/__init__.py	0	0	0	0	0	100%
t/smoke/tests/stamping/__init__.py	0	0	0	0	0	100%
t/smoke/tests/stamping/workers/__init__.py	0	0	0	0	0	100%
t/smoke/workers/__init__.py	0	0	0	0	0	100%
t/unit/__init__.py	0	0	0	0	0	100%
t/unit/app/__init__.py	0	0	0	0	0	100%
t/unit/apps/__init__.py	0	0	0	0	0	100%

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).