SUSE Hack Week 2024 - Day 1

Date:	2024-11-18
modified:	2024-11-18
tags:	SUSE, openSUSE, pagure, HackWeek, AWS, CodePipeline, Coding, OpenSource
description:	Experiences and outcome of my first day at SUSE Hack Week 2024
category:	Code
slug:	suse-hack-week-2024-day-1
Author:	Dominik Wombacher
lang:	en
transid:	suse-hack-week-2024-day-1
Status:	published

I started the work on my project AWS CodePipeline CI plugin for pagure on code.opensuse.org with my nemesis, the mysterious `Unable to find object` Exception that I couldn't solve in the past. This problem blocked my PR to Bump test containers to F40, bump pip version pinning to align with rpm versions, address some tech debts in dependencies.

One observation I've made last time was that the Database Session is rolled back and new created in the middle of the failing test. I started my troubleshooting there to understand which session was created and re-created during the test. And which one is actually used by the celery task that throws the exception.

Before the rollback (line 171 in test_pagure_flask_dump_load_ticket.py) `self.session` and `self.db_session` are not identical, afterwards they are. That was somehow suspicious, but after taking a closer look and playing around with the session and db_session value, it doesn't seem to make any difference and is a dead end.

Then I checked the error message and stack trace once more, the mock patch that injects a lambda function as override for `_maybe_wait` seem to cause the issues, a detail I didn't notice before:

```
tests/test_pagure_flask_dump_load_ticket.py:203:
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
pagure/lib/git.py:549: in update_ticket_from_git
    issue = pagure.lib.query.new_issue(
pagure/lib/query.py:1757: in new_issue
    pagure.lib.git.update_git(issue, repo=repo)
pagure/lib/git.py:165: in update_git
    _maybe_wait(queued)
/usr/lib/python3.12/site-packages/mock/mock.py:1100: in __call__
    return _mock_self._mock_call(*args, **kwargs)
/usr/lib/python3.12/site-packages/mock/mock.py:1104: in _mock_call
    return _mock_self._execute_mock_call(*args, **kwargs)
/usr/lib/python3.12/site-packages/mock/mock.py:1167: in _execute_mock_call
    result = effect(*args, **kwargs)
tests/test_pagure_flask_dump_load_ticket.py:43: in <lambda>
    mw.side_effect = lambda result: result.get()
/usr/lib/python3.12/site-packages/celery/result.py:1026: in get
    raise self.result if isinstance(
/usr/lib/python3.12/site-packages/celery/app/trace.py:477: in trace_task
    R = retval = fun(*args, **kwargs)
pagure/lib/tasks_utils.py:36: in decorated_function
    return function(self, session, *args, **kwargs)
```

I checked older logs, they confirm that this trace came up earlier already...

So what does `_maybe_wait` do? well ...

```
def _maybe_wait(result):
    """Function to patch if one wants to wait for finish.

    This function should only ever be overridden by a few tests that depend
    on counting and very precise timing."""
    pass
```

Seem to cover very few edge cases then?! How is it used by other tests?

```
tests/__init__.py:def create_maybe_waiter(method, getter):
tests/__init__.py:        self.app.get = create_maybe_waiter(self.app.get, self.app.get)
tests/__init__.py:        self.app.post = create_maybe_waiter(self.app.post, self.app.get)
tests/__init__.py:    """Helper function for definitely waiting in _maybe_wait."""
tests/test_pagure_flask_dump_load_ticket.py:    @patch("pagure.lib.git._maybe_wait")
tests/test_pagure_lib_drop_issue.py:    @patch("pagure.lib.git._maybe_wait", tests.definitely_wait)
tests/test_pagure_lib_drop_issue.py:    @patch("pagure.lib.git._maybe_wait", tests.definitely_wait)
tests/test_pagure_lib_git.py:        with patch("pagure.lib.git._maybe_wait", tests.definitely_wait):
```

Twice in tests/test_pagure_lib_drop_issue.py, once in tests/test_pagure_lib_git.py. Always with `tests.definitely_wait`, never in the way it's used in tests/test_pagure_flask_dump_load_ticket.py. A test with `@patch("pagure.lib.git._maybe_wait", tests.definitely_wait)` ended up in the same `Unable to find object` exception. After dropping the whole patch / mock, tests/test_pagure_flask_dump_load_ticket.py::PagureFlaskDumpLoadTicketTests::test_dumping_reloading_ticket is happy after all and passes.

I have to admit, I still don't fully understand why the test was implemented like that. My only explanation is, that some Celery behavior must have changed over time in a release. And after removing / bumping the version pinning and using Fedora 40 it came up as a problem.

Such tests are there to rely on, and if that one now passes without mocking, I'm fine with that.

After finally fixing this annoying problem, one of four Goals that I have for the Hack Week is completed.

Project Goals:

1. Resolve issues with outdated python dependencies to ensure pagure runs on Python 3.11 and current package versions. At least Fedora 40 python and python package versions. Fedora 41 would be even better. This will also satisfy openSUSE Tumbleweed and Leap 15.6 package versions used on code.opensuse.org
2. Validate if https://bugzilla.opensuse.org/show_bug.cgi?id=1229570 is fixed for code.opensuse.org and that all required packages / backports landed in openSUSE:infrastructure:pagure
3. Develop AWS CodePipeline pagure CI plugin and create an upstream pull request
4. Backport Plugin into openSUSE pagure package on OBS till new upstream release

While I was waiting for my pagure PR to be reviewed and merged, I worked on Goal 2. Based on the Bugzilla Ticket, the problem with the unsatisfied runtime dependency on cffi is solved for openSUSE:infrastructure:pagure. And still pending for openSUSE Leap 15.6 / SLES 15 SP6 in general. But for know I only care about pagure on code.opensuse.org. The OBS project had two broken packages that I fixed. Nothing else to do, Goal 2 completed.

In preparation for Day 2, I updated my pagure dev system to the latest master branch version, including the Patch from my pending PR. This allows me to develop the actual pagure CI plugin for AWS CodePipeline on the latest build. Avoids running into issues because of old code and refactoring to get it included into the pagure code base.

I'm pretty happy with the progress of the first Day, can't wait how things going tomorrow :)