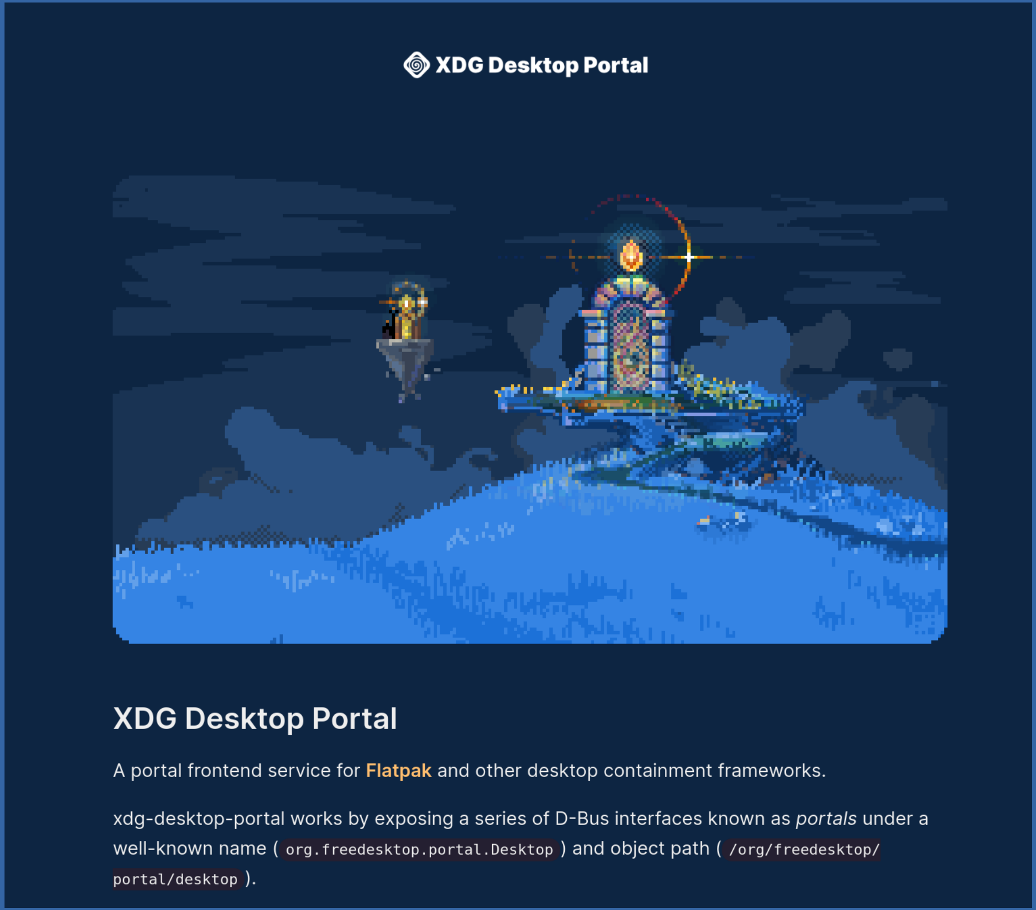
Testing
~~Thinking~~ with Portals

Sebastian Wick (swick)
fosstodon.org/@swick
sebastian.wick@redhat.com

LAS

What are Portals



XDG Desktop Portal

A portal frontend service for **Flatpak** and other desktop containment frameworks.

xdg-desktop-portal works by exposing a series of D-Bus interfaces known as *portals* under a well-known name (`org.freedesktop.portal.Desktop`) and object path (`/org/freedesktop/portal/desktop`).

LAS

What are Portals

- https://github.com/flatpak/xdg-desktop-portal
- https://flatpak.github.io/xdg-desktop-portal/docs/for-app-developers.html

What are Portals

- “XDG Desktop Portal is a session service that provides D-Bus interfaces for apps to interact with the desktop.”
- APIs are desktop agnostic
- APIs work for sandboxed and non-sandboxed applications
- Give users control over permissions

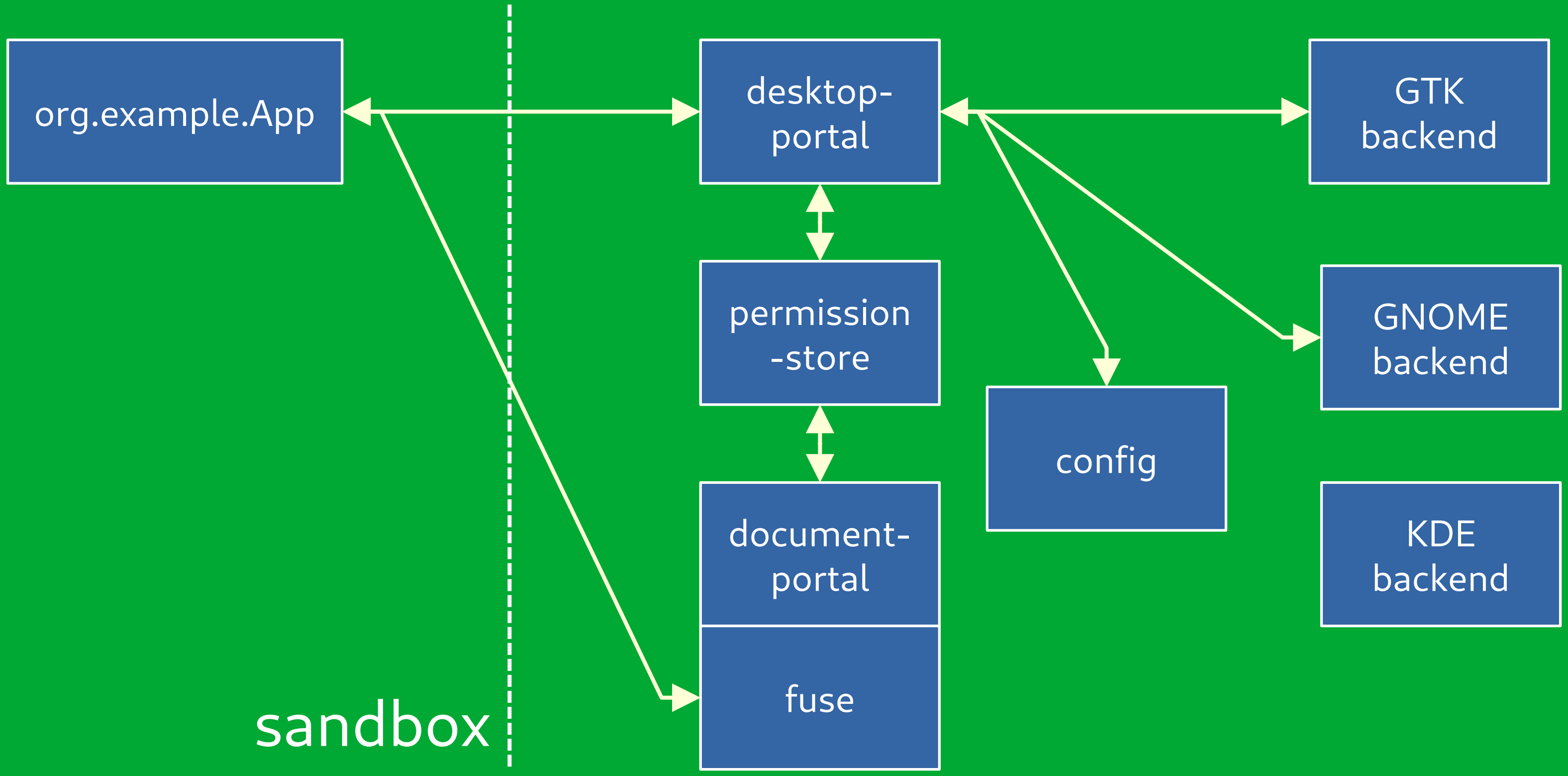
What are Portals

You Should Use Portals!

How Portals Work

- xdg-desktop-portal, xdg-permission-store, xdg-document-portal
- Exposing D-Bus interfaces and a fuse filesystem
- A subset of D-Bus interfaces are the actual API which is exposed to all apps
- xdg-desktop-portal authenticates the caller, validates input, checks and stores permissions, does agnostic work, calls into desktop-specific backends
- Desktop-specific backends show UI and do desktop-specific work

How Portals Work



LAS

How Portals Work

- Common conventions for the API:
  - Requests: method call creates a Request object, which eventually delivers the result in a signal
  - Sessions: method call creates a Session object; other method calls and Requests happen on the session

LAS

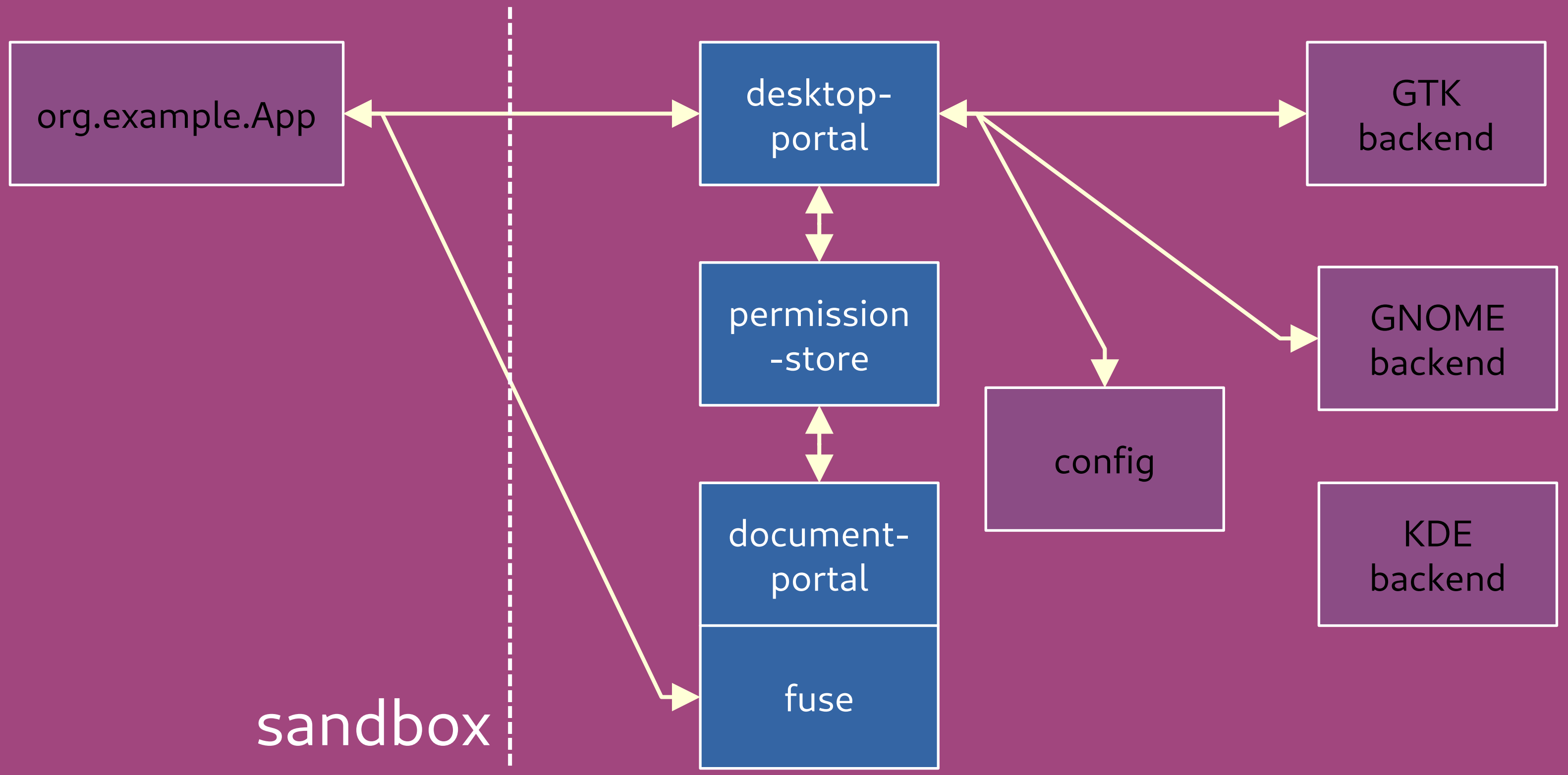
Testing: Where we Started

Great: There are existing tests!

Testing: Where we Started

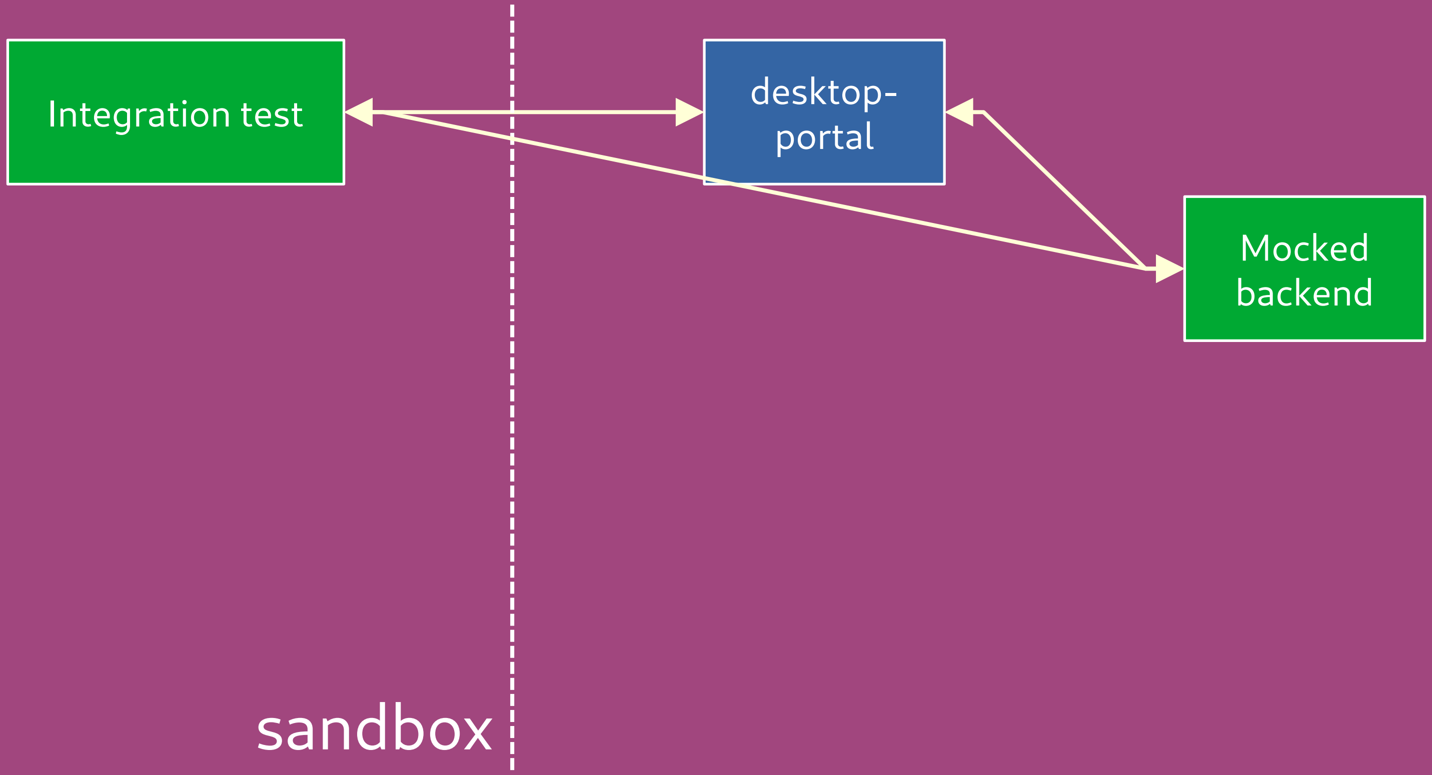
- GLib Testing Framework
  - Everything is written in C, so this makes some sense
- Two classes of tests
  - Unit tests: a few functions are tested in isolation
  - Integration tests
- Focusing on the integration tests (though it would be nice to expand on unit testing as well)

Testing: Where we Started



LAS

Testing: Where we Started



Testing: Where we Started

- Test spawns xdg-desktop-portal and a mocked backend
- Communication between backend and test via keyfiles (also known as desktop files, ini file)
- Backends and tests written in C
- Tests use libportal to talk to xdg-desktop-portal
  - New portals need tests, which needs support in libportal which needs the new portal (cycling dependency)
- CI was rotting away

LAS

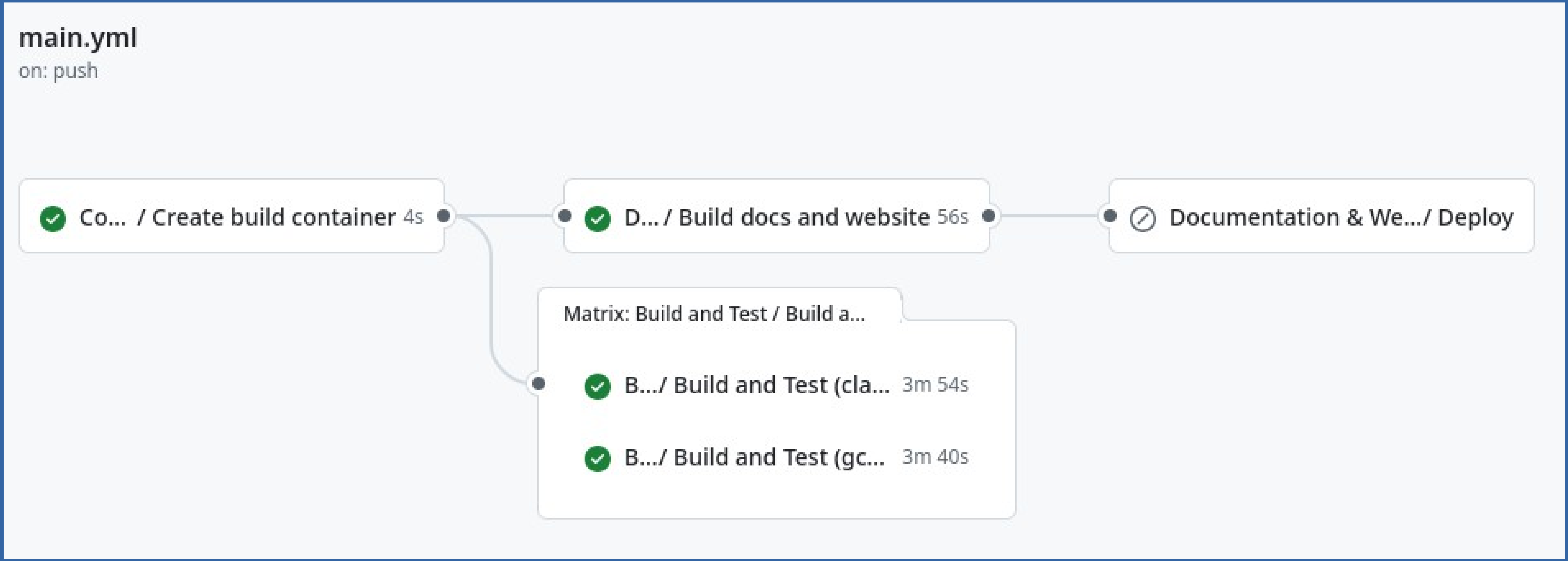
Testing: Where we Started

- Peter Hutterer started using python for some new integration tests
- pytest framework
- dbus-mock for mocking the backend
- Utils for interacting with the portal dbus API directly
- Existing C tests didn't get ported, new tests used C, limitations

Upgrading the CI

- You actually need to run the tests if they are of any use
- CI is perfect for this
- … but our CI is not in great shape
- Let’s improve it!

Upgrading the CI


main.yml
on: push
Co... / Create build container 4s
D... / Build docs and website 56s
Documentation & We.../ Deploy
Matrix: Build and Test / Build a...
B.../ Build and Test (cla... 3m 54s
B.../ Build and Test (gc... 3m 40s

LAS

Upgrading the CI

- GitHub workflows
- Creating an image on-demand, re-using existing one if it exists (similar to fdo and GNOME CI)
- Containerfile to create the image from
- Building, running linters, running tests, creating a release tarball
- Building docs and website
- Workflow for releases (Georges improved this to work via tags)

LAS

Python Integration Tests

- Isolate the tests from the system more
  - Create tempdirs for *XDG_HOME*, *XDG_DATA_DIR*, …
- Make it possible to run xdg-document-portal, xdg-permission-store
- Run address sanitizers on the DUTs to catch memory leaks
- Allow mocking multiple backends and services
- Improve the overall structure of the tests via pytest fixtures
- Make the tests more flexible

LAS

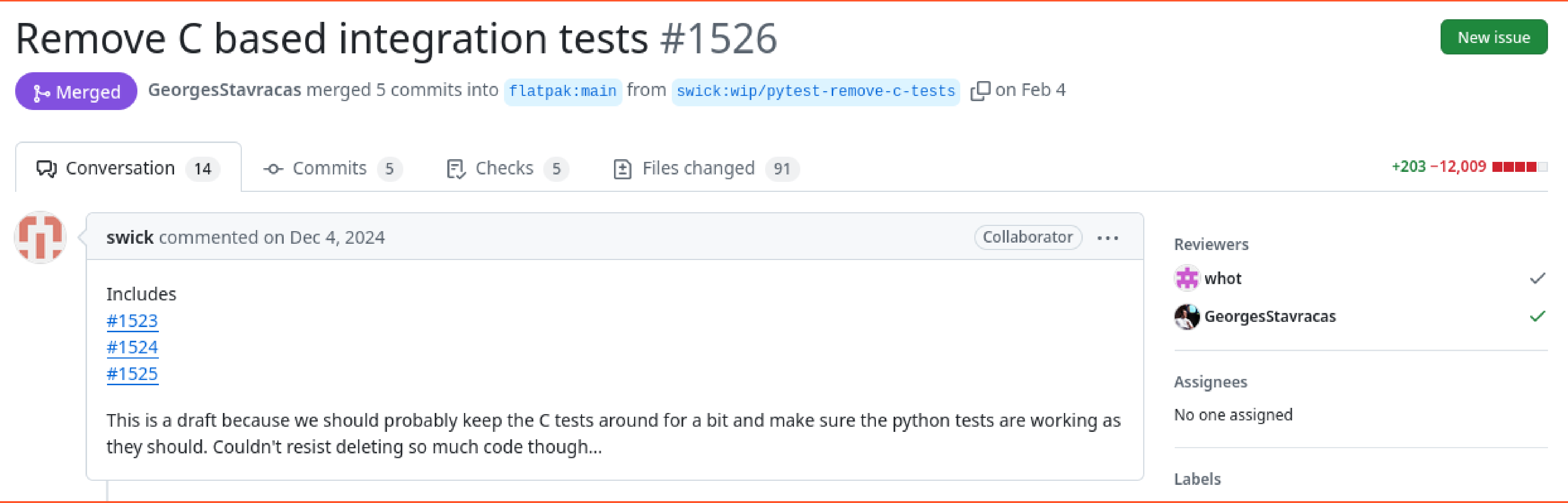
Python Integration Tests

- Support for umockdev to mock devices (used by the USB portal)
- Allow tests to configure xdg-desktop-portal (used e.g. by the Settings portal)
- Allow tests to set as which app and app kind (flatpak, snap, host) they are detected
- Improved testing utils and replaced timeout-based tests with waiting for conditions (eliminates races)

Python Integration Tests

- Port over existing C integration tests to the new pytest harness
  - Some integration tests do not test the portal directly but e.g. the document portal and permission store
  - A test for the document-store fuse was written in python; ported to the new harness
  - Some APIs use complex data structures which are hard to use with dbus-python
- Drop the C integration tests

Python Integration Tests



Remove C based integration tests #1526

New issue

Merged GeorgesStavracas merged 5 commits into `flatpak:main` from `swick:wip/pytest-remove-c-tests` on Feb 4

Conversation 14 · Commits 5 · Checks 5 · Files changed 91 · +203 −12,009

swick commented on Dec 4, 2024 · Collaborator

Includes
#1523
#1524
#1525

This is a draft because we should probably keep the C tests around for a bit and make sure the python tests are working as they should. Couldn't resist deleting so much code though...

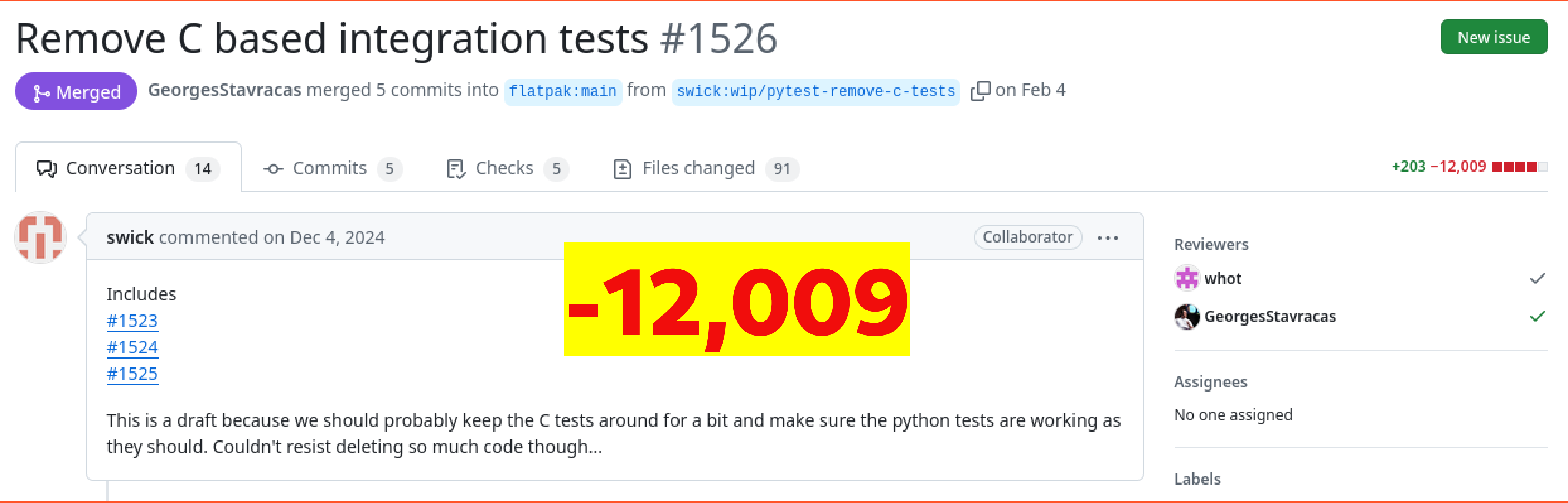
Reviewers
whot ✓
GeorgesStavracas ✓

Assignees
No one assigned

Labels

LAS

Python Integration Tests



Remove C based integration tests #1526

New issue

Merged GeorgesStavracas merged 5 commits into `flatpak:main` from `swick:wip/pytest-remove-c-tests` on Feb 4

Conversation 14 | Commits 5 | Checks 5 | Files changed 91 | +203 −12,009

swick commented on Dec 4, 2024 — Collaborator

Includes
#1523
#1524
#1525

This is a draft because we should probably keep the C tests around for a bit and make sure the python tests are working as they should. Couldn't resist deleting so much code though...

-12,009

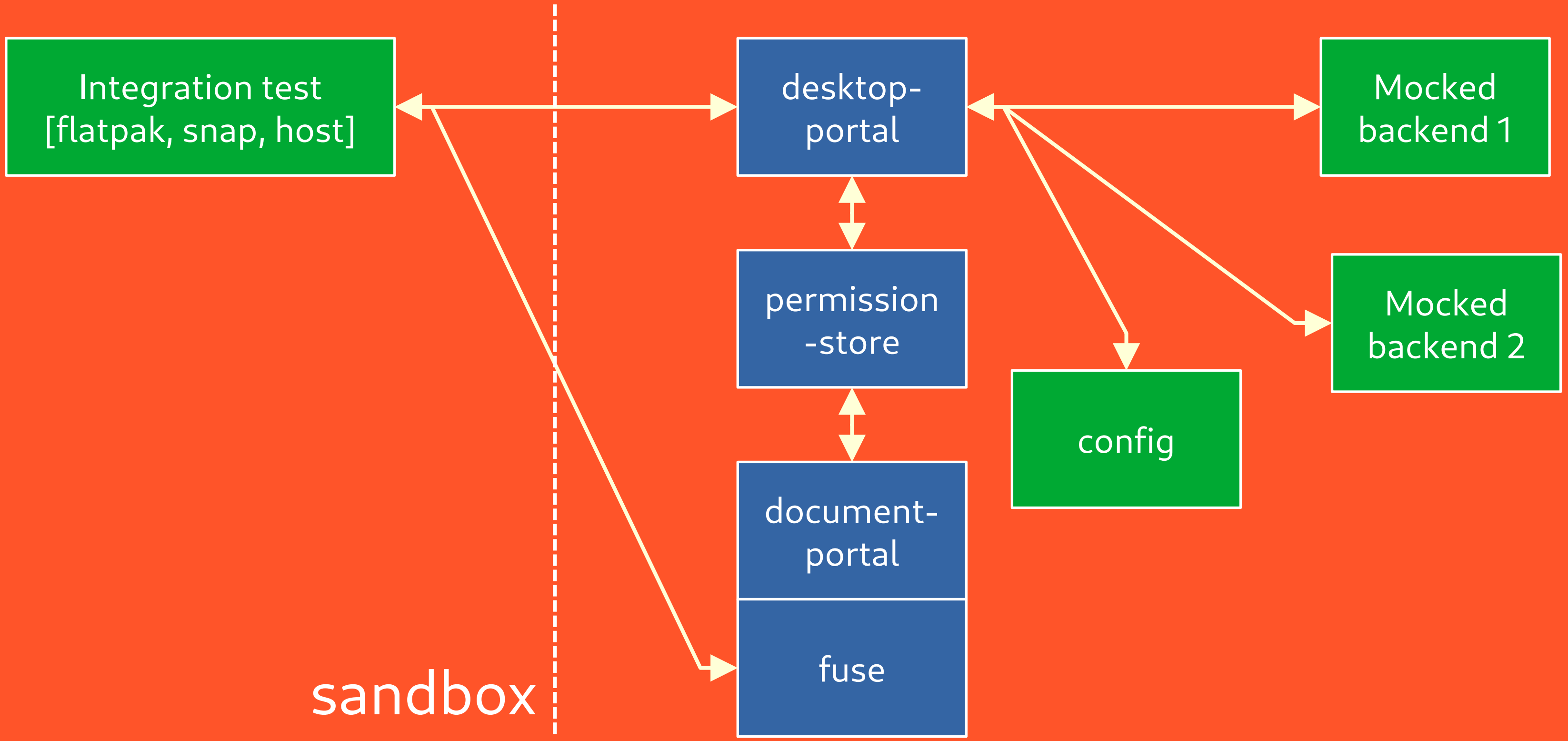
Reviewers
whot ✓
GeorgesStavracas ✓

Assignees
No one assigned

Labels

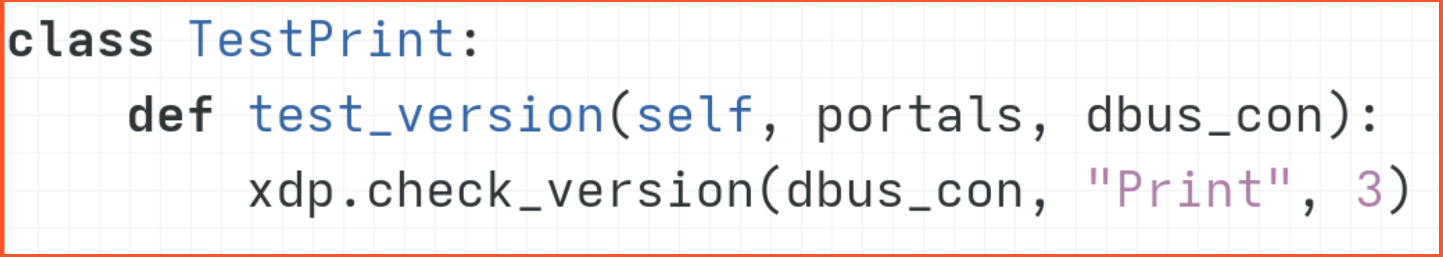
LAS

Python Integration Tests



LAS

Python Integration Tests

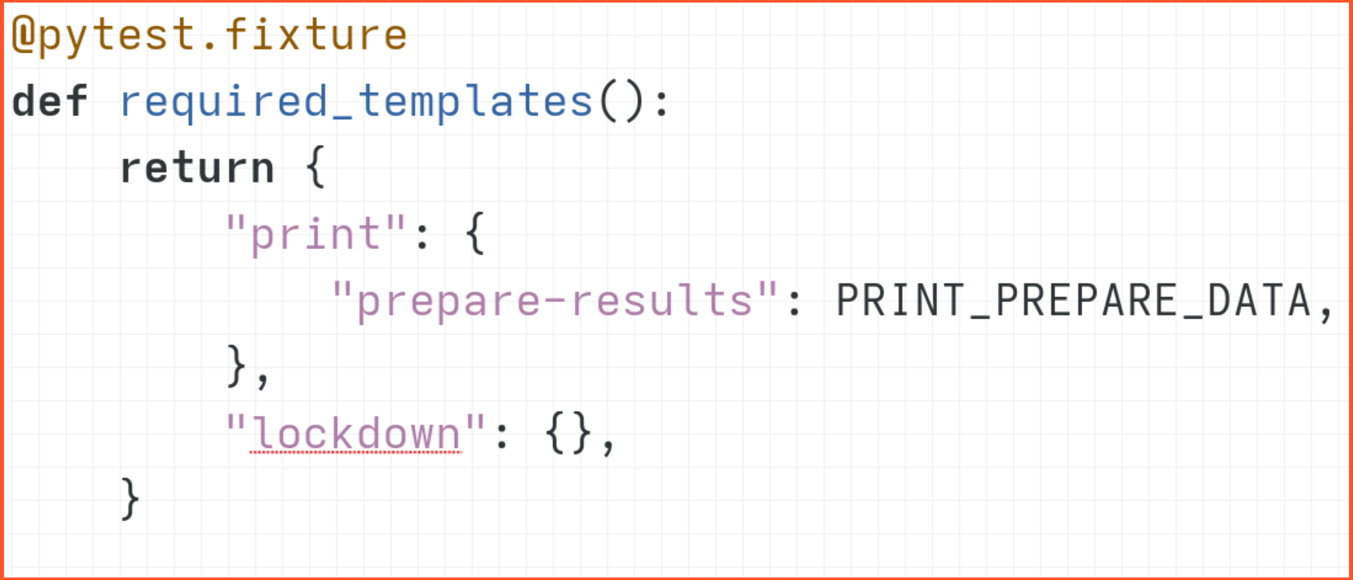


```python
class TestPrint:
    def test_version(self, portals, dbus_con):
        xdp.check_version(dbus_con, "Print", 3)
```

Python Integration Tests

```python
request = xdp.Request(dbus_con, print_intf)
response = request.call(
    "PreparePrint",
    parent_window="",
    title=title,
    settings=settings,
    page_setup=page_setup,
    options=options,
)

assert response
assert response.response == 0

# Check the impl portal was called with the right args
method_calls = mock_intf.GetMethodCalls("PreparePrint")
assert len(method_calls) == 1
_, args = method_calls.pop()
```

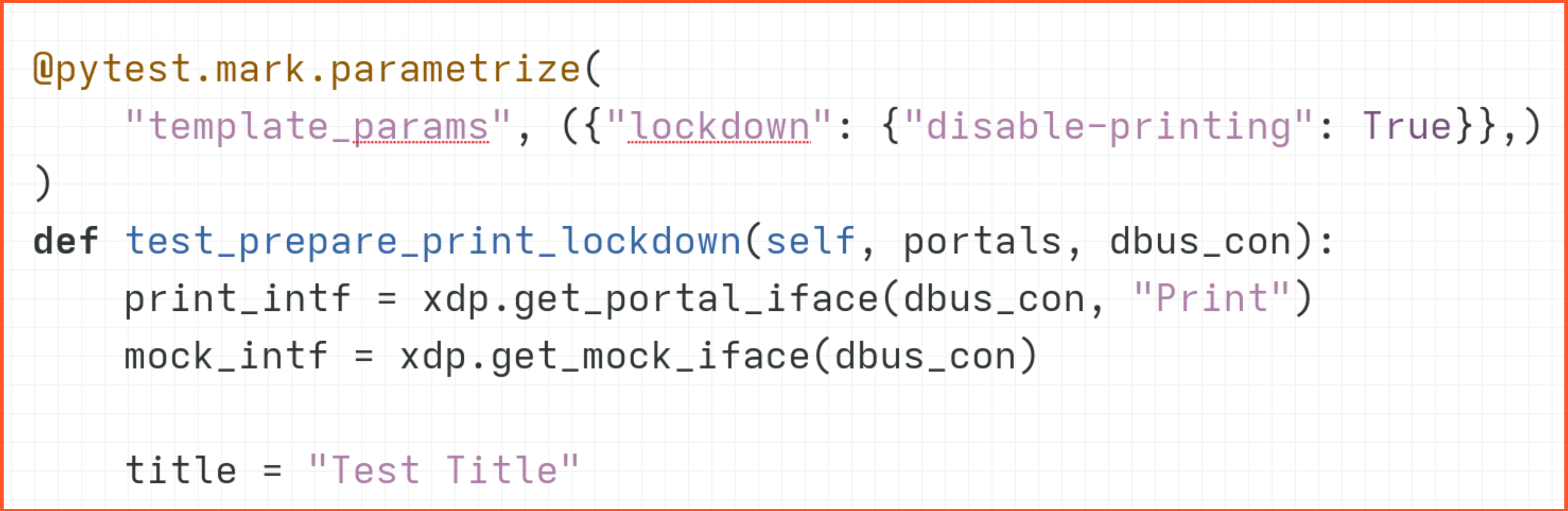
Python Integration Tests



```python
@pytest.fixture
def required_templates():
    return {
        "print": {
            "prepare-results": PRINT_PREPARE_DATA,
        },
        "lockdown": {},
    }
```

LAS

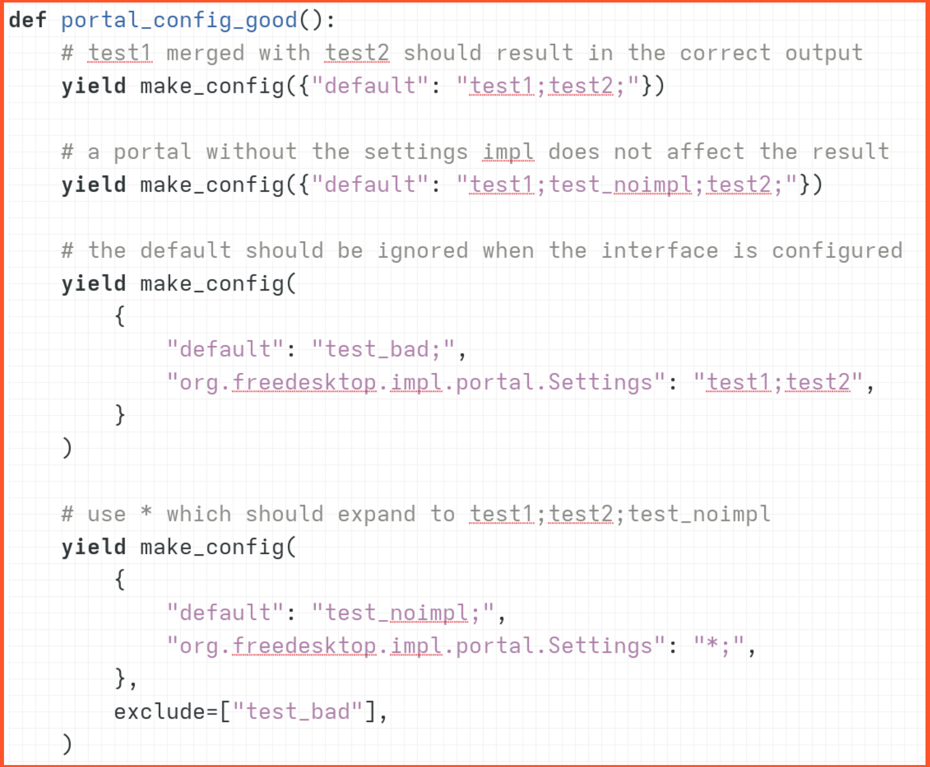
Python Integration Tests



```python
@pytest.mark.parametrize(
    "template_params", ({"lockdown": {"disable-printing": True}},)
)
def test_prepare_print_lockdown(self, portals, dbus_con):
    print_intf = xdp.get_portal_iface(dbus_con, "Print")
    mock_intf = xdp.get_mock_iface(dbus_con)

    title = "Test Title"
```

LAS

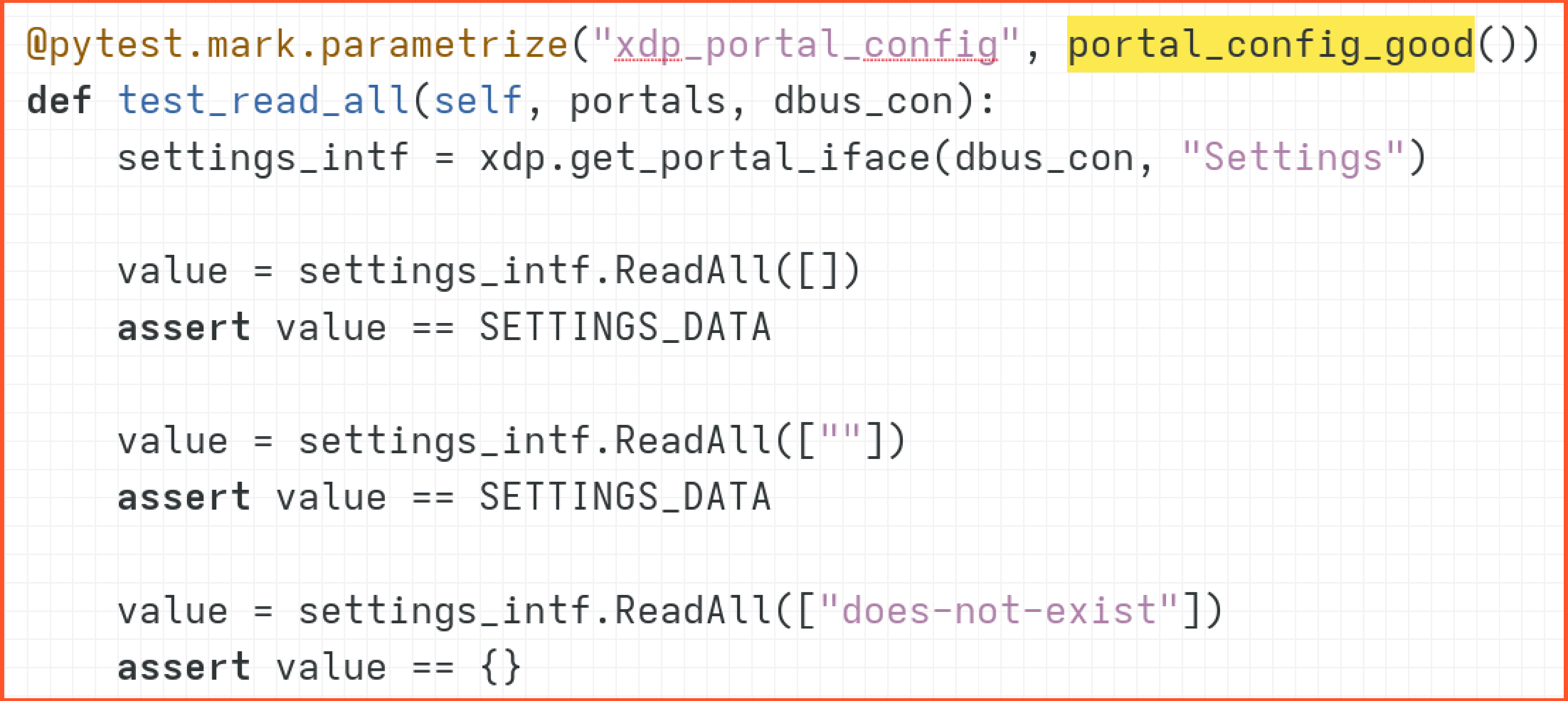
Python Integration Tests



```python
def portal_config_good():
    # test1 merged with test2 should result in the correct output
    yield make_config({"default": "test1;test2;"})

    # a portal without the settings impl does not affect the result
    yield make_config({"default": "test1;test_noimpl;test2;"})

    # the default should be ignored when the interface is configured
    yield make_config(
        {
            "default": "test_bad;",
            "org.freedesktop.impl.portal.Settings": "test1;test2",
        }
    )

    # use * which should expand to test1;test2;test_noimpl
    yield make_config(
        {
            "default": "test_noimpl;",
            "org.freedesktop.impl.portal.Settings": "*;",
        },
        exclude=["test_bad"],
    )
```

LAS

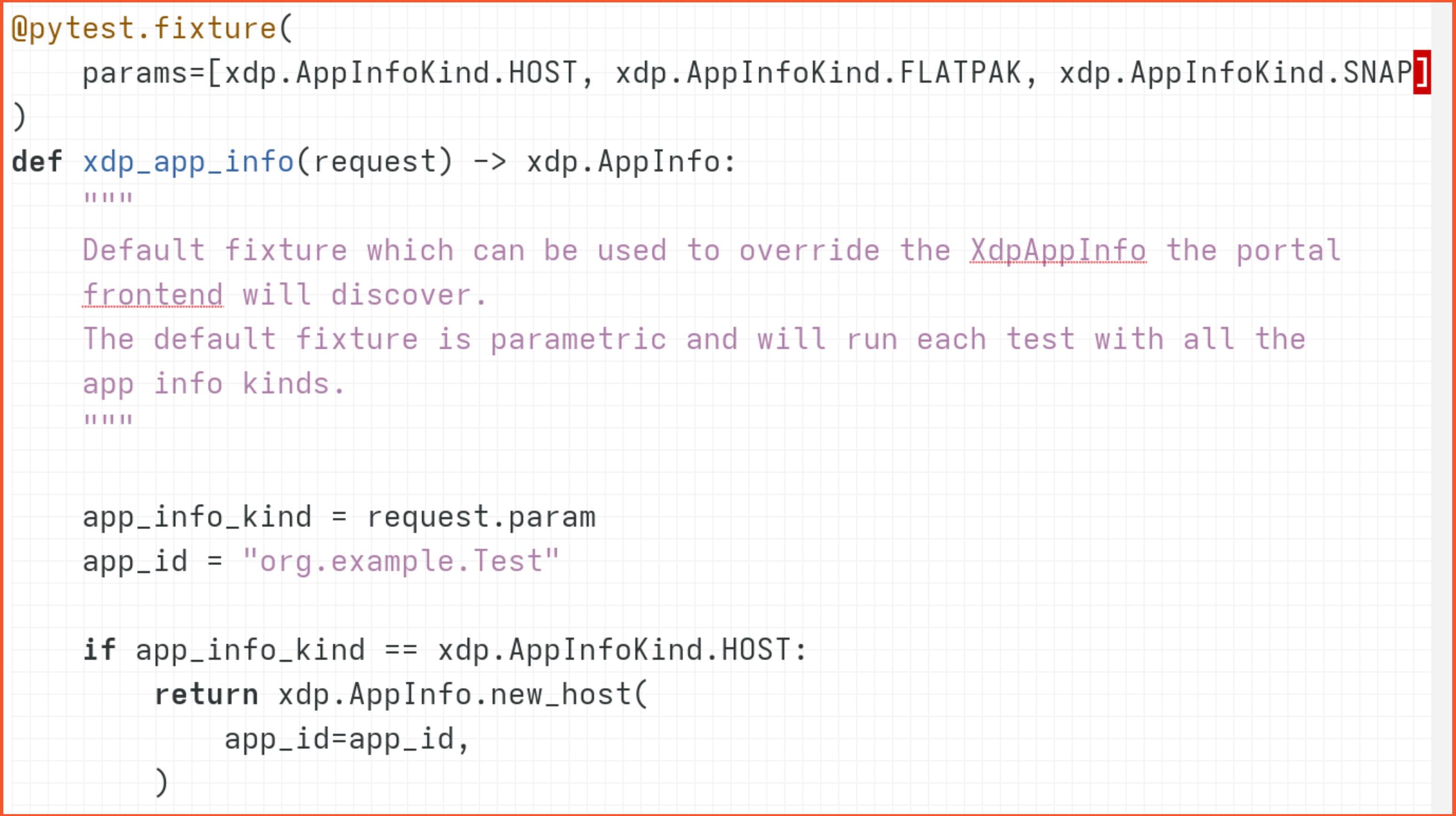
Python Integration Tests



```python
@pytest.mark.parametrize("xdp_portal_config", portal_config_good())
def test_read_all(self, portals, dbus_con):
    settings_intf = xdp.get_portal_iface(dbus_con, "Settings")

    value = settings_intf.ReadAll([])
    assert value == SETTINGS_DATA

    value = settings_intf.ReadAll([""])
    assert value == SETTINGS_DATA

    value = settings_intf.ReadAll(["does-not-exist"])
    assert value == {}
```

LAS

Python Integration Tests



```python
@pytest.fixture(
    params=[xdp.AppInfoKind.HOST, xdp.AppInfoKind.FLATPAK, xdp.AppInfoKind.SNAP]
)
def xdp_app_info(request) -> xdp.AppInfo:
    """
    Default fixture which can be used to override the XdpAppInfo the portal
    frontend will discover.
    The default fixture is parametric and will run each test with all the
    app info kinds.
    """


    app_info_kind = request.param
    app_id = "org.example.Test"

    if app_info_kind == xdp.AppInfoKind.HOST:
        return xdp.AppInfo.new_host(
            app_id=app_id,
        )
```

LAS

Sebastian Wick (swick)
fosstodon.org/@swick
sebastian.wick@redhat.com