Move Fast and Break Everything
Testing major changes to a core component of GNOME

Sam Thursfield
GUADEC 2020
Move Fast and Break Everything

Part 1. Get to know your daemons
Why does GNOME provide system services?

Part 2. Learn to control them
Testing changes to the Tracker search engine
Part 1. Get to know your daemons

Choose a Project

GNOME has got hundreds of projects. To make it easier for you to get started, we have highlighted the applications which are great starting points for making your first contribution.

**Polari (#polari)**
An easy to use IRC client, written in Javascript
**Project complexity: Simple**
**Code:** [https://gitlab.gnome.org/GNOME/polari.git](https://gitlab.gnome.org/GNOME/polari.git)
**Mentors:** Bastian Ilse (bastianilse), Florian Müllner (fmuellner)

**Games (#gnome-games)**
Game manager for your retro and Steam games, written in Vala
**Project complexity: Medium**
**Code:** [https://gitlab.gnome.org/GNOME/gnome-games.git](https://gitlab.gnome.org/GNOME/gnome-games.git)
**Mentors:** Alexander Mikhaylenko (exalm)

**Maps (#gnome-maps)**
A simple map application, written in Javascript.
**Project complexity: Simple**
**Code:** [https://gitlab.gnome.org/GNOME/gnome-maps](https://gitlab.gnome.org/GNOME/gnome-maps)
**Mentors:** Jonas Danielsson (jonasdn), Marcus Lundblad (marcus), Amisha Singla (amisha)
What is it?

Part 1. Get to know your daemons
Part 1. Get to know your daemons
Part 1. Get to know your daemons
System services

- May manage hardware, or a data store
- Communicate over D-Bus, or by a socket
- Apps access them directly or with a helper library

“Anything that wants to be running as a system service in combination with any kind of sandboxing system must have a protocol that is ABI stable and backwards compatible.”

Part 1. Get to know your daemons
Try this at home: pstree

Part 1. Get to know your daemons
Part 2: Control your daemons
...or better yet, hack on them.
Part 2. Control your daemons
Part 2. Control your daemons

Building and running Tracker with GNOME Builder

Open a File or Terminal

Use the page switcher above or use one of the following:

- Search
- Project sidebar
- File chooser
- New terminal

Application started at 23:43:22
usage: tracker3 [--version] [--help]
<command> [<args>]

Available tracker3 commands are:
- help: Get help on how to use Tracker and any of these commands
- endpoint: Create a SPARQL endpoint
- export: Export data from a Tracker database
- import: Import data into a Tracker database
- info: Show information known about local files or items indexed
- sparql: Query and update the index using SPARQL or search, list and tree the ontology
- sql: Query the database at the lowest level using SQL


Additional / third party commands are:
- daemon
- endpoint
- export
- extract
- help
Running an automated test

Copy a file between monitored directories ... ok
test 05 move from unmonitored to monitored (__main__.MinerCrawlTest)
Move a file from unmonitored to monitored directory ... skipped 'https://gitlab.gnome.org/GNOME/tracker-miners/issues/56'
test 06 move from monitored to unmonitored (__main__.MinerCrawlTest)
Move a file from monitored to unmonitored directory ... ok
test 07 move from monitored to monitored (__main__.MinerCrawlTest)
Move a file between monitored directories ... ok
test_08_deletion_single_file (__main__.MinerCrawlTest)
Delete one of the files ... ok
test_09_deletion_directory (__main__.MinerCrawlTest)
Delete a directory ... ok
test_10_folder_update (__main__.MinerCrawlTest)
Check that updating a folder updates nfo:belongsToContainer on its children ... ok

Ran 10 tests in 61.110s

OK (skipped=1)
Automated testing of daemons...

- Use `dbus-run-session` to create a throwaway message bus.
- Use `umockdev` to simulate real hardware.
- Write tests in Python (test your bindings and/or D-Bus API for free!)

See Tracker’s `tests/functional-tests` for a complex example.
Part 2. Control your daemons

NO MANUAL TESTING NECESSARY YOU SAY?

JUST CHECKING...
The best way to deploy a test build?

- Run from source tree
- Install into /usr
- Install into /opt (jhbuild)
- Use distro packaging tools
- Use BuildStream to build a VM image

Part 2. Control your daemons
Run from the source tree

- Unlikely to work, but try it!
- Project can provide a helper script.

https://gitlab.gnome.org/GNOME/tracker-miners/-/blob/master/run-uninstalled.in
Install into /usr

• Fast, easy and simple
• It will break the host OS.
Install into /opt

- Fast, and safe-ish
- System integration won’t work as normal.

```
export XDG_DATA_DIRS=/opt/tracker3/share:/usr/share
dbus-run-session /opt/tracker3/bin/tracker3 search Foo
```

- jhbuild can help, but you might not need it.
Use distro packaging tools

- Reproducible and safe-ish
- Use PPA for Ubuntu, COPR for Fedora, ...
- Turnaround time is slow (at least a few minutes)
Use BuildStream

- Reproducible and safe
- You need a VM (bst shell isn’t magic)
- Building and deploying VM update took me ~20 mins
- Work is ongoing to improve “GnomeOS” testing images (see Valentin’s talk)

More details:
## The best way to deploy a test build?

<table>
<thead>
<tr>
<th>Method</th>
<th>Fast</th>
<th>Reproducible</th>
<th>No extra codepaths needed</th>
<th>No extra computer or VM needed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run from source tree</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Install into /usr</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Install into /opt (jhbuild)</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Use distro packaging tools</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>~</td>
</tr>
<tr>
<td>Use BuildStream to build a VM image</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Part 2. Control your daemons
In summary...

• Contributing to daemons shouldn’t be scary.
  – *Do you maintain a daemon? Update the README :)*

• Automated testing is best
  – *If a service project doesn’t have functional testing, look at how to add it!*

• Manual testing is often needed too.
  – *How can we make sure it’s frictionless?*