

Move Fast and Break Everything

Sam Thursfield



Move Fast and Break Everything

Part 1. Get to know your daemons

Part 2. Learn to control them





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>

Mentors: [Bastian Ilso](#) ([bastianilso](#)), [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>

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>

Mentors: [Jonas Danielsson](#) ([jonasdn](#)), [Marcus Lundblad](#) ([marcus](#)), [Amisha Singla](#) ([amisha](#))



Part 1. Get to know your daemons



Apps

Shell

Part 1. Get to know your daemons



Apps

Shell



Part 1. Get to know your daemons



Apps

Shell

Services

Libraries

-

-

-

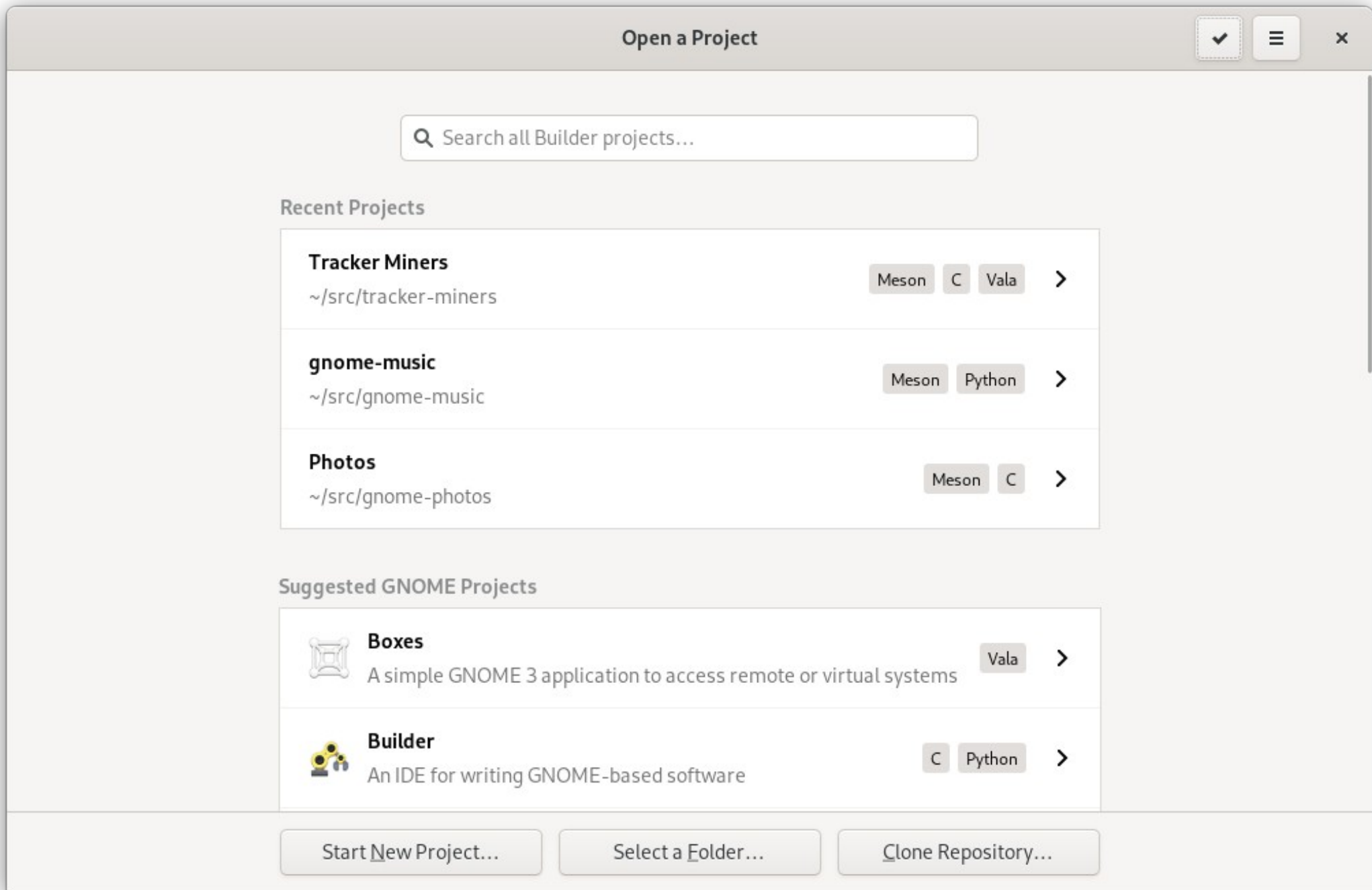


pstree

```
systemd—ModemManager—2*[{ModemManager}]
         —NetworkManager—2*[{NetworkManager}]
         —abrt-dbus—2*[{abrt-dbus}]
         —3*[abrt-dump-journ]
         —abrt-d—2*[{abrt-d}]
         —accounts-daemon—2*[{accounts-daemon}]
         —alsactl
         —auditd—sedispatch
                  —2*[{auditd}]
         —avahi-daemon—avahi-daemon
         —colord—2*[{colord}]
         —crond
         —cupsd
         —dbus-broker-lau—dbus-broker
         —2*[dnsmasq—dnsmasq]
         —earlyoom
         —firewalld—{firewalld}
```

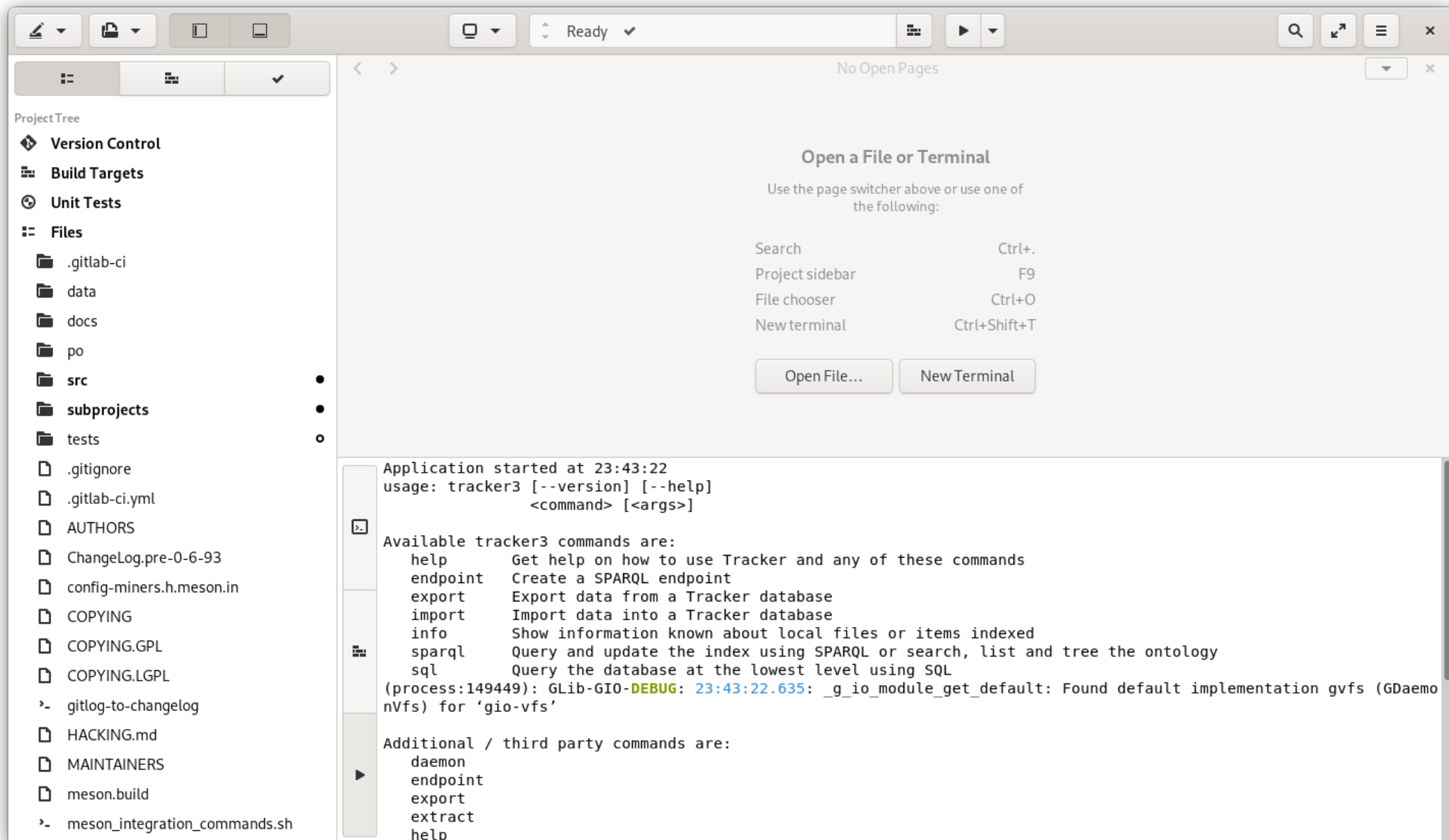
Part 2: Control your daemons





Part 2. Control your daemons

Building and running Tracker with GNOME Builder



Part 2. Control your daemons

Running an automated test

The screenshot shows a web browser window with the title "Running test 'miner-basic'...". The browser has a toolbar with icons for back, forward, and search. The main content area displays "No Open Pages" and a message "Open a File or Terminal" with instructions to use the page switcher or one of the following shortcuts: Search (Ctrl+.), Project sidebar (F9), File chooser (Ctrl+O), and New terminal (Ctrl+Shift+T). Below this, there are two buttons: "Open File..." and "New Terminal".

On the left side, there is a "Project Tree" sidebar. It lists several folders and files under the "miner-basic" project:

- tracker-miners:miner
- tracker-miners:extract
- tracker-miners:extractor
- tracker-miners:functional
- miner-basic** (selected)
- miner-resource-removal
- fts-basic
- fts-file-operations
- fts-stopwords
- extractor-decorator
- extractor-flac-cuesheet
- writeback-images
- writeback-audio

Below the "Project Tree", there is a "Files" section showing the file structure:

- .gitlab-ci
- data
- docs
- po
- src
- subprojects
- tests
- .gitignore
- .gitlab-ci.yml
- AUTHORS

The main content area also displays a list of test results:

```
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

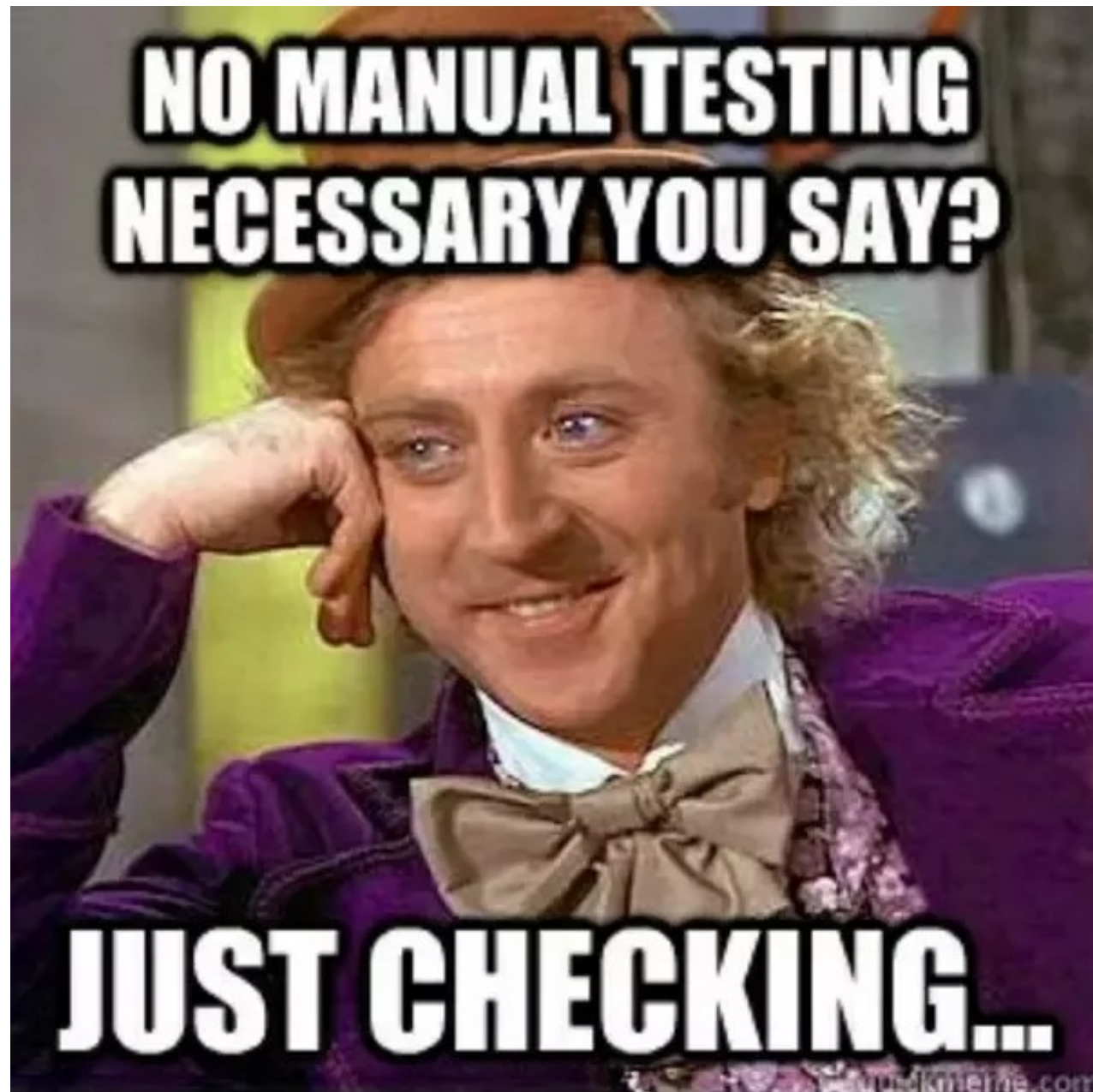
OK (skipped=1)
```

Part 2. Control your daemons

- `dbus-run-session`
- `umockdev`
-

`tests/functional-tests`





The best way to deploy a test build?

Run from source tree

Install into /usr

Install into /opt

Use distro packaging tools

Use BuildStream to build a VM image



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>



-

-



Part 2. Control your daemons

-
-

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

- `jhbuild`







-
-
-
-

bst shell

(see Valentin's talk)

The best way to deploy a test build?

	Fast	Reproducible	No extra codepaths needed	No extra computer or VM needed
Run from source tree	✓	✓		✓
Install into /usr	✓		✓	
Install into /opt	✓			✓
Use distro packaging tools		✓	✓	
Use BuildStream to build a VM image		✓	✓	



In summary...

- - *Do you maintain a daemon? Update the README :)*
- - *If a service project doesn't have functional testing, look at how to add it!*
- - *How can we make sure it's frictionless?*

