Flatpak workshop

flatpak.org

Carlos Soriano Sanchez - GNOME Developer

csoriano

csoriano@gnome.org

Felipe Borges - GNOME Developer

feborges

felipeborges@gnome.org

Overview

- Flatpak introduction
  - What is Flatpak
  - Features
  - Docs & tooling
  - Sdk and runtimes
  - Security & sandboxing
  - Create a simple app
- Qt & Portals
  - Qt flatpaking
  - Portals
  - Docs
  - Qt + Flatpak Tooling
- Hacking Gtk+ app & distributing in Flathub
  - Hacking with a Gtk+ app
  - Debugging
  - Common tricks
  - Distribute your app

Before we start - prep

Install Flatpak and Flathub - https://flatpak.org/setup/

Install GNOME Builder:

```
$ flatpak install flathub org.gnome.Builder
```

Builder -> clone -> gitlab.gnome.org/csoriano/baseapp

What is Flatpak

Build orchestration

Distribute

Sandbox

Features of Flatpak - Developers

Cross distro

Reproducible builds

Develop against a specific stack version

Isolated from your system and user’s system

Security by default - Apps are sandboxed

Permissions handling system built-in

Features of Flatpak - Developers

Signed distribution & updates

Not tied to a single source of distribution

Delta updates

Install multiple versions of the same app

Well defined processes (e.g. resources consumption)

Well known container & kernel technologies

Technology

OStree

Bubblewrap & CGroups

D-Bus

OCI format

Systemd

AppStream

Technology - OSTree

Git alike management

Delta upgrades

Multiple versions per app

Reproducible builds

Branch ↔ app & AppStream branch ↔ all app branches

OCI format

Technology - Bubblewrap & cgroups

Well defined processes

Sandboxing

Resource management

System isolation

Technology - D-Bus

Communication sandbox ↔ system

Portals - permission access (i.e. files, other apps, etc.)

Tooling

- Flatpak CLI
- Flatpak builder
- Flatpak manifest
- GNOME Builder
- GNOME Software

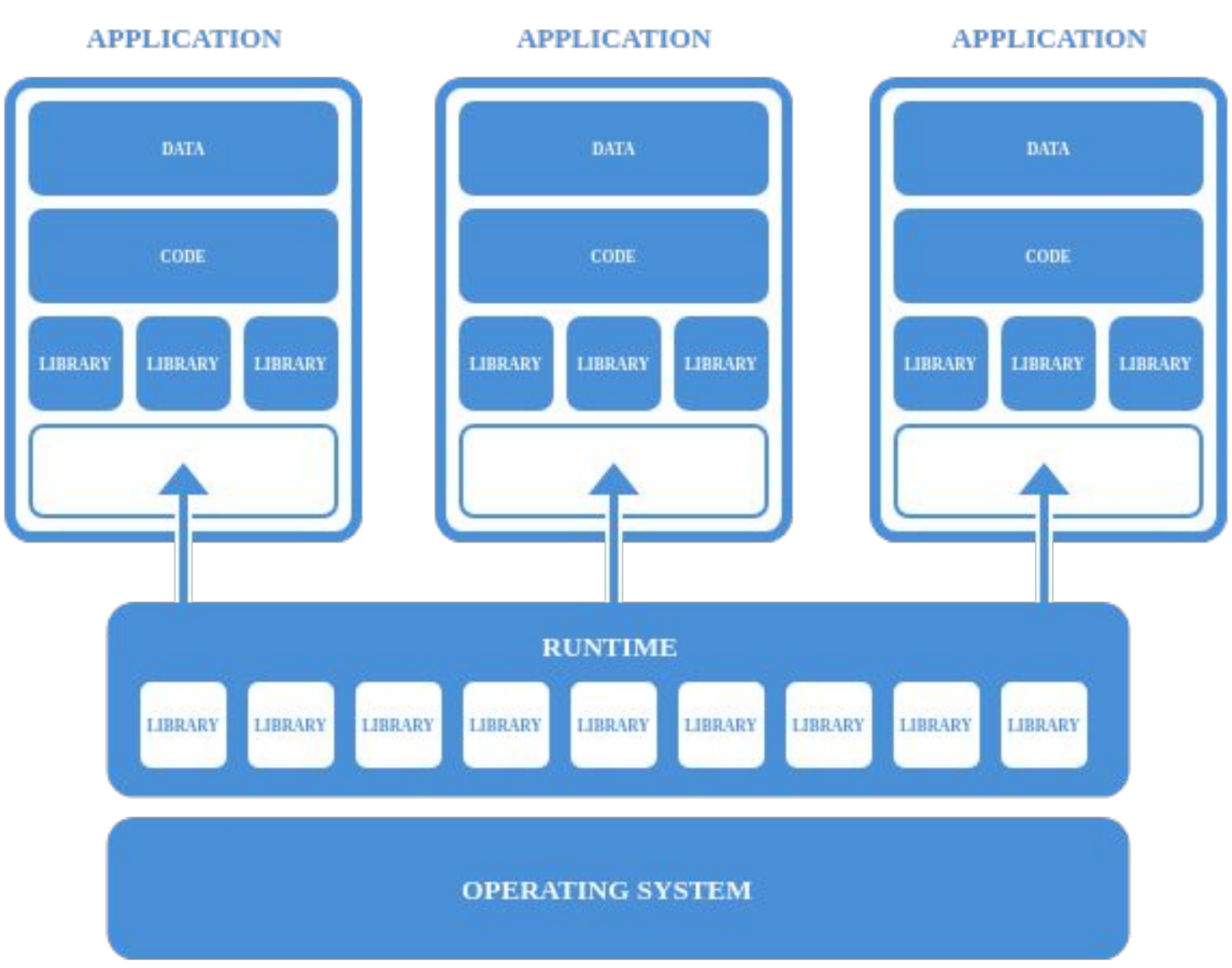
Documentation

docs.flatpak.org

Runtimes

Basic runtime dependencies e.g. alsa, cairo, clang, glib

Can be thought of as a /usr filesystem


APPLICATION
APPLICATION
APPLICATION
DATA
CODE
LIBRARY
LIBRARY
LIBRARY
DATA
CODE
LIBRARY
LIBRARY
LIBRARY
DATA
CODE
LIBRARY
LIBRARY
LIBRARY
RUNTIME
LIBRARY
LIBRARY
LIBRARY
LIBRARY
LIBRARY
LIBRARY
LIBRARY
LIBRARY
LIBRARY
OPERATING SYSTEM

Runtimes

Freedesktop, GNOME, KDE

Specific versioning of the stack

See the most used at flatpak.org/runtimes

SDK

The devel parts of a runtime

Headers, compilers, debuggers, packaging tools

Specific versioning of the stack

See the most used at flatpak.org/runtimes

Security & Sandboxing

App is fully containerized/sandboxed

Upfront permissions e.g. dbus names, directory paths, dconf, network...

Portals e.g. app chooser, file chooser, ... (more later)

Simple app code

Generate Python template in GNOME Builder

but, let’s do

gitlab.gnome.org/csoriano/baseapp

Simple app build

First building:

flatpak-builder --repo=repo .build org.gnome.BaseApp.json

Adding repo:

flatpak --user remote-add --no-gpg-verify --if-not-exists workshop repo

flatpak --user install workshop org.gnome.BaseApp

For updating:

flatpak --user update org.gnome.BaseApp

Simple app recommendations

Reverse DNS name

Desktop file

Icon

Standard build system e.g. Meson, Autotools, CMake

AppStream file

Important manifest options

Buildsystem - automake, cmake, meson, simple

config-opts

build-commands (especially for simple)

build-options - cflags, env, build-args (global), finish-args

post-install

cleanup

Important manifest options

Sources

archive

git

Build tweaks

patch

shell

script

Qt and KDE flatpaking

- Runtime and SDK
- Specific finish-args
- Qt and KDE build systems
- Integration with other desktops + extensions
- Documentation

Runtime and SDK

- Libraries
  - Qt (not all modules, just most used ones)
  - KDE Frameworks
  - Plasma integration libraries and theme (breeze, plasma-integration, kwayland-integration)
- Runtime version based on used Qt version (currently 5.9lts and 5.11)

Finish-args

- --talk-name=org.kde.StatusNotifierWatcher
  - needed for system tray support
- --talk-name=org.freedesktop.Notifications
  - needed for notification support
  - not needed when using KNotification framework
- --filesystem=xdg-config/kdeglobals:ro
  - needed for access to most common KDE configuration (colors, icons, font, theme)

Finish-args (for GNOME integration)

- --env=DCONF_USER_CONFIG_DIR=.config/dconf
- --filesystem=xdg-run/dconf
- --filesystem=~/.config/dconf:ro

Build systems

- CMake
- CMake-ninja
- QMake

Integration and extensions

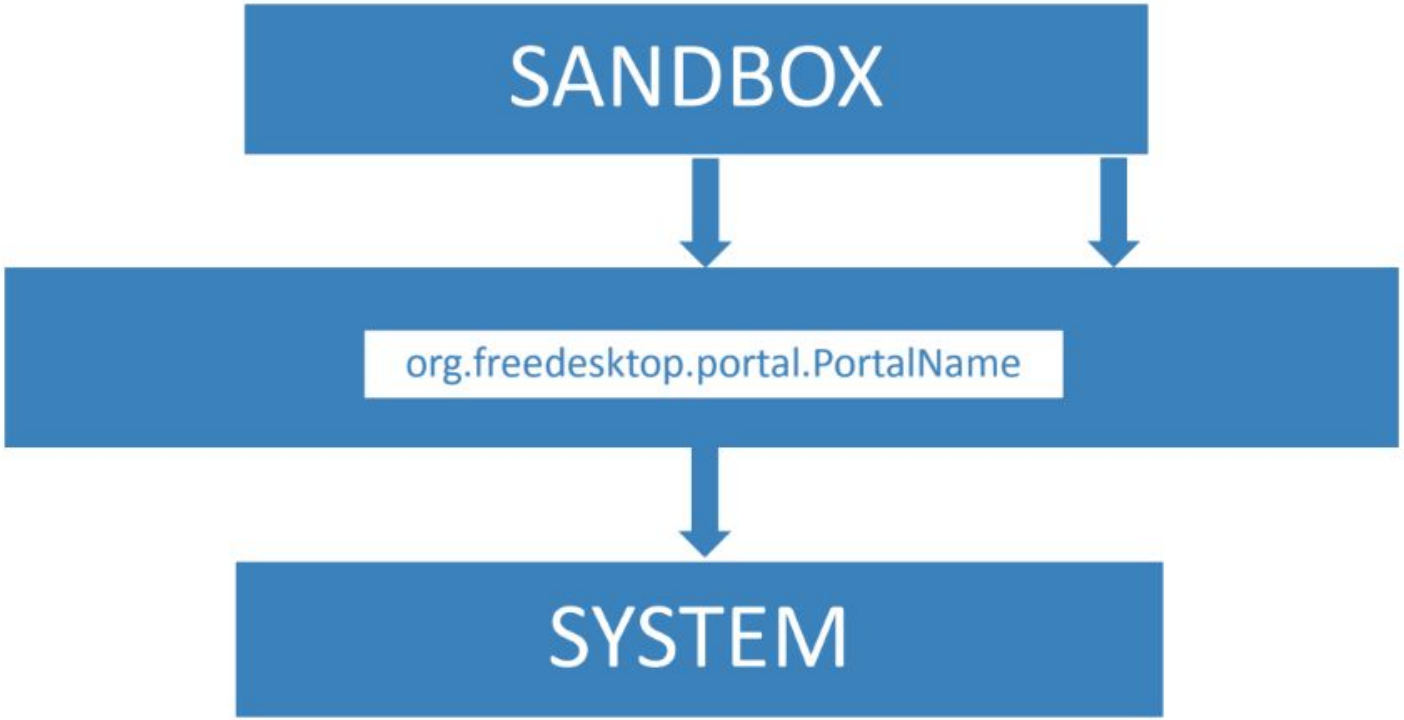
Default theme = Breeze

Flatpak uses extensions to support non-default stuff

KDE Extensions:

- org.kde.KStyle.StyleName (org.kde.KStyle.Adwaita)
- org.kde.PlatformTheme.PlatformName (org.kde.PlatformTheme.QGnomePlatform)

Portals



Portals

Support implemented in libraries (Qt, Gtk, KDE Frameworks)

Sandboxed apps communicate with org.freedesktop.portal.Desktop service

Requests are then forwarded to backend implementations:

- xdg-desktop-portal-kde (part of Plasma releases)
- xdg-desktop-portal-gtk

Portals

- Account
- Device
- Email
- FileChooser
- Inhibit
- NetworkMonitor
- Notification
- OpenURI
- Print
- RemoteDesktop
- ScreenCast
- Screenshot
- Trash

Portals demo

https://flathub.org/apps/details/org.flatpak.qtdemo

Tooling

KDevelop support

(http://jgrulich.cz/2018/09/03/flatpak-support-in-kdevelop)

Hack a Gtk+ app

Let’s use GNOME Builder to look and hack in Nautilus

Debugging

GNOME Builder

or

flatpak-builder --run builddir org.foo.bar.json sh

Debugging - filesystem structure

Your binaries are at /app/bin

Your data at /app/share

Generated manifest at /app/manifest.json

Your build at /run (use --keep)

Common tricks

Allow network while building (discouraged)

```
"build-args": [ "--share=network" ]
```

Custom build system

```
"buildsystem": "simple",
        "build-commands": [
            "python2 setup.py install --prefix=/app"
        ],
```

Common tricks

Using prebuilt binaries

VScode as example: github.com/flathub/com.visualstudio.code

Hack in installed app - Use “dir” & commit code changes before building.

```
"sources": [
 {
   "type": "git",
   "dir": "/home/$User/Projects/$Project"
 }
]
```

Common tricks

Override permissions
flatpak --user override --filesystem=home org.gnome.Testjs

Debugging a failing build

When building with flatpak-builder, pass --keep-build-dirs

flatpak-builder --run appdir org.my.Manifest.json sh

cd /run/build/failed-modulename

Distribute your app

Bundled app with .flatpak file

Good for deployments with no public connection

Bad for updates

Reference link with .flatpakref file

Distribute your app

flathub.org

Thank you!

flatpak.org

Suggested apps to package

- Qt applications
  - QtCreator
  - Wireshark
  - Trojita
  - Clementine
  - Yakuake
  - Deluge
  - Qbittorrent
  - Dogecoin-qt
  - Otter-browser
  - RStudio
- Other
  - Any gtk3+ theme
  - Any qt theme

- Gtk apps
  - Purpleegg
  - Mpv
  - smplayer/baka-mplayer (to be worked with mpv)
  - http://gnomepomodoro.org/
  - Mumble
  - pidgin

This slides + apps proposals

goo.gl/Q2WXWT