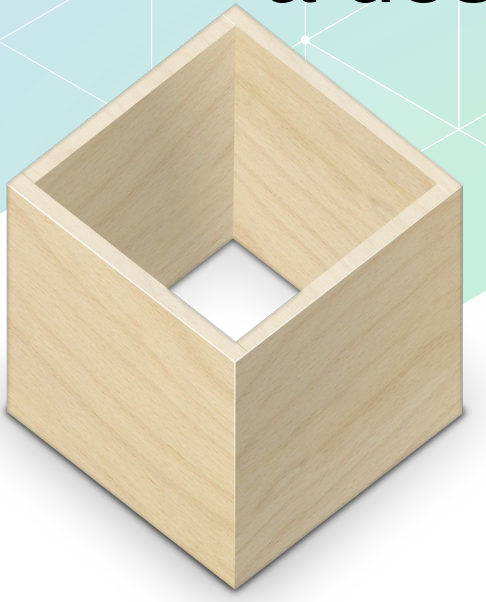
Flatpak

a desktop version of containers



Alexander Larsson, Red Hat

What is Flatpak?

A **distribution-independent**, **Linux-based** application **distribution** and **deployment** mechanism for **desktop applications**

distribution-independent

- **run** on any distribution
- **build** on any distribution
- Any **version** of the distribution

Linux-based

- Flatpak runs only on Linux
- Uses linux-specific features
- However, needs to run on older kernel
- Current minimum target
  - RHEL 7
  - Ubuntu 16.04 (Xenial)
  - Debian 9 (Stretch)

Distribution mechanism

- Built in support for install
- Built in support for updates
- Anyone can set up a repository

Deployment mechanism

- Run apps in a controlled environment
  - “container”
- Sandbox for improved security
  - Default sandbox is very limited
  - Apps can ask for more permissions

Desktop application

- Focus on GUI apps
- No root permissions
- Automatically integrates with desktop
- App lifetimes are ad-hoc and transient
- Nothing assumes a “sysadmin” being available

How is flatpak different from containers

Filesystem layout

Docker requirements

- Examples:
  - REST API micro-service
  - Website back-end
- Few dependencies, all hand-picked
- Runs as a daemon user
- Writes to nonstandard locations in file-system
- Not a lot of integration with host
  - DNS
  - Port forwarding
  - Volumes for data
- No access to host filesystem
- Updates are managed

Docker layout

- One image for the whole fs
  - Bring your own dependencies
  - Layout up to each app
- Independent of host filesystem layout

Flatpak requirements

- Examples
  - Firefox
  - Spotify
  - gedit
- Expects standard filesystem layout
- App data read-only
- Store state in users home directory
- Share file paths with host
- Lots of dependencies
  - Constant updates

Flatpak layout

- App image on /app, read-only
- Runtime in /usr
  - Shared dependencies
  - Versioned
  - Bundle everything else
- Some private mounts (/dev, /run, /proc, /tmp)
- Other paths mounted as on host (if visible)
- Store state in ~/.var/app/$APPID
- Some host data exposed in /run/host
  - Icons
  - fonts

How is flatpak different from containers

Filesystem implementation

Docker implementation

- Must support arbitrary writes
  - Needs union-style file-system
- / is one mount
  - Volumes are bind-mounted on top
- Docker daemon babysits
  - Mounts fs on container start
  - Unmount on container exit
- Uses layers to improve sharing

Flatpak implementation

- / is per-instance tmpfs
  - Unmounts when last process dies
- Images are read-only with “prefix” layout
  - Regular directories
  - Mounted read-only at /usr, /app
- Exposed host directories bind-mounted in place
- GC unused images using file locks
- Image content is shared via hard-linking
  - OSTree
  - Opportunistic sharing
  - Share disk and page-cache

The flatpak sandbox

Flatpak sandbox

- Two reasons:
  - Distro independence
  - Security
- Base of everything
  - User namespaces
  - PR_SET_NO_NEW_PRIVS
  - No root permissions!
  - Always use user uid
- App permissions:
  - Static
    - Set up at app launch time
  - Dynamic
    - Interactive
- Security domain is the application id
  - “org.gnome.gedit”

Bubblewrap

- Sandboxing setup helper
- Extracted from flatpak
  - https://github.com/containers/bubblewrap
- Builds up tmpfs from inside
- Useful from shell

```
bwrap --ro-bind /usr /usr
 --symlink usr/lib64 /lib64
 --proc /proc --dev /dev
 --unshare-pid bash
```

- Setuid alternative mode

Namespace use

- Pid ns unshared
  - Pid 1 is babysitter
- User ns unshared (if possible)
- Network ns unshared (by default)
  - Only loopback available
- Ipc ns unshared (by default)
  - Unfortunately important for XShm performance

Seccomp use

- Blocks
  - Syslog, accounting, quota
  - Various scary VM syscalls
  - Weird socket families (x25, ipx, etc)
  - Kernel keyring
  - Recursive namespaces
- Optionally allow
  - Multiarch
  - Perf
  - PTrace

CGroup use

- Creates systemd --user scope
  - “app-flatpak-$appid-$pid.scope”
- Hard to do more unprivileged

Device nodes

- Default /dev
  - full, null, zero
  - stdin, stdout, stderr
  - random, urandom
  - tty, pts, ptmx, console
- Optionally
  - dri, nvidia
  - kvm
- Also optionally whole host /dev

Sockets

- Optional
  - X11
  - Wayland
  - PulseAudio
  - Cups
  - ssh agent
  - pcsc (smartcard)
  - System dbus
  - Sesson dbus
- Always
  - p11-kit server (pkcs11 certs)

DBus filtering

- Connect to session bus via filtering proxy
  - xdg-dbus-proxy
- Default access
  - Can talk to the bus itself
  - Can receive messages
  - Can own app-id name (*org.gnome.gedit*)
  - Can talk to org.freedesktop.portal.*
- Extensible via permissions
- Also a11y, system busses

Portals

- Accessible due to default filter
- Permissions are enforced by portal itself
  - Based on peer socket credentials
  - Interactive
- Existing portals
  - Xdg-desktop-portal
    - File chooser, print, open-uri, screencast, etc...
    - Backends: gtk, kde
  - Document portal
    - Fuse mount
  - Flatpak portal
    - Sub-sandbox
    - Self-updates

Plumbing issues

Tagging containers

- Set Immutable ID on new container
- Portals need to identify containers
  - Given a unix domain socket fd
- Current options
  - SO_PEERCREDS
    - pid, uid, gid
  - SO_PEERSEC
    - SELinux label, AppArmour context, ...
  - CGroup path
- Now: SO_PEERCREDS → pid → /proc/$PID/root/.flatpak-info
  - Pids are racy
  - Disallows recursive namespace use

Abstract sockets

- Bound to network namespace
- Broken scenarios
  - Network access, but not all abstract sockets
  - No network, but some abstract socket
- Abstract sockets are lame
  - Limited path length
  - Can’t rearrange in namespace
- Can everyone stop using them!
  - Just use regular sockets in /run

Userspace network filtering

- Currently all or nothing
- Want in-between option
  - NAT:ed
  - IP range filtering
  - …
- Current options
  - slirp4netns
  - CGroup eBPF socket filter
    - Needs root

Want unprivileged overlayfs

- Not currently possible
- Mainly for building

fs-verity

- Immutable files
- Merkle tree allows signatures
- Very good match for OSTree
- Needs more fs support

CGroups v2

- Seems like it could be used for flatpak
- Needs research

Dynamic device nodes

- Want device type filtering
  - “All joysticks”
  - “All usb devices with this vendor/class”
  - Dynamic
- Currently only possible if in subdir
  - /dev/dri
- Want better approach?

Proxy-less dbus filtering

- Don’t want proxy
  - Less processes
  - Less copies
- Filter in the bus itself
- dbus-daemon implementation:
  - https://gitlab.freedesktop.org/dbus/dbus/-/issues/185
  - Status
    - WIP
    - No updates in years
- Other implementations
  - dbus-broker?

PipeWire

- New video & audio daemon
- Replaces PulseAudio
- Built with sandboxing in mind
- Needs work to integrate as portal

GPU drivers

- Needs to match kernel version
  - Nvidia – hard requirement
  - Mesa/DRM – soft requirement
- ABI issues makes it hard to use host driver
  - libcapsule / dlmopen
- DRI kernel ABI needs to be backwards compat

Questions

- Links:
  - https://flatpak.org/
  - https://github.com/flatpak/flatpak/
  - https://github.com/ostreedev/ostree
- Reaching us
  - alexander.larsson@gmail.com
  - flatpak@lists.freedesktop.org
  - #flatpak on freenode