Bootstrappable Freedesktop SDK

Dor Askayo

Linux App Summit 2023

Basic concepts

Freedesktop SDK

- A minimal Linux runtime platform and SDK
- Builds hundreds of Linux software projects from source in multiple architectures
  - x86_64, i686, aarch64, ppc64le, riscv64
- Easy to consume through Flatpak and BuildStream
- Fully reproducible
- A key component in multiple projects
  - The base runtime of practically all Flatpak apps on Flathub and GNOME Nightly
  - The base runtime of GNOME OS
  - The runtime used for bootstrapping carbonOS

Source code: https://gitlab.com/freedesktop-sdk/freedesktop-sdk

The bootstrap challenge

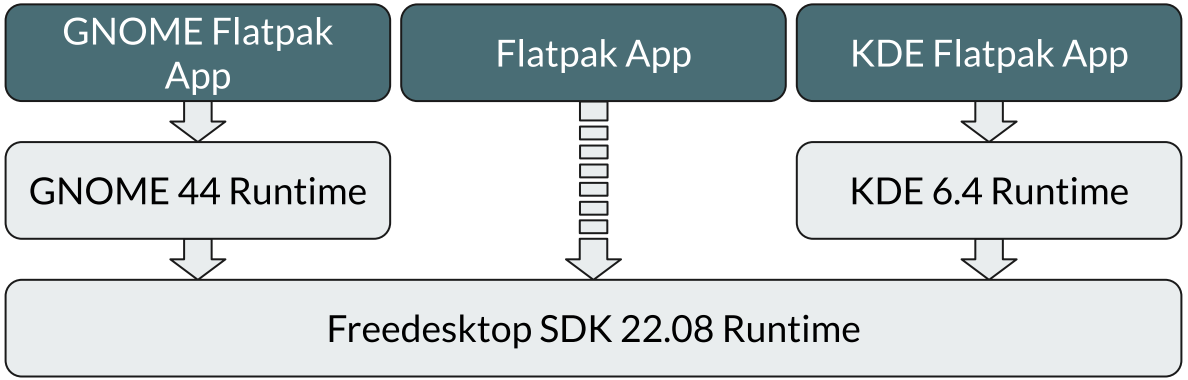
- Building software can sometimes be a chicken-and-egg situation
  - For example, building a C compiler written in C
- Bootstrapping is the process of breaking such circular dependencies
- Modern software stacks are built on many historical layers of bootstrapping steps
- No single solution to bootstrapping
  - Projects usually expect their users to deal with bootstrapping
  - Some projects are more considerate
- “Bootstrappable” projects provide an “out of the box” solution
  - More information: https://bootstrappable.org

BuildStream

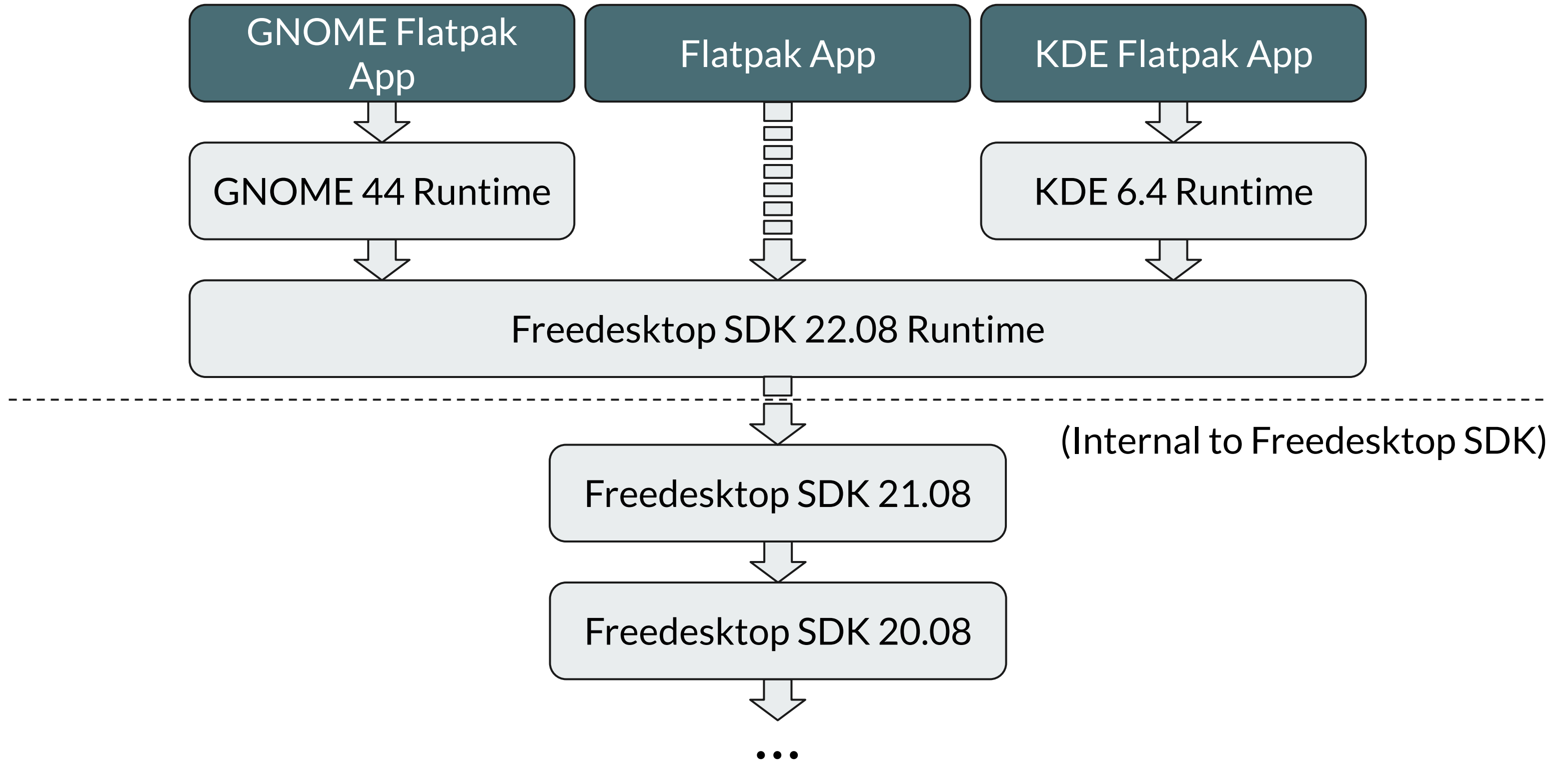
- A tool for automating the integration of software components
- Plugins allow automation of many tasks
  - Building using all common build systems (Meson, Autotools, make, etc.)
  - Fetching and tracking source files of various types (Git, tarballs, pypi, etc.)
- Sandboxed build environment allows exposing a minimal set of build dependencies
- Built-in support for artifact caching, source caching and mirrored sources
- Used to build the Freedesktop SDK project

More information: https://www.buildstream.build

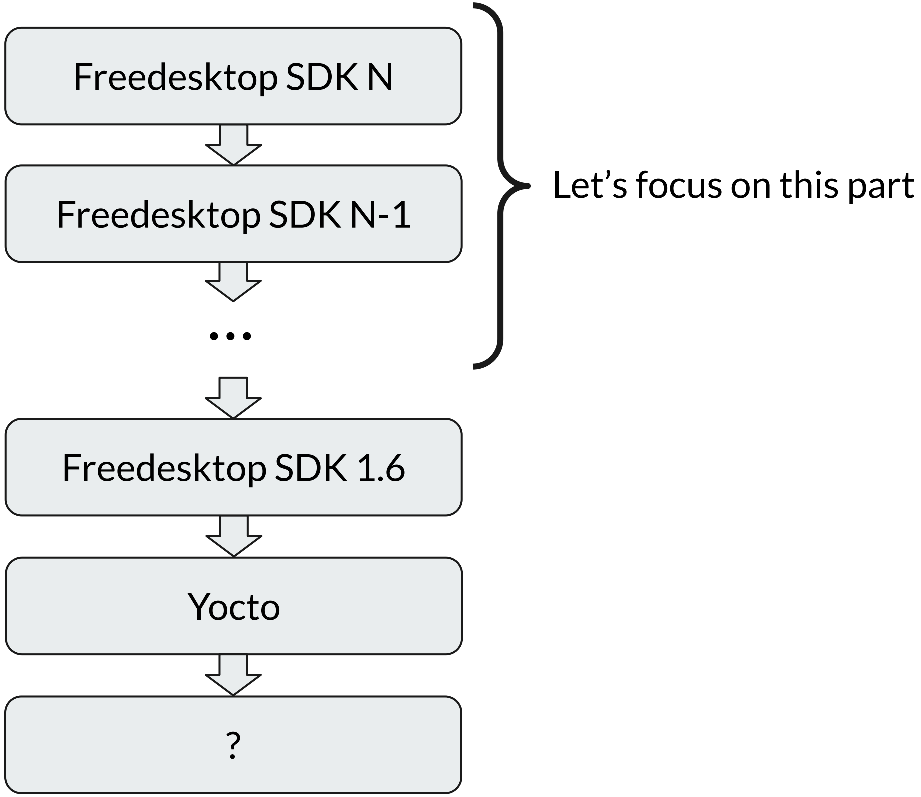
Freedesktop SDK Bootstrap – Design & Limitations



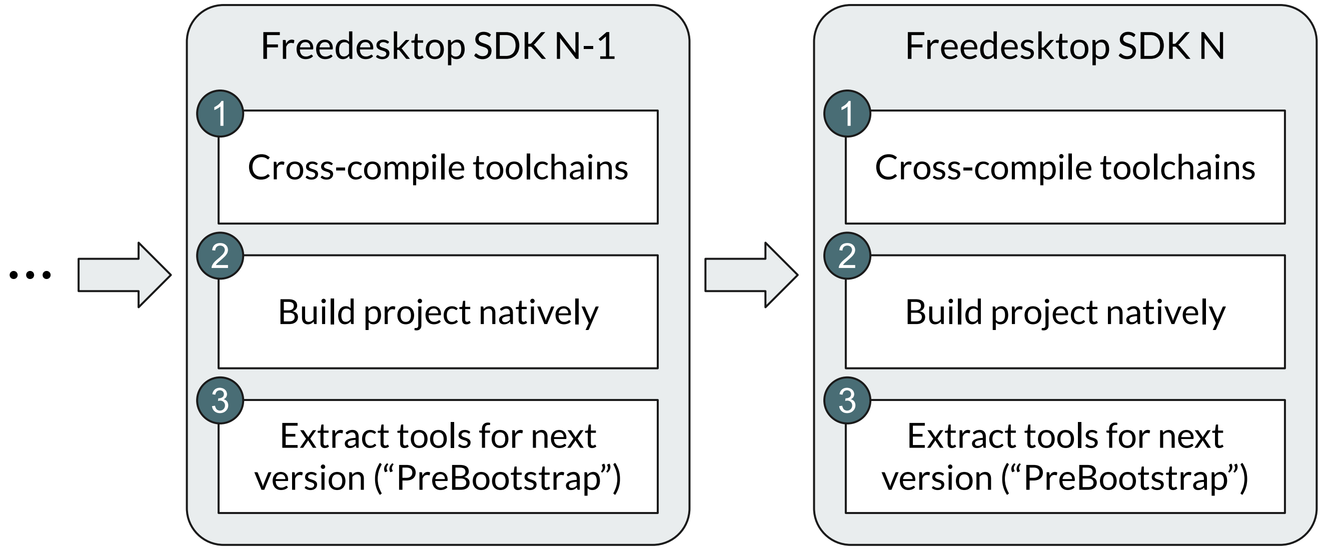
Flatpak high-level dependency chain



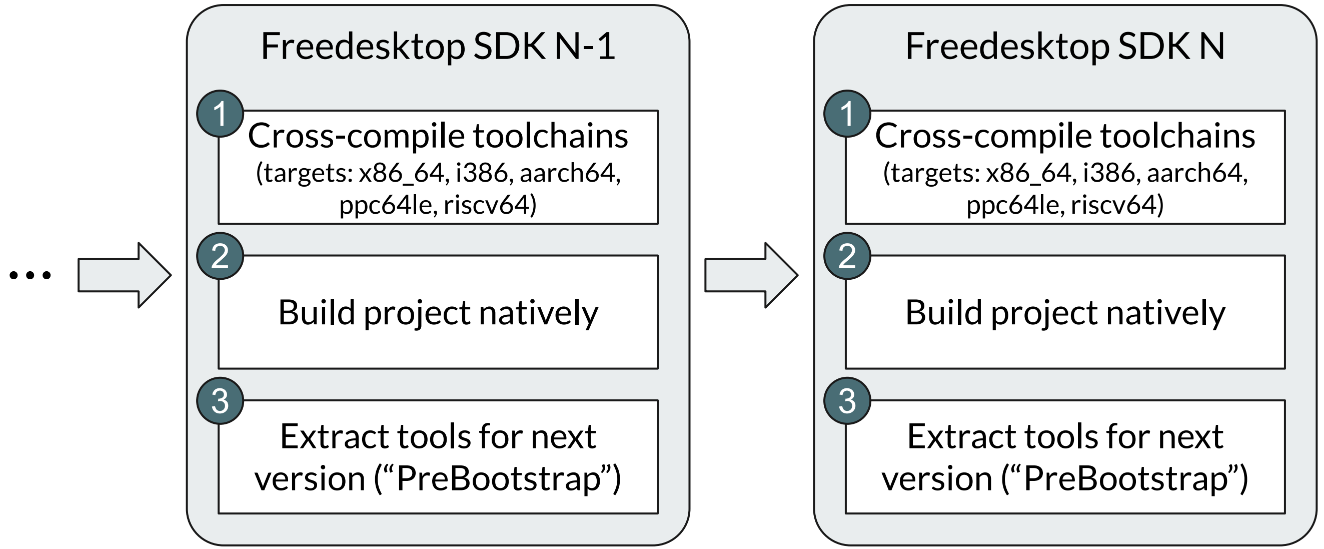
Flatpak high-level dependency chain



Freedesktop SDK high-level bootstrap dependency chain



Freedesktop SDK high-level bootstrap build steps (snippet)


...
Freedesktop SDK N-1
1
Cross-compile toolchains
(targets: x86_64, i386, aarch64, ppc64le, riscv64)
2
Build project natively
3
Extract tools for next version ("PreBootstrap")
Freedesktop SDK N
1
Cross-compile toolchains
(targets: x86_64, i386, aarch64, ppc64le, riscv64)
2
Build project natively
3
Extract tools for next version ("PreBootstrap")

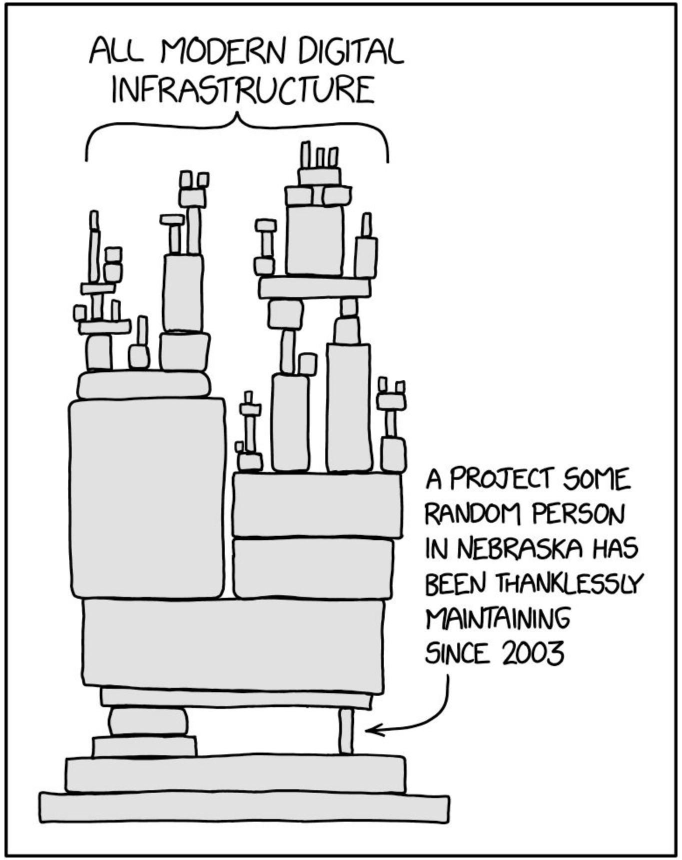
Freedesktop SDK high-level bootstrap build steps (snippet)

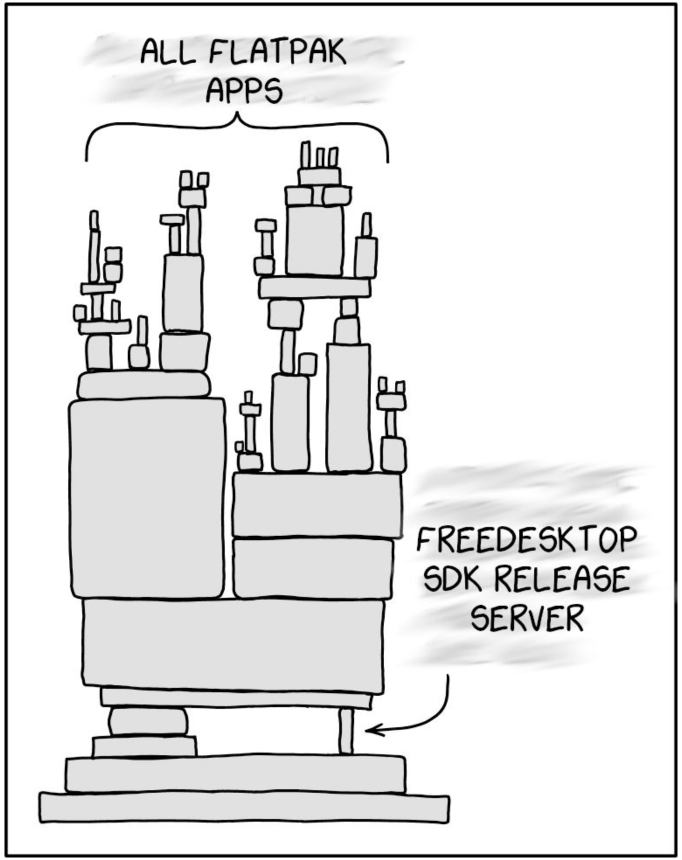
Limitations

- Dependency graph is not clear
  - The full set of inputs required to build the project is not feasible to map, document or audit
- Project can’t be built from source code alone
  - Building all previous Freedesktop SDK versions is not a reasonable expectation
- “PreBootstrap” image – a single point of failure
  - The bootstrap path heavily depends on these large binary images
  - Images are hosted on Freedesktop SDK’s release server
  - What would happen if an image is lost?
    - The bootstrap path breaks ⇒ the Freedesktop SDK project becomes unable to build
    - This already happened multiple times!

Limitations

- Dependency graph is not clear
    - The full set of inputs required to build the project is not feasible to map, document or audit
- Project can't be built from source code alone
    - Building all previous Freedesktop SDK versions is not a reasonable expectation
- "PreBootstrap" image – a single point of failure
    - The bootstrap path heavily depends on these large binary images
    - Images are hosted on Freedesktop SDK's release server
    - What would happen if an image is lost?
        - The bootstrap path breaks ⇒ the Freedesktop SDK project becomes unable to build
        - This already happened multiple times! (yes, workarounds were found)


ALL MODERN DIGITAL INFRASTRUCTURE
A PROJECT SOME RANDOM PERSON IN NEBRASKA HAS BEEN THANKLESSLY MAINTAINING SINCE 2003


ALL FLATPAK APPS
FREEDESKTOP SDK RELEASE SERVER

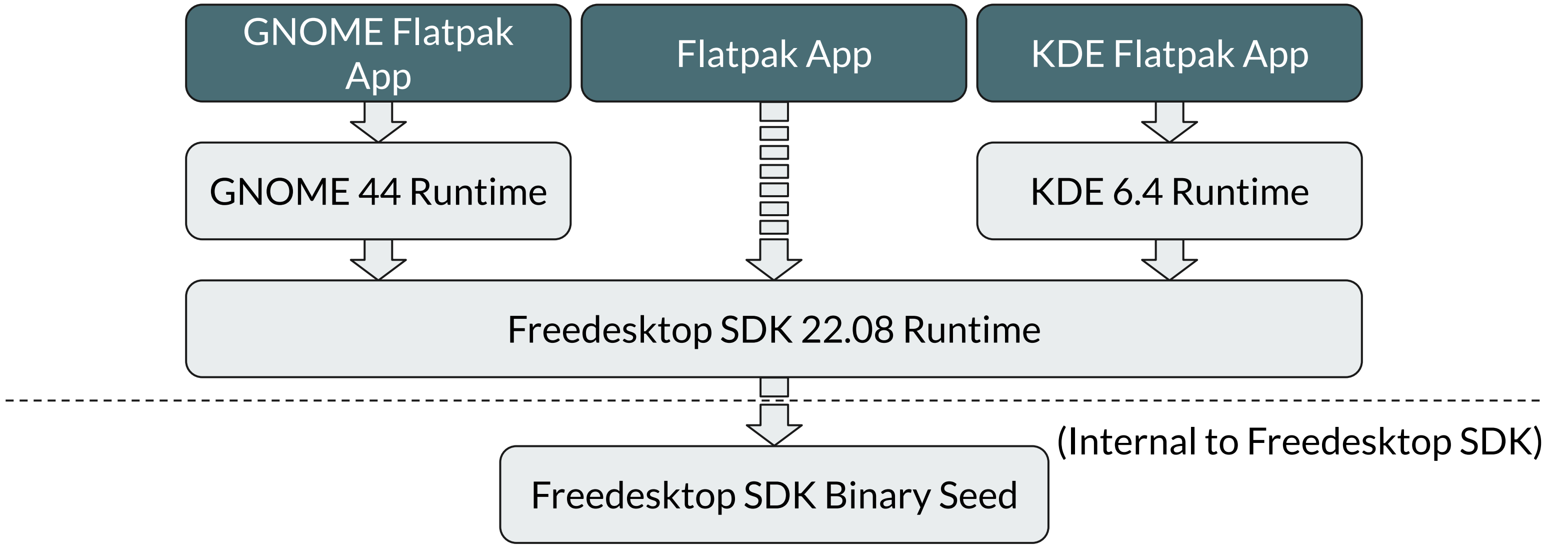
Freedesktop SDK Bootstrap – New Design

Design goals

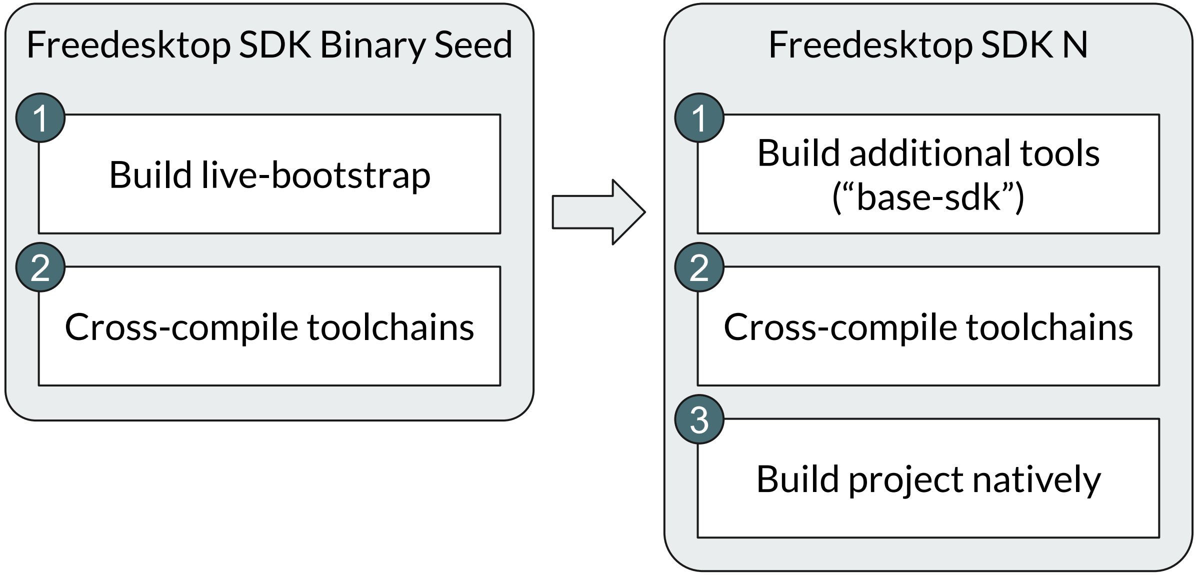
1. Break the recursive dependency on previous versions of Freedesktop SDK
2. Have a clear and simple dependency graph
3. Use minimal binary seeds for the bootstrap
4. Build all non-seed binaries from source
5. Allow auditing of all source files
6. Maintenance should be easy and automated (as much as possible)

Benefits

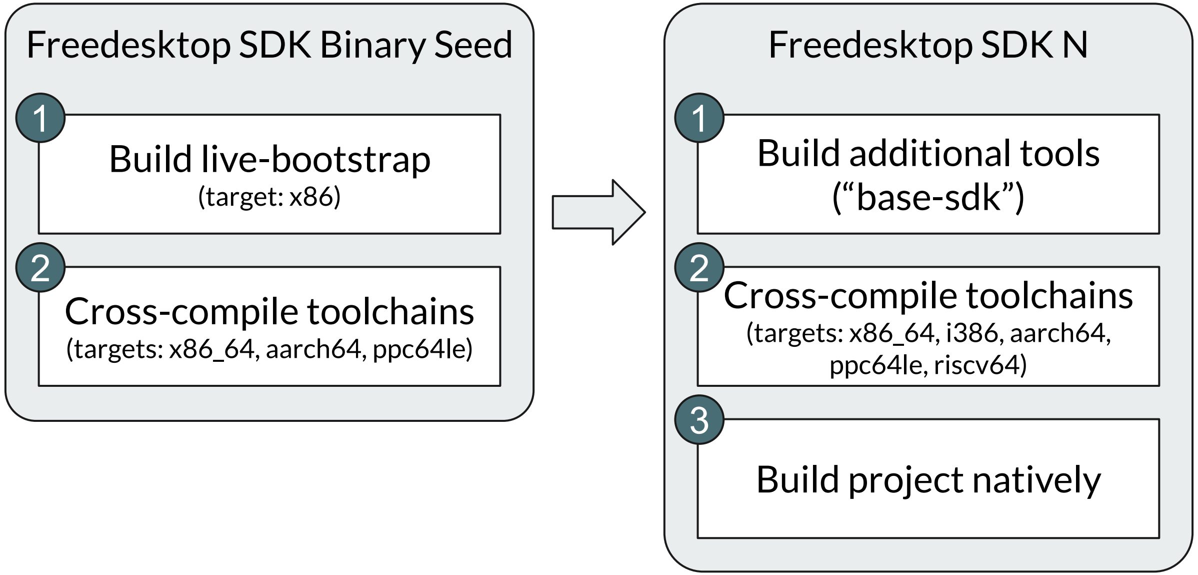
- The entire bootstrap path is documented and auditable
- Project can be built and reproduced from source code alone
- Easy to bootstrap new architectures


GNOME Flatpak App
Flatpak App
KDE Flatpak App
GNOME 44 Runtime
KDE 6.4 Runtime
Freedesktop SDK 22.08 Runtime
(Internal to Freedesktop SDK)
Freedesktop SDK Binary Seed

Flatpak high-level dependency chain – New design



Freedesktop SDK high-level bootstrap build steps – New design



Freedesktop SDK high-level bootstrap build steps – New design

Caveats

- Rust still isn’t bootstrapped from source
    - It’s not so simple – more on that later
- Some host tools are still used by BuildStream to fetch sources and set up the build sandbox
    - For example: Git, Python, Bubblewrap, etc.

live-bootstrap –
Doing the Heavy Lifting

What is live-bootstrap?

- Started by fosslinux (@fossy) and Andrius Štikonas (@stikonas)
  - Initial version published in December 2020
  - Actively maintained and expanding
- Builds upon the immense efforts of the Bootstrappable Builds community
  - #bootstrappable @ Libera.Chat
- Bootstraps a fully working x86 (32-bit) system from minimal binary seeds
  - Currently: ~1 KB binary + Linux kernel
  - Soon™: ~3KB binary (without a Linux kernel)
- Supports multiple bootstrap modes: bare-metal, QEMU, bubblewrap, chroot

Source code: https://github.com/fosslinux/live-bootstrap

live-bootstrap – Bootstrap strategy

- Early bootstrap steps build projects dedicated for bootstrapping
  - stage0-posix (https://github.com/oriansj/stage0-posix)
  - GNU Mes (https://www.gnu.org/software/mes)
- Later bootstrap steps typically follow a historical path
  - Luckily, software projects are usually bootstrappable at their early versions
  - Bootstrap starts with projects from the 90s and early 2000s, and gradually gets to 2023
- Patching and workarounds are used when needed, especially early in the bootstrap
- All bootstrap steps are documented: https://github.com/fosslinux/live-bootstrap/blob/master/parts.rst

live-bootstrap – Gaps (early 2022)

1. Bootstrapped software was too old to bootstrap Freedesktop SDK
2. Building in BuildStream was not possible
    - All bootstrap modes required root permissions

Freedesktop SDK Bootstrap – New Design Implementation

live-bootstrap – Modernization & improvements

- Dynamic linking support for musl
- Modernization of bootstrap and components
  - Python 3.11.1
  - GCC 10.4.0
  - Binutils 2.38
  - Modern Autotools
  - Much more!
- Many bug fixes and improvements

live-bootstrap – Preparation for BuildStream

- A new Bubblewrap-based bootstrap mode
  - Allows the project to be built without root permissions in a chroot-like environment
  - Build environment is similar to BuildStream’s build sandbox
- Simplification of project directory structure
- Support for generating source manifests
- Avoiding the use of chroot(2) during bootstrap

Freedesktop SDK Binary Seed – New project

- Builds live-bootstrap and extracts select packages from it
- Cross-compiles native GNU toolchains for x86_64, aarch64 and ppc64le
  - Adding support for additional architectures is trivial
- Builds and publishes Docker images for easy consumption by Freedesktop SDK
- Uses newly-written BuildStream plugins for live-bootstrap
  - “live_bootstrap_manifest” – Tracks and fetches source files required by live-bootstrap
  - “live_bootstrap_prepare” – Performs simple preparatory operations in the source directory
  - “command” – Executes a command in the build sandbox

Source code: https://gitlab.com/freedesktop-sdk/freedesktop-sdk-binary-seed

Freedesktop SDK – Build using the new seed

- Use the Freedesktop SDK Binary Seed to kick-start the bootstrap process
- Build additional tools during the initial bootstrap (“base-sdk”)
    - Autotools, Perl, Tar, xz, etc.
- Improve reuse of built components during early bootstrap

Implementation Status

Implementation status

- All changes were merged upstream
- The upcoming Freedesktop SDK 23.08 will be bootstrapped from the new binary seed
- Also… backported to 22.08 already!
    - Available since Freedesktop SDK 22.08.9 (released on March)

Thanks to everyone involved!

- Made possible thanks to contributions and review from multiple people
- Special thanks:
  - fosslinux (@fossy)
  - Andrius Štikonas (@stikonas)
  - Seppo Yli-Olli (@nanonyme)

Future work

Future work – Automated source mirroring

- live-bootstrap requires some very old source tarballs for its bootstrap
    - Some are from the 90s!
- Source tarballs are usually obtained from upstream hosting or third party mirroring
- Source tarballs disappearing could mean losing a critical part of the bootstrap path
- Freedesktop SDK should set up its own mirroring to be on the safe side
- Lorry would be a good approach for automated mirroring: (also used by gnome-build-meta)
    - https://gitlab.com/CodethinkLabs/lorry/lorry
    - https://gitlab.com/CodethinkLabs/lorry/lorry-controller

Future work – Source-based bootstrap for Rust

- Currently bootstrapped in Freedesktop SDK using official pre-built binaries
    - The last remaining binary files that can't be audited!
- Each version of rustc (the Rust compiler) expects to be built using its previous version
- Two potential bootstrap paths from C/C++:
    - mrustc – can build Rust 1.54.0, from 2021 (https://github.com/thepowersgang/mrustc)
    - Rust-GCC – early stage, upstreamed to GCC (https://github.com/Rust-GCC/gccrs)
- No sustainable source-based bootstrap path, yet
    - The shortest path is mrustc, which still requires 17 steps to bootstrap Rust 1.69.0 (latest stable)

Questions?

Thank you!