Building Software Faster with the Power of Open Source

Christopher Phang
Codethink Ltd.
My background

- Academic background. Worked as an organic chemist and in drug discovery for 6 years.
- Switched to Software development in 2015.
- Joined Codethink in 2016
- Working in systems integration and build engineering since 2018
Outline

- Codethink
- Open source software
- Software at scale
- Some solutions
- Collaboration for fun and profit: Alex Lyon
- Why you should get involved!
Who is Codethink?

- Based in Manchester UK!
- Started out just over 10 years ago as a small open source contracting firm
- We specialise in embedded - Open Source - Systems Integration
- Have grown year on year to over 90 employees, with a wide range of skills!
Open Source - What is it?

- Freedom 0: The freedom to *run* the program for any purpose.
- Freedom 1: The freedom to *study* how the program works, and change it to make it do what you wish.
- Freedom 2: The freedom to *redistribute* and make copies so you can help your neighbor.
- Freedom 3: The freedom to *improve* the program, and release your improvements (and modified versions in general) to the public, so that the whole community benefits.
Open Source projects you may have heard of...
Linux

- 70% of the web
- 92% of Amazon Cloud Servers
- 85% of supercomputers
- Two billion Android devices
- Primary OS for embedded devices
- Linux supports more devices than any other operating system
- Started in 1991 by Finnish student named Linus Torvalds
Open Source - Why do people care?

**Freedom**
Open source technology respects your freedoms, it allows you to control how you interact, distribute, utilise a piece of software.

**Control**
It gives you the control to review/modify/fix/improve the software you are using, you are not tied to a user agreement which limits those abilities.

**Reliability**
Due to the culture around open source technology and the ability to modify/improve the source code, the code inherently can be more reliable, especially as the community grows in size.

**Knowledge**
It is common to have a sense of community around open software, which encourages people to learn and develop new skills.
Open Source - why should I care?

- Improve your technical skills
- Feel a sense of purpose/responsibility
- Give back to your community
- Make new connections/friends
- Attend awesome conferences ([FOSDEM.org](http://FOSDEM.org))
- Earn money whilst giving back to a wider community
How can you get started?

- There are many projects and places to start
- Start small (documentation)
- Be creative (new ideas/improvements)
- Product testing (beta users)
- User groups
<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>.gitignore</td>
<td>Add directories from make test-apps</td>
<td>9 months ago</td>
</tr>
<tr>
<td>.gitlab-ci.yml</td>
<td>Make VM tests using images manual</td>
<td>6 days ago</td>
</tr>
<tr>
<td>.gitmodules</td>
<td>Revert &quot;BuildStream wants absolute URL&quot;</td>
<td>3 months ago</td>
</tr>
<tr>
<td>.pylintrc</td>
<td>remove deprecated in disabled messages</td>
<td>1 month ago</td>
</tr>
<tr>
<td>BOOTABLE_IMAGES.md</td>
<td>BOOTABLE_IMAGES.md: Provides notes on how to use</td>
<td>1 month ago</td>
</tr>
<tr>
<td>CONTRIBUTING.md</td>
<td>Completely remove references to mesa-aco</td>
<td>2 months ago</td>
</tr>
<tr>
<td>LICENSE</td>
<td>Add MIT license</td>
<td>2 years ago</td>
</tr>
<tr>
<td>Makefile</td>
<td>Makefile (all): build oci/layers/flatpak.bst</td>
<td>6 days ago</td>
</tr>
<tr>
<td>NEWS</td>
<td>NEWS: Update for freedesktop-sdk 19.08beta9</td>
<td>4 months ago</td>
</tr>
<tr>
<td>README.md</td>
<td>Completely remove references to mesa-aco</td>
<td>2 months ago</td>
</tr>
<tr>
<td>code-of-conduct.md</td>
<td>Add Code of Conduct</td>
<td>4 months ago</td>
</tr>
<tr>
<td>flatpak-version.yml</td>
<td>Bump flatpak branch to 20.08beta</td>
<td>4 months ago</td>
</tr>
<tr>
<td>project.conf</td>
<td>Correct name of re_import plugin</td>
<td>2 weeks ago</td>
</tr>
<tr>
<td>snap-version.yml</td>
<td>Bump flatpak branch to 20.08beta</td>
<td>4 months ago</td>
</tr>
</tbody>
</table>
Pull/Merge Request (branches)

- You’ve found a project on GitHub, GitLab
- You simply fork the project, make the change and then open a pull request upstream
- This opens up a communication channel where your changes can be discussed/improved/merged into master

Source: bitbucket.com
Software at scale
Software at scale

- Up to hundreds of millions of lines of code
- Up to tens of thousands of developers
- How do we build software as quickly and reproducibly as possible?
# Google repository statistics

As of Jan 2015

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total number of files*</td>
<td>1 billion</td>
</tr>
<tr>
<td>Number of source files</td>
<td>9 million</td>
</tr>
<tr>
<td>Lines of code</td>
<td>2 billion</td>
</tr>
<tr>
<td>Depth of history</td>
<td>35 million commits</td>
</tr>
<tr>
<td>Size of content</td>
<td>86 terabytes</td>
</tr>
<tr>
<td>Commits per workday</td>
<td>45 thousand</td>
</tr>
</tbody>
</table>

*The total number of files includes source files copied into release branches, files that are deleted at the latest revision, configuration files, documentation, and supporting data files.

Source: https://www.youtube.com/watch?v=W71BTkUbdqE
Systems integration

- Linux Kernel
- Toolchain
- Middleware libs
- Application
Layers of cake: managing dependencies

- Package managers *don’t* solve the whole problem.
- How do you update artifacts and ensure everything still works on a daily or hourly basis?
- What if you need to make source code changes?
- What if you don’t want to bring in an entire distribution?
The Not Rocket Science Rule of Software Engineering

Automatically maintain a repository of code that always passes all the tests

Graydon Hoare: Creator of Rust

- Provide single source of truth: *continuous integration*
- How do I keep the cycle time as short as possible?

Source: cloud.google.com
Some solutions
Building reproducibly: managing dependencies

- **Integration tool**: Brings many different software packages together into a single cohesive whole
- Support for a wide range of build systems
  - A build describes the ‘recipe’ of how to build a piece of software.
  - The DSL, used to describe the recipe, and the executable(s) to process it is the build system
  - Examples: Make, CMake, Autotools, Meson, Bazel
- Produce a whole system, in a variety of outputs
An example component

kind: cmake

depends:
- filename: bootstrap-import.bst
- filename: public-stacks/buildsystem-cmake.bst
type: build
- filename: components/nasm.bst
type: build

variables:
  cmake-local: |
  -DCMAKE_SKIP_RPATH:BOOL=NO

sources:
- kind: git_tag
  url: github:libjpeg-turbo/libjpeg-turbo.git
  track: master
  ref: 2.0.3-0-g5db6a6819d0f904e0b58f34ae928feaa234adb1a0
Why does this help?

● We build in a controlled *sandboxed* environment
● This means you create a *reproducible, traceable* environment.
● You have a scalable, rigorous solution towards *dependency management*

*A minimal linux runtime*

freedesktop-sdk

*A SDK for machine learning*

libreML
“It only works on my machine”

- Reproducibility on builds and on tests is critically important for the success of any software engineering project.
- This starts with an environment you can have confidence in, and dependencies that you trust.
Systems integration

- Linux Kernel
- Toolchain
- Middleware libs
- Application

BuildStream
Building as quickly as possible

- Task Parallelism
- Build and integration tools have a dependency graph that can be expressed as a DAG
Using remote execution

- Can we exploit task parallelism through usage of distributed computing resources?
- Is there an open API that allows client and server implementations to interoperate?

```plaintext
--
34 // The Remote Execution API is used to execute an
36 // workers.
37 //
38 // As with other services in the Remote Execution API, any call may return an
39 // error with a [RetryInfo][google.rpc.RetryInfo] error detail providing
40 // information about when the client should retry the request; clients SHOULD
41 // respect the information provided.
42 service Execution {
43 // Execute an action remotely.
```

https://github.com/bazelbuild/remote-apis
Why this is useful

- A build tool for large applications
- An integration tool for dependency management

Remote Execution API (REAPI) Clients!
Systems integration
Server implementations

https://github.com/buildbarn
Just deploy and go?

- What hardware do I use?
- How do I declare the hardware that I use in a reproducible manner?
- How do I monitor what is happening on clients and servers?
Bringing it all together

- How do we build software as **quickly** and **reproducibly** as possible?
- How do we make this approach scale to millions of lines of code, and thousands of developers?
  - *Using integration tools such as buildstream to manage 3rd party dependencies.*
  - *Deploying infrastructure that exploits large task parallelism in builds*
  - *Using build tools such as bazel that are interoperable and have a common open API*
Collaboration for Fun and Profit
collaboration
/kələˈbeɪʃən/ noun

1. the action of working with someone to produce something.
   "he wrote a book in collaboration with his son"

synonyms: cooperation, alliance, partnership, participation, combination, association, concert; More
Lets Learn Something About Your Favourite Tech
My Start

Programming → Github → Open Source
Why Avoid Closed-Source?
- Computers aren’t getting faster
- “Horizontal Scaling”
- How do we use more cores at once?

- Constrained developer resources
- Offload it?
- What do we need for that?

- ‘Pure’ build process
- Pooled cache
- Some sort of API

- Remote execution!
- Not a solved problem
- Buggy, and extremely complex
Code I’ve Written Is...

- In your media player
- In your (little brothers) minecraft world
- In your build client
- In your build server
- Enabling geospatial queries in python
- Helping you look up crime nearby

And that’s a really cool feeling!
How Can I Start?
Building software faster with the power of open-source

- Kubernetes
- freedesktop-sdk
- Terraform
- libreML
- Bazel
- BuildStream
- Grafana
Why is it interesting?

- Wide range of skills required
  - Data analytics
  - Software Development
  - IT Operations
  - Virtualisation
  - Distributed computing
  - Networking
  - Filesystems
Getting involved

- Codethink is heavily involved in this space.
  - Started Buildstream
  - Active contributors to many of the projects already mentioned
  - Developing new, open-source approaches towards software construction and deployment

- Help us!
  - Learn new skills
  - Meet interesting people
  - Develop a portfolio of work that you can show to employers

- [https://gitlab.com/celduin](https://gitlab.com/celduin)
- #celduin on freenode [https://webchat.freenode.net](https://webchat.freenode.net)
Join us!

- Summer placements
- Year in Industry
  - A chance to work on customer projects
  - An opportunity to attend technical conferences
- Last chance!
- jobs@codethink.co.uk
Thank you!

christopher.phang@codethink.co.uk
https://gitlab.com/celduin
#celduin on freenode