Rebuilding the build farm

A new package builder architecture

Achill Gilgenast | Patrycja Rosa

What is a builder?

builder (n.) – machine that turns
APKBUILDs into .apks

Current architecture: build.a.o

Current architecture: build.a.o

- Each release/arch has a dedicated builder

Current architecture: build.a.o

- Each release/arch has a dedicated builder
- Building all packages where $pkgver != $built_pkgver

Current architecture: build.a.o

- Each release/arch has a dedicated builder
- Building all packages where $pkgver != $built_pkgver
- After each repository (main, community, testing), uploads the repo to master.a.o

"decentralized" builders

"decentralized" builders

- Each builder has its own signing key

"decentralized" builders

- Each builde

```
~ ❯ apk info -L alpine-keys
alpine-keys-2.6-r0 contains:
etc/apk/keys/alpine-devel@lists.alpinelinux.org-4a6a0840.rsa.pub
etc/apk/keys/alpine-devel@lists.alpinelinux.org-5261cecb.rsa.pub
etc/apk/keys/alpine-devel@lists.alpinelinux.org-6165ee59.rsa.pub
usr/share/apk/keys/alpine-devel@lists.alpinelinux.org-4a6a0840.rsa.pub
usr/share/apk/keys/alpine-devel@lists.alpinelinux.org-5243ef4b.rsa.pub
usr/share/apk/keys/alpine-devel@lists.alpinelinux.org-524d27bb.rsa.pub
usr/share/apk/keys/alpine-devel@lists.alpinelinux.org-5261cecb.rsa.pub
usr/share/apk/keys/alpine-devel@lists.alpinelinux.org-58199dcc.rsa.pub
usr/share/apk/keys/alpine-devel@lists.alpinelinux.org-58cbb476.rsa.pub
usr/share/apk/keys/alpine-devel@lists.alpinelinux.org-58e4f17d.rsa.pub
usr/share/apk/keys/alpine-devel@lists.alpinelinux.org-5e69ca50.rsa.pub
usr/share/apk/keys/alpine-devel@lists.alpinelinux.org-60ac2099.rsa.pub
usr/share/apk/keys/alpine-devel@lists.alpinelinux.org-6165ee59.rsa.pub
usr/share/apk/keys/alpine-devel@lists.alpinelinux.org-61666e3f.rsa.pub
usr/share/apk/keys/alpine-devel@lists.alpinelinux.org-616a9724.rsa.pub
usr/share/apk/keys/alpine-devel@lists.alpinelinux.org-616abc23.rsa.pub
usr/share/apk/keys/alpine-devel@lists.alpinelinux.org-616ac3bc.rsa.pub
usr/share/apk/keys/alpine-devel@lists.alpinelinux.org-616adfeb.rsa.pub
usr/share/apk/keys/alpine-devel@lists.alpinelinux.org-616ae350.rsa.pub
usr/share/apk/keys/alpine-devel@lists.alpinelinux.org-616db30d.rsa.pub
usr/share/apk/keys/alpine-devel@lists.alpinelinux.org-66ba20fe.rsa.pub
usr/share/apk/keys/aarch64/alpine-devel@lists.alpinelinux.org-58199dcc.rsa.pub
usr/share/apk/keys/aarch64/alpine-devel@lists.alpinelinux.org-616ae350.rsa.pub
usr/share/apk/keys/armhf/alpine-devel@lists.alpinelinux.org-524d27bb.rsa.pub
usr/share/apk/keys/armhf/alpine-devel@lists.alpinelinux.org-616a9724.rsa.pub
usr/share/apk/keys/armv7/alpine-devel@lists.alpinelinux.org-524d27bb.rsa.pub
usr/share/apk/keys/armv7/alpine-devel@lists.alpinelinux.org-616adfeb.rsa.pub
usr/share/apk/keys/loongarch64/alpine-devel@lists.alpinelinux.org-66ba20fe.rsa.pub
usr/share/apk/keys/mips64/alpine-devel@lists.alpinelinux.org-5e69ca50.rsa.pub
usr/share/apk/keys/ppc64le/alpine-devel@lists.alpinelinux.org-58cbb476.rsa.pub
usr/share/apk/keys/ppc64le/alpine-devel@lists.alpinelinux.org-616abc23.rsa.pub
```

"decentralized" builders

- Each builder has its own signing key
- All builders work independently of each other

Sequential

Sequential

- Each architecture only has a single builder, building one package at a time

Sequential

- Each architecture only has a single builder, building one package at a time
- Works fine on architectures with powerful machines

Sequential

- Each architecture only has a single builder, building one package at a time
- Works fine on architectures with powerful machines
- ...but not so well on architectures like riscv64, where hardware is not quite there

Manually deployed

Manually deployed

- Each release requires someone to set up the builders

Manually deployed

- Each release requires someone to set up the builders
- Only 2 people who know how to do it

Manually deployed

- Each release requires someone to set up the builders
- Only 2 people who know how to do it
- Unreproducible failures

Fragile

Fragile

- Pieces held together by three scripting languages

Fragile

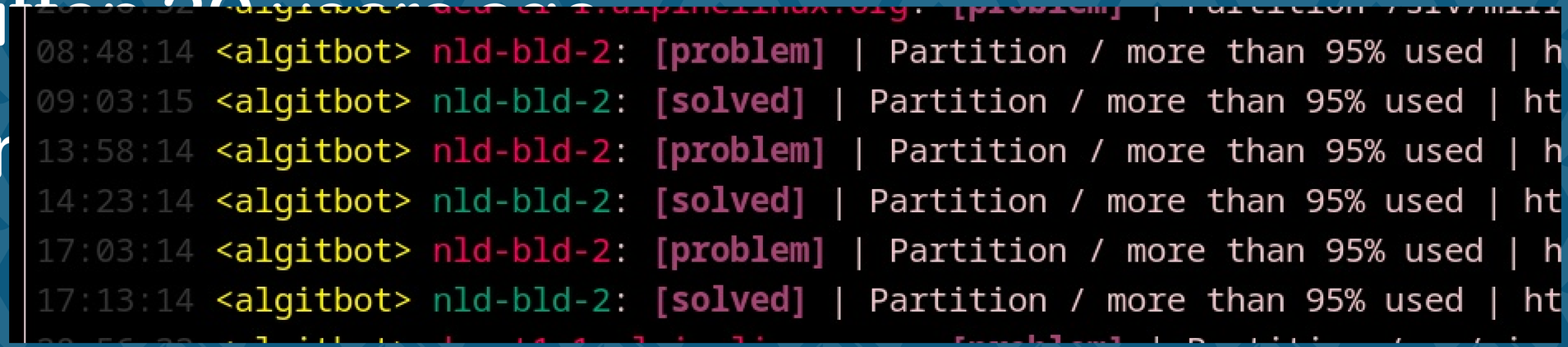
- Pieces held together by three scripting languages
- Written 20 years ago

Fragile

- Pieces held together by three scripting languages
- Written 20 years ago
- Can randomly go out of disk space

Fragile

- Pieces held together by three scripting languages
- Writ
- Car



Fragile

- Pieces held together by three scripting languages
- Written 20 years ago
- Can randomly go out of disk space
- Needs system dependencies

Inconsistent

Inconsistent

- Cyclic dependencies = Undefined behaviour

Inconsistent

- Cyclic dependencies = Undefined behaviour
- Same package might be built with different dependency versions

Inconsistent

- Cyclic dependencies = Undefined behaviour
- Same package might be built with different dependency versions
- Missing pkgrel bumps

Inconsistent

- Cyclic dependencies = Undefined behaviour
- Same package might be built with different dependency versions
- Missing pkgrel bumps
- Repo-categorization inconsistencies

So… how do we fix that?

Introducing: builder director

Introducing: builder director

- Central point of ~~failure~~ trust

Introducing: builder director

- Central point of ~~failure~~ trust
- Director can hold the keys and receive unsigned packages from the corresponding builders, then sign them and generate the index

Introducing: builder director

- Central point of ~~failure~~ trust
- Director can hold the keys and receive unsigned packages from the corresponding builders, then sign them and generate the index
- Each architecture can have multiple builders

Single codebase

Single codebase

- Both director and builder use the same code

Single codebase

- Both director and builder use the same code
- No more "C executable calls shell script calls Lua"

Single codebase

- Both director and builder use the same code
- No more "C executable calls shell script calls Lua"
- ...Rust?

Dynamic analysis

Dynamic analysis

- For every package, check for changed dependencies

Dynamic analysis

- For every package, check for changed dependencies
- Parse APKBUILDs to look for possible providers

Dynamic analysis

- For every package, check for changed dependencies
- Parse APKBUILDs to look for possible providers
- Generate a tree of dependencies

Dynamic analysis

- For every package, check for changed dependencies
- Parse APKBUILDs to look for possible providers
- Generate a tree of dependencies
- Detect cyclic dependencies and fail

Dynamic analysis

- For every package, check for changed dependencies
- Parse APKBUILDs to look for possible providers
- Generate a tree of dependencies
- Detect cyclic dependencies and fail
- => deterministic

Hotpluggable

Hotpluggable

- Spin up builders just like CI runners

Hotpluggable

- Spin up builders just like CI runners
- All you need is a config file and one executable

Hotpluggable

- Spin up builders just like CI runners
- All you need is a config file and one executable
- Build jobs are split into isolated small chunks,
  so that any builder can pick them up

Package “difficulty”

Package “difficulty”

- Collect and save system metrics during build (average load, peak RAM usage, used disk space)

Package “difficulty”

- Collect and save system metrics during build (average load, peak RAM usage, used disk space)
- On next version, assume 1.2x needed resources

Package “difficulty”

- Collect and save system metrics during build (average load, peak RAM usage, used disk space)
- On next version, assume 1.2x needed resources
- When data is missing, assume nothing and run on most powerful builder without concurrent builds

Atomic builds

Atomic builds

- Build every commit until there’s no more stuff left

Atomic builds

- Build every commit, until there's no more stuff left
- If $pkgname-$pkgver-r$pkgrel already built, error out

Atomic builds

- Build every commit, until there's no more stuff left
- If $pkgname-$pkgver-r$pkgrel already built, error out
- Build with the real latest version of dependencies

Atomic builds

- Build every commit, until there's no more stuff left
- If $pkgname-$pkgver-r$pkgrel already built, error out
- Build with the real latest version of dependencies
- If a package fails, check whether there's a newer commit

Shallow Git fetches

Shallow Git fetches

- Current aports checkout is ~600 MiB

Shallow Git fetches

- Current aports checkout is ~600 MiB
- Request only needed Git trees and blobs

Shallow Git fetches

- Current aports checkout is ~600 MiB
- Request only needed Git trees and blobs

```
59  def make_req(command: str, args: list[str]):
60      data = [
61          [
62              f'command={command}\n',
63              'agent=git/2.46.0',
64              'object-format=sha1',
65          ],
66          list(map(lambda arg: arg + '\n', args))
67      ]
68      return serialize_git_pack(data).encode()
```

Improve rootbld upstream?

Improve rootbld upstream?

- Use random username for system user

Improve rootbld upstream?

- Use random username for system user
- Map all users and groups

Improve rootbld upstream?

- Use random username for system user
- Map all users and groups
- Make sure abuild uses the correct version of itself

Improve rootbld upstream?

- Use random username for system user
- Map all users and groups
- Make sure abuild uses the correct version of itself
- Reduce hardcoded stuff

Improve rootbld upstream?

- Use random username for system user
- Map all users and groups
- Make sure abuild uses the correct version of itself
- Reduce hardcoded stuff
- ...maybe rootbld should not be part of abuild?

Implementation

Implementation

- Deploy side-by-side building edge with new infra

Implementation

- Deploy side-by-side building edge with new infra
- Verify package metadata to catch mismatches early

Future ideas

- Use idle builders to test reproducibility, a'la reproducible-builds.org

Questions?