FOSDEM'23

NTT

Bit-for-bit reproducible builds with Dockerfile

Deterministic timestamps and deterministic apt-get

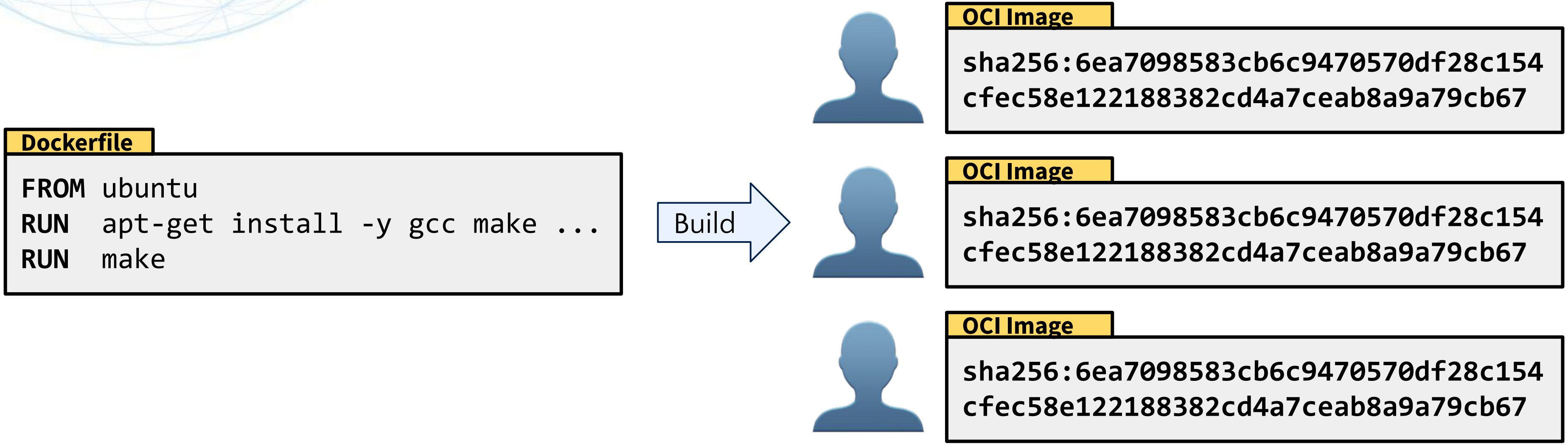
Demo:
https://github.com/reproducible-containers/repro-get/releases/tag/v0.3.0

Akihiro Suda, NTT

What are reproducible builds?

- Same source, same binary
- Attestable by anybody, at anytime

But often needs a specific (virtual) machine



Dockerfile

```
FROM ubuntu
RUN  apt-get install -y gcc make ...
RUN  make
```

OCI Image

```
sha256:6ea7098583cb6c9470570df28c154
cfec58e122188382cd4a7ceab8a9a79cb67
```

OCI Image

```
sha256:6ea7098583cb6c9470570df28c154
cfec58e122188382cd4a7ceab8a9a79cb67
```

OCI Image

```
sha256:6ea7098583cb6c9470570df28c154
cfec58e122188382cd4a7ceab8a9a79cb67
```

OCI = Open Container Initiative

Why do we need reproducible builds?

- Because we want to verify the ***actual*** source code of the binary, not the ***claimed*** source code

- ***actual*** != ***claimed***, when the build env is compromised, or when the developer is malicious

- If the builds are reproducible, we can be sure that ***actual*** == ***claimed***

Not a panacea...

- Reproducibility has nothing to do with whether the source code is safe to use
- The source code may still contain malicious codes
- Reproducible builds make sense only when you review the source code

Why couldn't we make them reproducible?

- Timestamps
  - Timestamps of the files in tar layers
  - Timestamps in OCI Image Spec JSONs (“`org.opencontainers.image.created`”, etc.)
- “aptgettable” packages
  - The package version changes on every invocation of `apt-get`, `dnf`, etc.
- Filesystem characteristics
  - Hardlinks, xattrs, …

BuildKit v0.11 supports reproducible builds! 🎉

- **BuildKit**: a modern image building framework made for Docker/Moby
  - Embedded in the Docker daemon since Docker 18.06
  - Can be also used with Kubernetes, nerdctl, Podman, etc.

- v0.11 (Jan 2023) contains built-in support for reproducing timestamps
  - Thanks to Tõnis Tiigi (Docker) for the large portion

- Still needs very complex Dockerfile
  - v0.12 will require less complex Dockerfiles

https://github.com/moby/buildkit

Reproducing timestamps

- The `SOURCE_DATE_EPOCH` build arg can be used for specifying the UNIX epoch

```
$ buildctl build --opt build-arg:SOURCE_DATE_EPOCH=<uint64> ...
```

- Conforms to: https://reproducible-builds.org/specs/source-date-epoch/
- Usually set to `$(git log -1 --pretty=%ct)`

- The build arg is exposed to the “`RUN`” containers as an env var
- The build arg is also consumed by BuildKit itself for the timestamps in the OCI JSONs (but not for the file timestamps in the tar layers, in v0.11)

https://github.com/moby/buildkit/blob/v0.11/docs/build-repro.md

Caveats in v0.11 (Being resolved in PR #3560, targeted for v0.12)

- The file timestamps in the tar layers need to be explicitly `touch`-ed

```
ARG SOURCE_DATE_EPOCH
RUN find $( ls / | grep -E -v "^(dev|mnt|proc|sys)$" ) \
  -newermt "@${SOURCE_DATE_EPOCH}" -writable -xdev \
  | xargs touch --date="@${SOURCE_DATE_EPOCH}" --no-dereference
```

- The layers have to be squashed to remove unreproducible overlayfs whiteouts

```
FROM scratch
COPY --from=0 / /
```

- Mount points can be created only under `/dev` (tmpfs)

```
RUN --mount=type=cache,target=/dev/.cache ...
```

- Hardlinks are not reproducible depending on the filesystem snapshotter

https://github.com/moby/buildkit/blob/v0.11/docs/build-repro.md

https://github.com/moby/buildkit/pull/3560

Reproducing packages

- “aptgettable” package versions are hard to reproduce
- Most distros do not retain old packages
- Debian retains old packages (thank you!), but not mirrored widely

/etc/apt/sources.list

```
deb http://snapshot.debian.org/archive/debian/20230101T091029Z/ bullseye main
```

  - Too much load on the central `snapshot.debian.org`
  - Can’t be used in CI practically, due to slowness and flakiness
- The situation is similar for Fedora and ArchLinux

repro-get: decentralized & reproducible apt/dnf/apk/pacman...

- Cryptographically locks the package versions with `SHA256SUMS`

SHA256SUMS-amd64

```
35b1508eeee9c1dfba798c4c04304ef0f266990f936a51f165571edf53325cbc  pool/main/h/hello/hello_2.10-2_amd64.deb
```

- Blobs can be fetched from several places to avoid overloading

`http://deb.debian.org/debian/{{.Name}}` (Fast, ephemeral)

`http://debian.notset.fr/snapshot/by-hash/SHA256/{{.SHA256}}` (Slow, persistent)

`oci://example.com/oras-image@sha256:{{.SHA256}}`

`http://ipfs.io/ipfs/{{.CID}}`

- Supports Debian, Ubuntu, Fedora, Alpine, and ArchLinux

https://github.com/reproducible-containers/repro-get

repro-get: decentralized & reproducible apt/dnf/apk/pacman...

```
$ repro-get hash generate >SHA256SUMS-amd64.old
$ apt-get install -y hello
$ repro-get hash generate --dedupe=SHA256SUMS-amd64.old >SHA256SUMS-amd64
```

```
$ cat SHA256SUMS-amd64
35b1508eeee9c1dfba798c4c04304ef0f266990f936a51f165571edf53325cbc  pool/main/h/hello/hello_2.10-2_amd64.deb

$ repro-get install SHA256SUMS-amd64
(001/001) hello_2.10-2_amd64.deb Downloading from
http://debian.notset.fr/snapshot/by-hash/SHA256/35b1508eeee9c1dfba798c4c04304ef0f266990f936a51f165571edf53325cbc
...
Preparing to unpack .../35b1508eeee9c1dfba798c4c04304ef0f266990f936a51f165571edf53325cbc ...
Unpacking hello (2.10-2) ...
Setting up hello (2.10-2) ...
```

https://github.com/reproducible-containers/repro-get

Demo

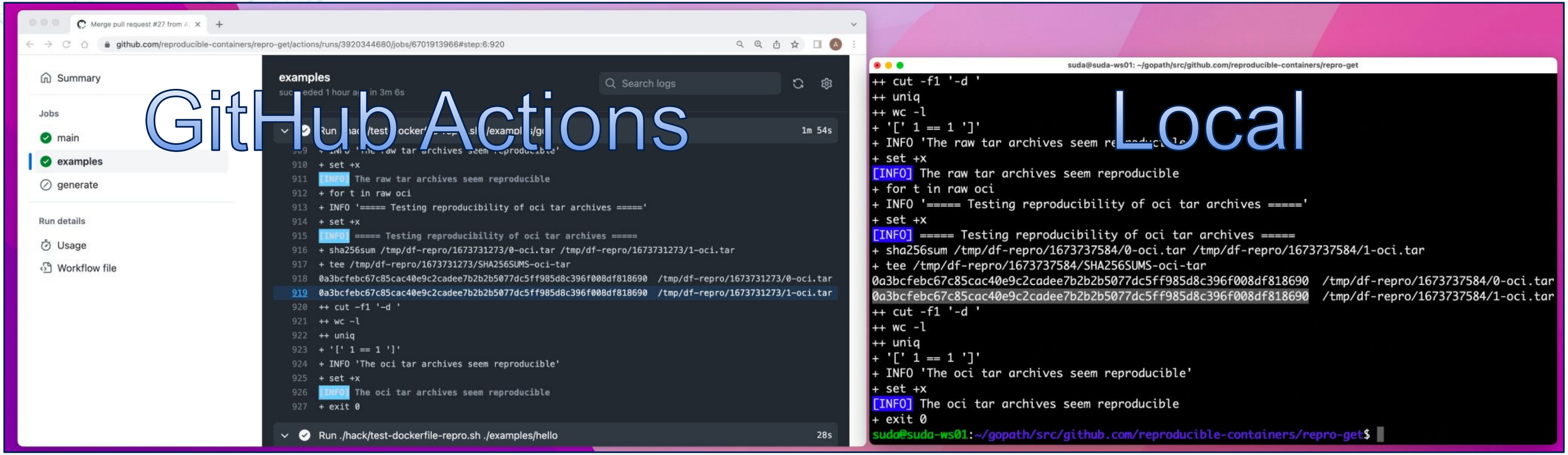
https://github.com/reproducible-containers/repro-get/releases/tag/v0.3.0

NTT

```
$ docker run -d --name buildkitd --privileged moby/buildkit:v0.11.0
$ docker cp buildkitd:/usr/bin/buildctl /usr/local/bin/buildctl
$ export BUILDKIT_HOST=docker-container://buildkitd
$ ./hack/test-dockerfile-repro.sh examples/gcc
...
0a3bcfebc67c85cac40e9c2cadee7b2b2b5077dc5ff985d8c396f008df818690  /.../0-oci.tar
0a3bcfebc67c85cac40e9c2cadee7b2b2b5077dc5ff985d8c396f008df818690  /.../1-oci.tar
```

BuildKit version MUST be pinned

The filesystem (ext4) and the OS version (Ubuntu 22.04) SHOULD be pinned too



GitHub Actions

Merge pull request #27 from A
github.com/reproducible-containers/repro-get/actions/runs/3920344680/jobs/6701913966#step:6:920
Summary
Jobs
main
examples
generate
Run details
Usage
Workflow file
Search logs

examples

succeeded 1 hour ago in 3m 6s

Run ./hack/test-dockerfile-repro.sh ./examples/gcc 1m 54s

```
+ INFO 'The raw tar archives seem reproducible'
+ set +x
[INFO] The raw tar archives seem reproducible
+ for t in raw oci
+ INFO '===== Testing reproducibility of oci tar archives ====='
+ set +x
[INFO] ===== Testing reproducibility of oci tar archives =====
+ sha256sum /tmp/df-repro/1673731273/0-oci.tar /tmp/df-repro/1673731273/1-oci.tar
+ tee /tmp/df-repro/1673731273/SHA256SUMS-oci-tar
0a3bcfebc67c85cac40e9c2cadee7b2b2b5077dc5ff985d8c396f008df818690  /tmp/df-repro/1673731273/0-oci.tar
0a3bcfebc67c85cac40e9c2cadee7b2b2b5077dc5ff985d8c396f008df818690  /tmp/df-repro/1673731273/1-oci.tar
++ cut -f1 '-d '
++ wc -l
++ uniq
+ '[' 1 == 1 ']'
+ INFO 'The oci tar archives seem reproducible'
+ set +x
[INFO] The oci tar archives seem reproducible
+ exit 0
```

Run ./hack/test-dockerfile-repro.sh ./examples/hello 28s

Local

suda@suda-ws01: ~/gopath/src/github.com/reproducible-containers/repro-get

```
++ cut -f1 '-d '
++ uniq
++ wc -l
+ '[' 1 == 1 ']'
+ INFO 'The raw tar archives seem reproducible'
+ set +x
[INFO] The raw tar archives seem reproducible
+ for t in raw oci
+ INFO '===== Testing reproducibility of oci tar archives ====='
+ set +x
[INFO] ===== Testing reproducibility of oci tar archives =====
+ sha256sum /tmp/df-repro/1673737584/0-oci.tar /tmp/df-repro/1673737584/1-oci.tar
+ tee /tmp/df-repro/1673737584/SHA256SUMS-oci-tar
0a3bcfebc67c85cac40e9c2cadee7b2b2b5077dc5ff985d8c396f008df818690  /tmp/df-repro/1673737584/0-oci.tar
0a3bcfebc67c85cac40e9c2cadee7b2b2b5077dc5ff985d8c396f008df818690  /tmp/df-repro/1673737584/1-oci.tar
++ cut -f1 '-d '
++ wc -l
++ uniq
+ '[' 1 == 1 ']'
+ INFO 'The oci tar archives seem reproducible'
+ set +x
[INFO] The oci tar archives seem reproducible
+ exit 0
suda@suda-ws01:~/gopath/src/github.com/reproducible-containers/repro-get$
```

https://twitter.com/_AkihiroSuda_/status/1614401664009789440

Future works

- Simplify Dockerfile
- Find an easier way to cache old packages locally
- Interoperability with `xx-apt` and `xx-apk` for cross-compilation
- Interoperability with SLSA Provenances
- Single-click attestation of reproducibility

Wrap-up

- Reproducible build helps attesting the true origin of the binary
- Challenges: non-deterministic timestamps, package versions, etc.
- BuildKit v0.11 adds preliminary support for `SOURCE_DATE_EPOCH`
- repro-get reproduces the package versions with `SHA256SUMS`

https://github.com/moby/buildkit/blob/v0.11/docs/build-repro.md

https://github.com/reproducible-containers/repro-get