BuildKit – new image build engine

Dave Chen, ARM China
What’s BuildKit?

A toolkit to build docker image, with brilliant new features such as concurrent build, efficient caching, multi-platform images etc.

- Automatic garbage collection
- Extendable frontend formats
- Concurrent dependency resolution
- Efficient instruction caching
- Build cache import/export
- Nested build job invocations
- Distributable workers
- Multiple output formats
- Pluggable architecture
- Execution without root privileges
- Rootless execution

https://github.com/moby/buildkit
BuildKit backends

- Containerd – Open Container runtime
  - Snapshotters
  - Distribution
  - Garbage collection

- OCI(runc) – Open Container Initiative
  - Process execution with OCI Runtime specification
  - Build result can be exported with OCI images specification
Concurrent Build
FROM golang AS stage0
...
RUN go build -o /main main.go

FROM gcc:latest AS stage1
...
RUN gcc -o /hello hello.c

FROM debian AS stage2
COPY --from=stage0 /main /
COPY --from=stage1 /hello /
BuildKit LLB

- LLB (low-level builder) is DAG-structured graph, encoded in protobuf, typically compiled from Dockerfile

- LLB defines a content-addressable dependency graph that can be used to put together very complex build definitions.
  - Efficient caching
  - Concurrent execution
  - Extensible for data mounting and nested invocation
FROM golang AS stage0
...  
RUN go build -o /main main.go

FROM gcc:latest AS stage1
...  
RUN gcc -o /hello hello.c

FROM debian AS stage2
COPY --from=stage0 /main /
COPY --from=stage1 /hello /
BuildKit LLB

"Op": { [0]
  ...
  "Digest": "sha256:***048981d78530d1fdbc76f90541f06d",
  "description": {
    "llb.customname": "[stage0 3/3] RUN go build -o /main main.go"
  }
  ...
  "Op": { [2]
    "inputs": [
      {
        "digest": "sha256:***048981d78530d1fdbc76f90541f06d",
      },
      ...
      "Digest": "sha256:***95aadc6a31f34e8e192b4d7eb",
      "OpMetadata": {
        "description": {
          "llb.customname": "[stage2 2/3] COPY --from=stage0 /main /
        }
      }
    }
  }

Op: represents a vertex of the LLB DAG

SourceOp: specifies a source such as build contexts and image

ExecOp: executes a command in a container

FileOp: operations on the file, such as copy or remove
Cache
BuildKit – cache [1]

Legacy Docker build:

- Modifying line N always invalidates the cache for line(N+1)

```
FROM debian
EXPOSE 80
RUN apt update && apt install -y git
```
BuildKit build:

- Instruction of line(N+1) is cached and no more time is consumed.

```
buildctl build --frontend dockerfile.v0 ...
[+] Building 4.3s (6/6) FINISHED
 => [internal] load build definition from Dockerfile 0.3s
 ..
 => => resolve
 docker.io/library/debian@sha256:2f04d3d33b6027bb74ecc81397abe780649ec89f1a2af18d7022737d0482 0.0s
 => CACHED [2/2] RUN apt update && apt install -y git 0.0s
 => exporting to image 0.6s
 => => exporting layers 0.1s
 => => exporting manifest
sha256:aa5a2865665d3b5e363e255773f431994d5d7ca653e177869843c5a7575d237a 0.2s
 ..
```
BuildKit – cache [2]

- Allows to cache directories for compilers and package managers

```dockerfile
# syntax = docker/dockerfile:experimental
...
RUN --mount=type=cache,target=/root/.cache/go-build go build ...
...

# syntax = docker/dockerfile:experimental
FROM ubuntu
RUN --mount=type=cache,target=/var/cache/apt apt update apt install -y gcc
...
```
BuildKit – cache [3]

- Exporting / Importing build cache to / from registry

```bash
buildctl build ... --export-cache type=registry,ref=localhost:5000/myrepo:buildcache
buildctl build ... --import-cache type=registry,ref=localhost:5000/myrepo:buildcache
```

- Exporting / Importing build cache to / from filesystem

```bash
buildctl build ... --export-cache type=local,dest=path/to/output-dir
buildctl build ... --import-cache type=local,src=path/to/input-dir
```
Multi-platform images
BuildKit – multi-platform images

- Docker

```bash
> docker manifest create new-list-ref-name image-ref [image-ref...]  
> docker manifest annotate new-list-ref-name image-ref-arm --os linux --arch arm  
> docker manifest annotate new-list-ref-name image-ref-amd64 --os linux --arch amd64  
> docker manifest push new-list-ref-name
```

- BuildKit Dockerfile

```
FROM --platform=$BUILDPLATFORM golang:alpine AS build

ARG TARGETOS
ARG TARGETARCH

ENV GOOS=$TARGETOS GOARCH=$TARGETARCH
WORKDIR /home/dave/demo/multi-arch-buildkit
COPY main.go .
RUN go build -o /bin/hello.

FROM scratch AS release
COPY --from=build /bin/hello .

buildctl build ... --opt platform=linux/amd64,linux/arm64
```
BuildKit – multi-platform images

- Enable the kernel support for miscellaneous binary formats (binfmt-misc)
- BuildKit detects which platforms are supported for each of the worker when the daemon is started.
- # buildkitd --debug --oci-worker=false --containerd-worker=true
- # buildctl debug workers

<table>
<thead>
<tr>
<th>ID</th>
<th>PLATFORMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>8j7ttc...</td>
<td>linux/amd64, linux/arm64, linux/386</td>
</tr>
<tr>
<td>3fkpjy2...</td>
<td>linux/amd64, linux/arm64, linux/386</td>
</tr>
</tbody>
</table>

- buildctl build --frontend dockerfile.v0 --local dockerfile=. --local context=. --output type=image, name=jungler/multiarch, push=true --exporter-opt name=jungler/multiarch --opt platform=linux/amd64, linux/arm64

Demo:

Online:
https://www.loom.com/share/0784730f97cb431c99ba598608a6a9f6
Use BuildKit
BuildKit – BuildKit in Docker

- Docker
  - Integrated into “docker build” v18.09+
  - Opt-in
    - export DOCKER_BUILDKIT=1

- Docker buildx
  - Available as an experimental feature since 19.03
  - Next generation Build command from Docker
  - Familiar Docker UI + full BuildKit
  - With container driver, works with any version of Docker engine
  - etc.
BuildKit – BuildKit in K8S

- Rootless BuildKit can be deployed on K8S
- Useful for CI/CD pipeline
Buildkit – BuildKit in K8S

- Benefit from the rich resource of k8s cluster
BuildKit – BuildKit in K8S

$ kubectl run --generator=run-pod/v1 \
   --image=moby/buildkit:master-rootless \
   bk -- --oci-worker-no-process-sandbox

$ export BUILDKIT_HOST=kube-pod://bk

$ buildctl build --frontend dockerfile.v0 ...
Performance Comparison
BuildKit - performance

Based on github.com/moby/moby Dockerfile, master branch.
Time for full build from empty status, smaller is better (https://t.co/aUKqQCVmXa).
BuildKit - performance

RUN --mount moby/buildkit Dockerfile: time to binary rebuild after code change

- v18.03: 139.3 sec (33.1x slower)
- v18.03 + commented out all unused commands: 41.4 sec (10.6x slower)
- BuildKit: 31.8 sec (8.1x slower)
- BuildKit + RUN --mount: 3.9 sec
Thank you

Join Linaro to accelerate deployment of your Arm-based solutions through collaboration

contactus@linaro.org
Backup
binfmt_misc

• # echo -1 > qemu-aarch64 (disable invalid entry)

• Set “-F” flag
# echo '
:aarch64:M:\x7fELF\x02\x01\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x02\x00\xb7:\xff\xff\xff\xff\xff\xff\xff\xff\xff\x00\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xf
rootless mode

BuildKit can be executed as a non-root user to protect the host from potential BuildKit vulns.

```
$ rootlesskit bash
$ buildkitd
$ buildctl --addr unix:///run/user/1001/buildkit/buildkitd.sock build --frontend dockerfile.v0 --local dockerfile=. --local context=. --output type=oci > /home/dave/output.tar
```
RUN --mount=type=secret

Allow accessing private keys without leaking them into the image

```
# syntax = docker/dockerfile:experimental
...
RUN --mount=type=secret,id=aws,target=/root/.aws/credentials aws s3 cp s3://... ...
```

```
buildctl build --frontend=dockerfile.v0 --local context= --local dockerfile=. --secret id=aws,src=$HOME/.aws/credentials
```
BuildKit - performance

Fresh build with --cache-from remote source

v18.03: 0:05:28
BuildKit: 0:00:36

9.1x faster