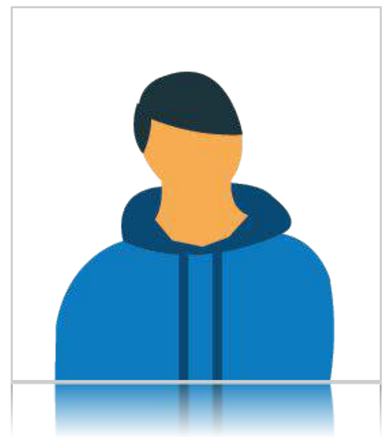
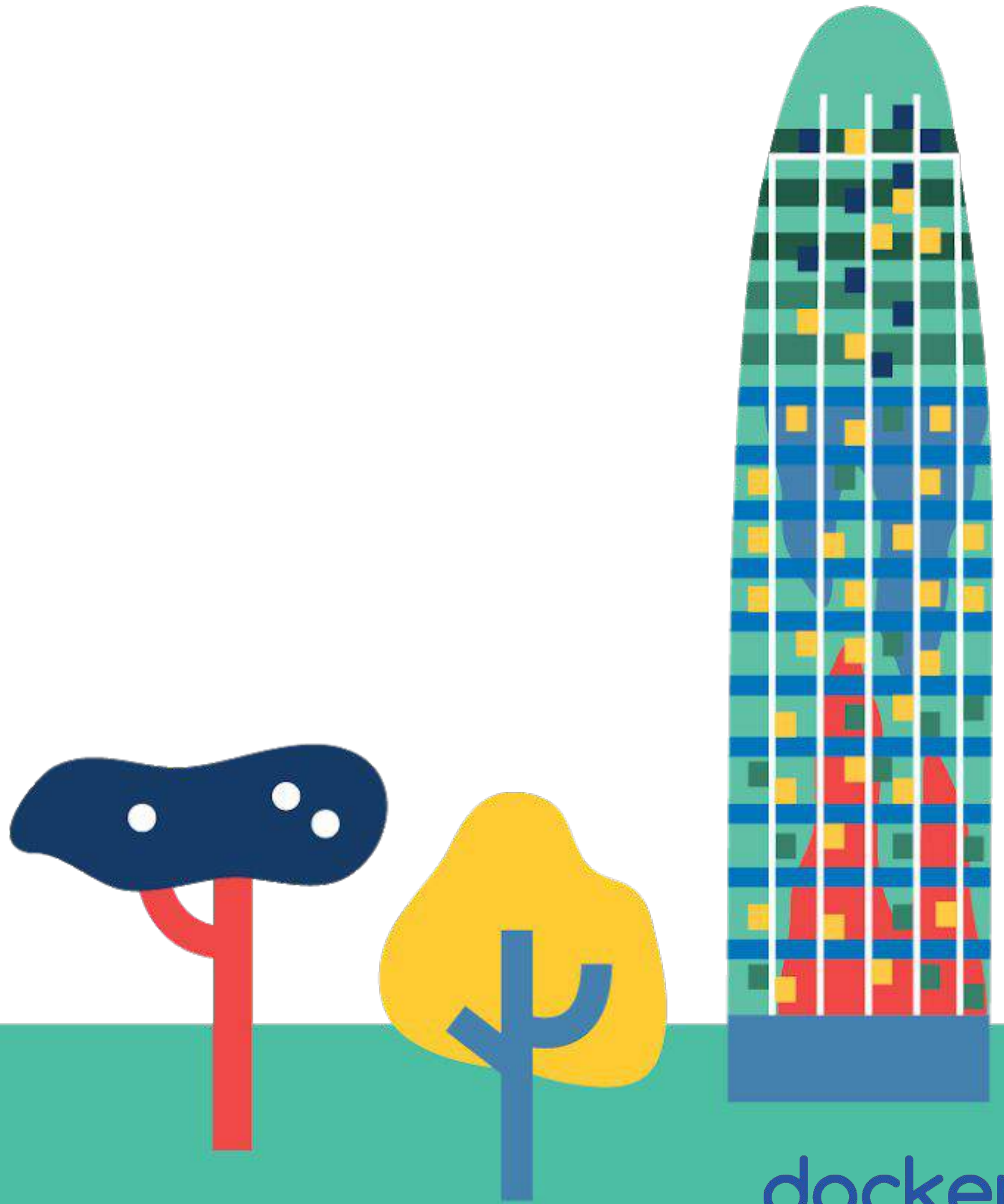
Supercharged Docker Build with BuildKit

Tõnis Tiigi
@tonistiigi

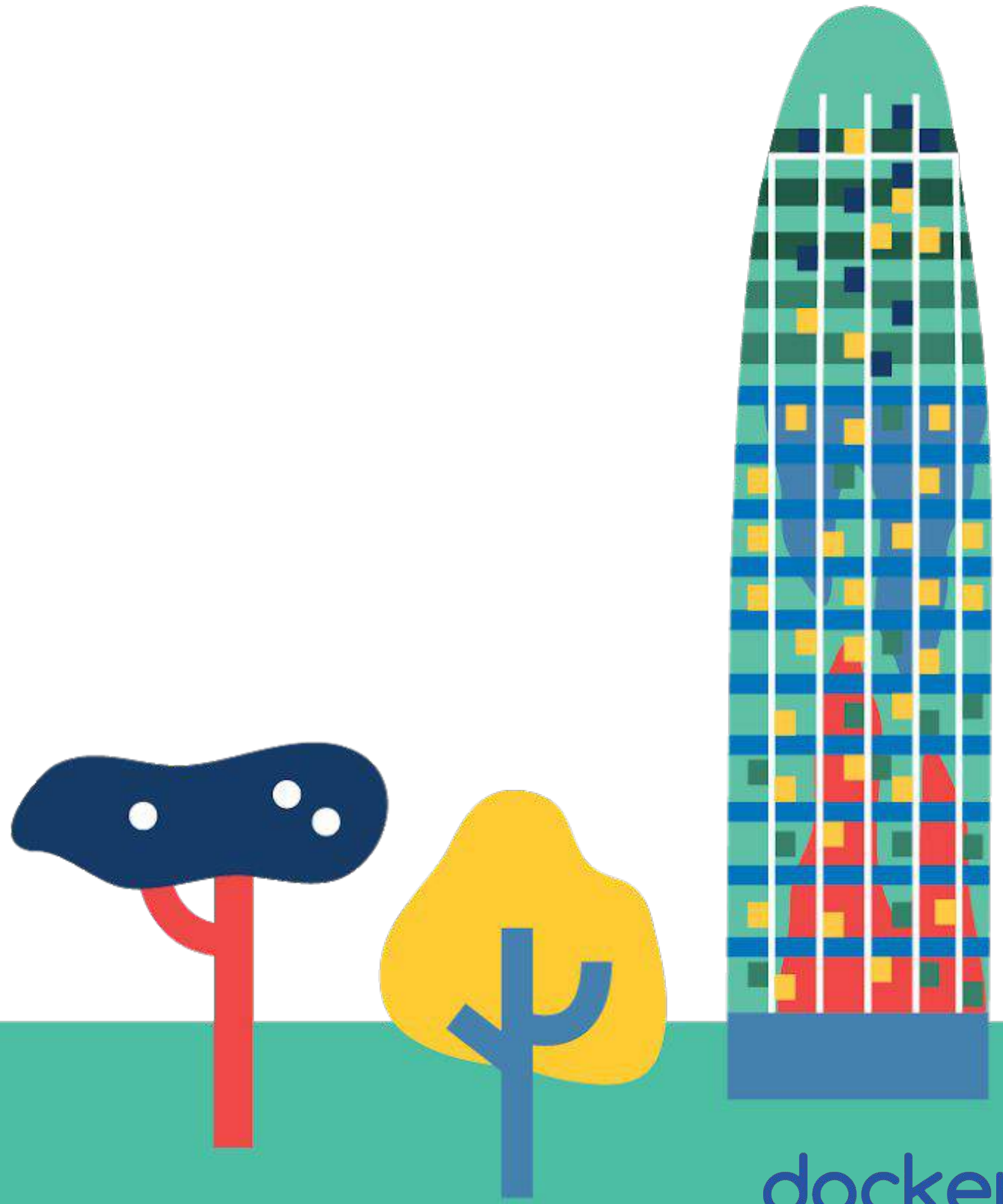

Ian Campbell
@ijc



dockercon 18 EUROPE

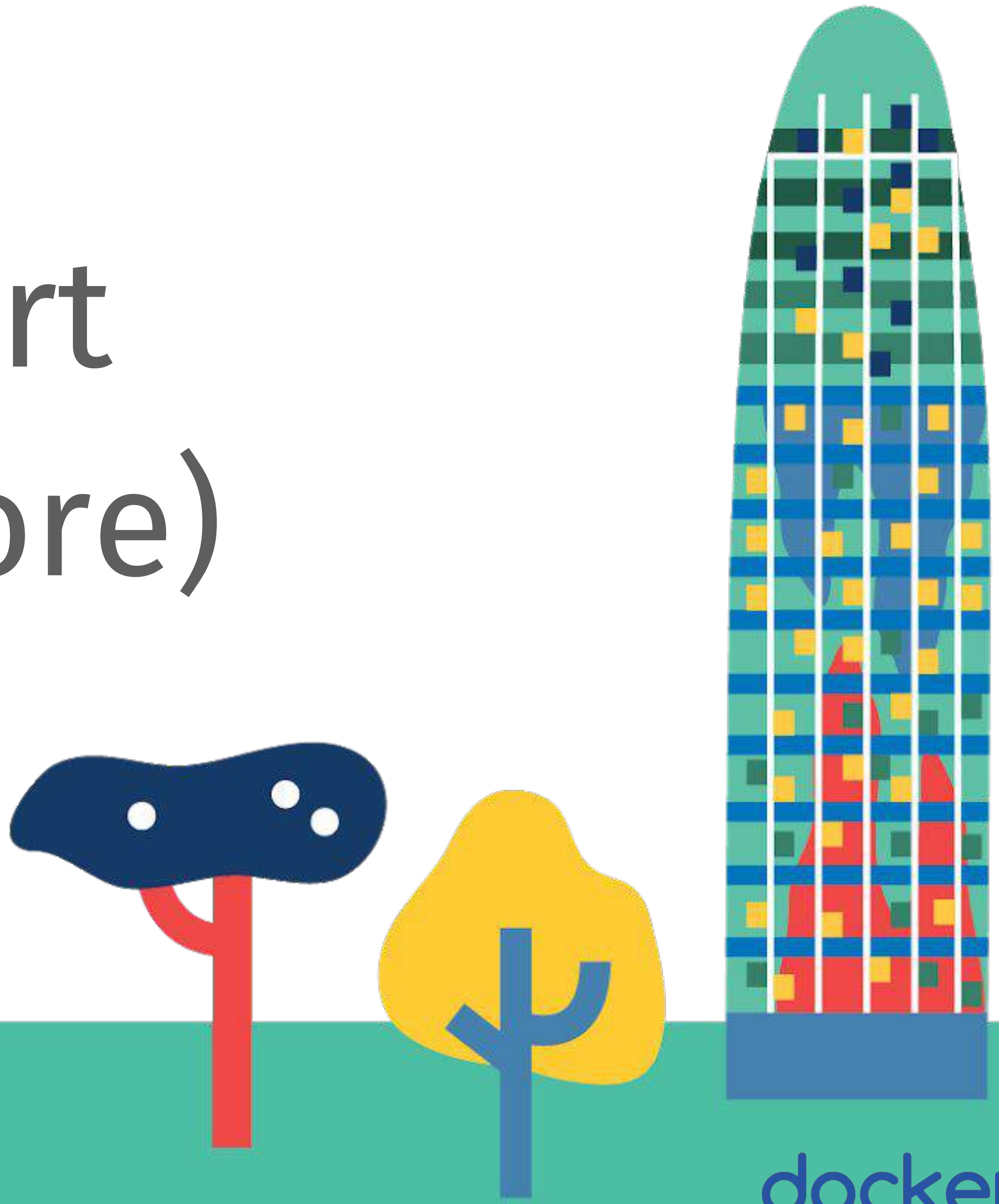
```
> docker build .
```

Dockerfile



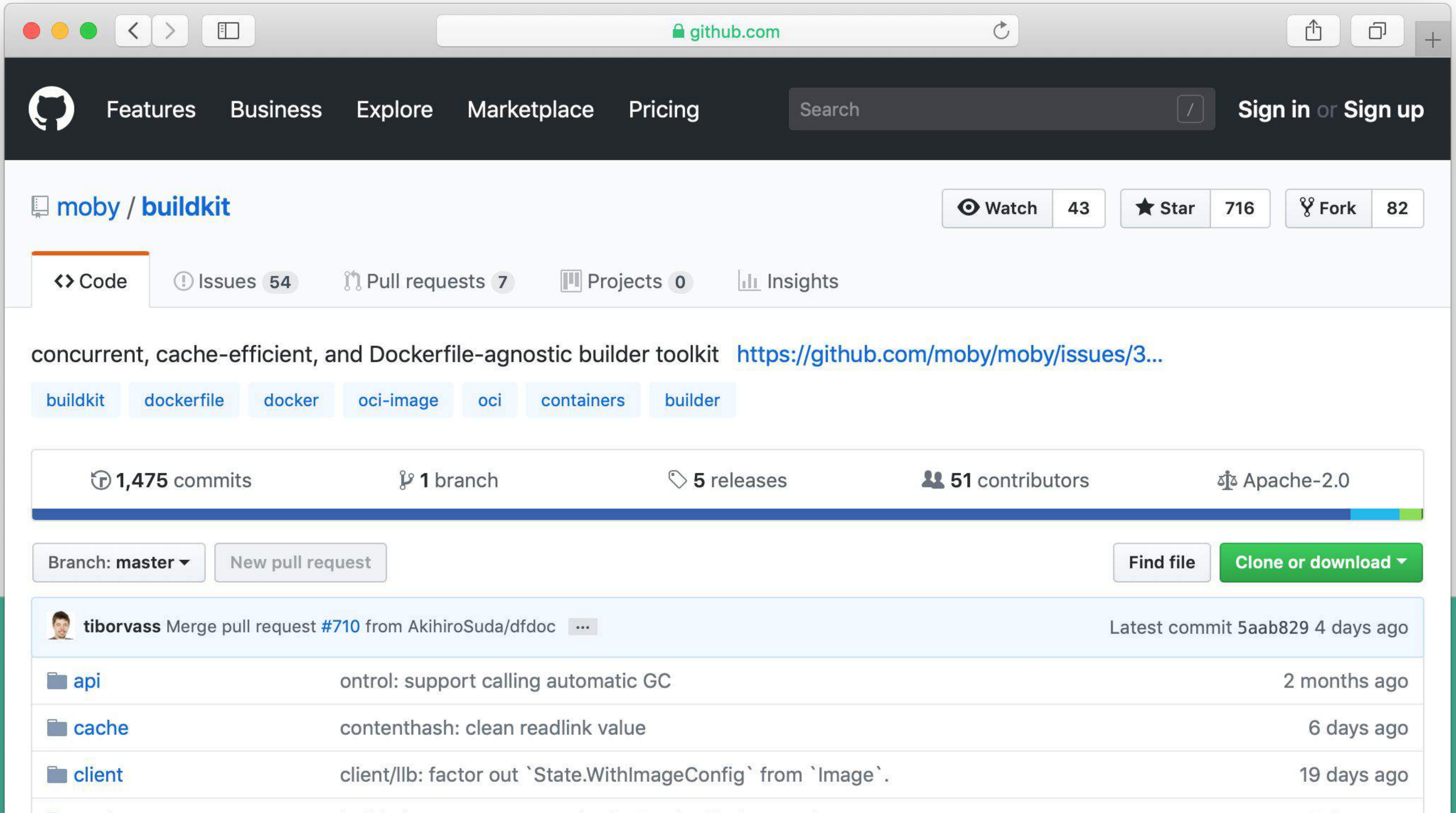
Docker v18.09

First release with BuildKit support
(experimental not needed anymore)



BuildKit

https://github.com/moby/buildkit



moby / buildkit

concurrent, cache-efficient, and Dockerfile-agnostic builder toolkit https://github.com/moby/moby/issues/3...

buildkit dockerfile docker oci-image oci containers builder

1,475 commits | 1 branch | 5 releases | 51 contributors | Apache-2.0

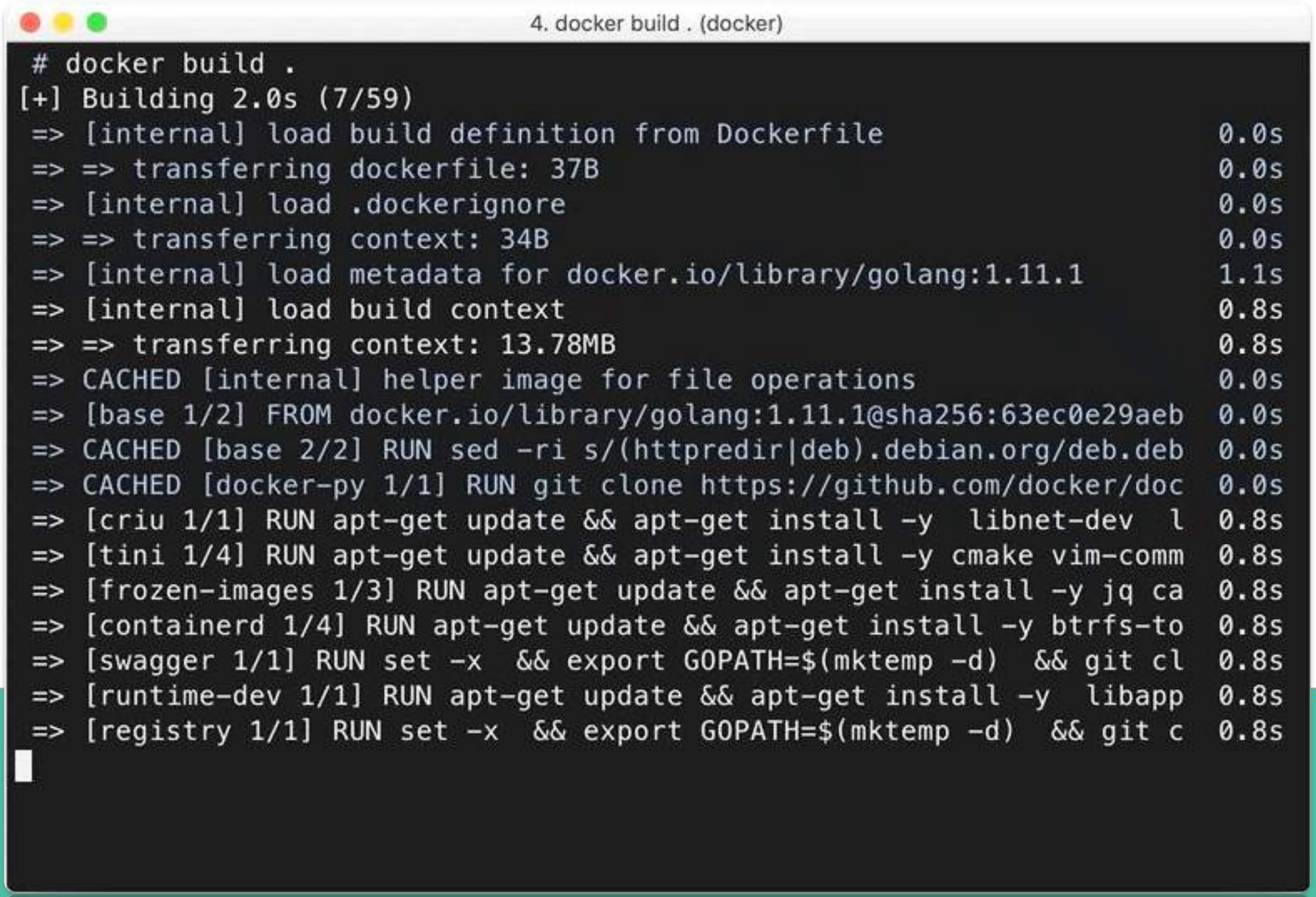
tiborvass Merge pull request #710 from AkihiroSuda/dfdoc … Latest commit 5aab829 4 days ago

api	ontrol: support calling automatic GC	2 months ago
cache	contenthash: clean readlink value	6 days ago
client	client/llb: factor out `State.WithImageConfig` from `Image`.	19 days ago

Enabling BuildKit support

```
export DOCKER_BUILDKIT=1
```

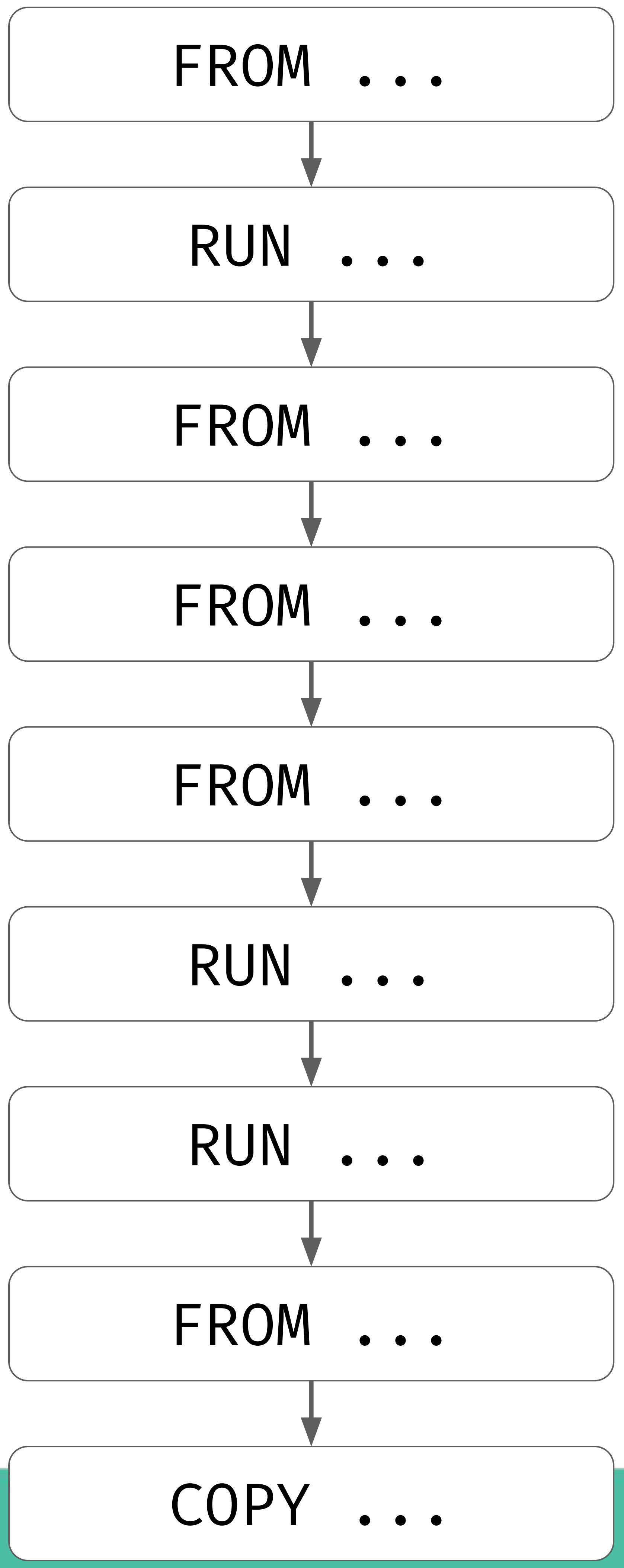
... or enable on daemon level in `/etc/docker/daemon.json`

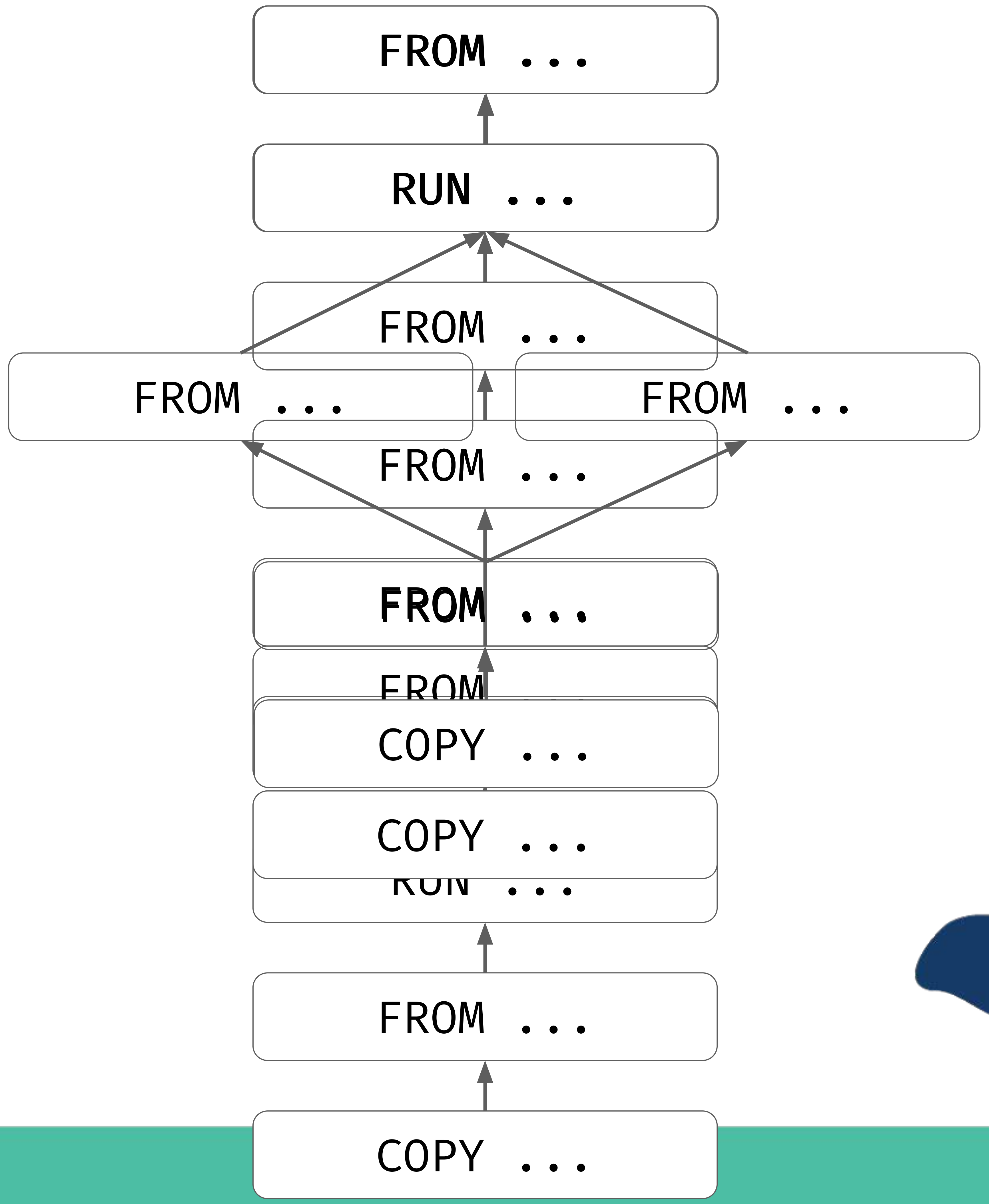

4. docker build . (docker)

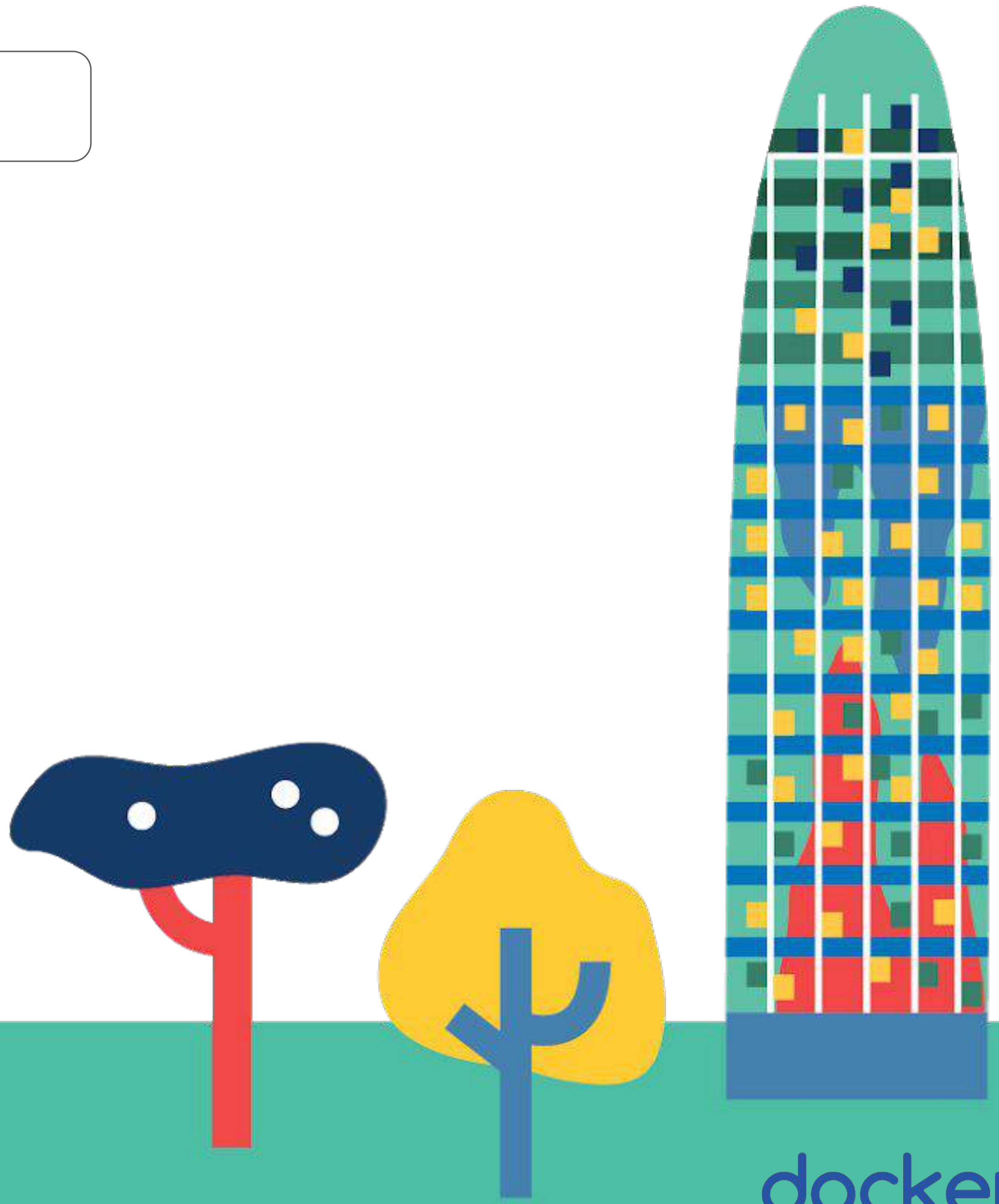
```
 # docker build .
[+] Building 2.0s (7/59)
 => [internal] load build definition from Dockerfile                     0.0s
 => => transferring dockerfile: 37B                                      0.0s
 => [internal] load .dockerignore                                        0.0s
 => => transferring context: 34B                                         0.0s
 => [internal] load metadata for docker.io/library/golang:1.11.1         1.1s
 => [internal] load build context                                        0.8s
 => => transferring context: 13.78MB                                     0.8s
 => CACHED [internal] helper image for file operations                   0.0s
 => [base 1/2] FROM docker.io/library/golang:1.11.1@sha256:63ec0e29aeb  0.0s
 => CACHED [base 2/2] RUN sed -ri s/(httpredir|deb).debian.org/deb.deb  0.0s
 => CACHED [docker-py 1/1] RUN git clone https://github.com/docker/doc  0.0s
 => [criu 1/1] RUN apt-get update && apt-get install -y  libnet-dev  l  0.8s
 => [tini 1/4] RUN apt-get update && apt-get install -y cmake vim-comm  0.8s
 => [frozen-images 1/3] RUN apt-get update && apt-get install -y jq ca  0.8s
 => [containerd 1/4] RUN apt-get update && apt-get install -y btrfs-to  0.8s
 => [swagger 1/1] RUN set -x  && export GOPATH=$(mktemp -d)  && git cl  0.8s
 => [runtime-dev 1/1] RUN apt-get update && apt-get install -y  libapp  0.8s
 => [registry 1/1] RUN set -x  && export GOPATH=$(mktemp -d)  && git c  0.8s
```

Legacy

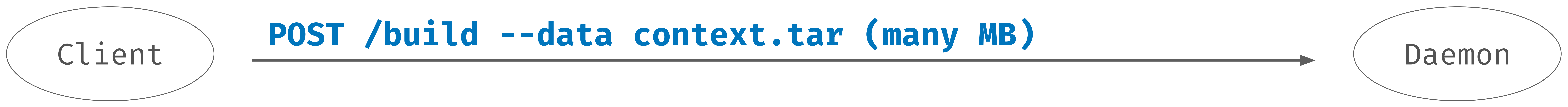
BuildKit



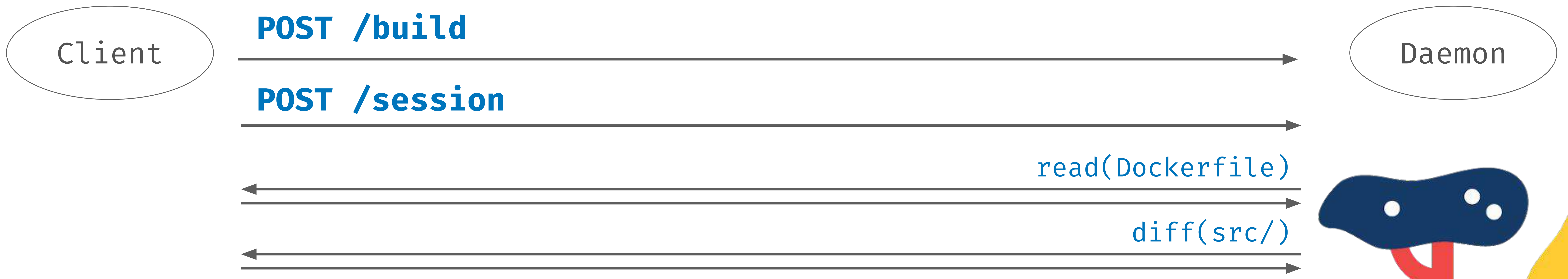


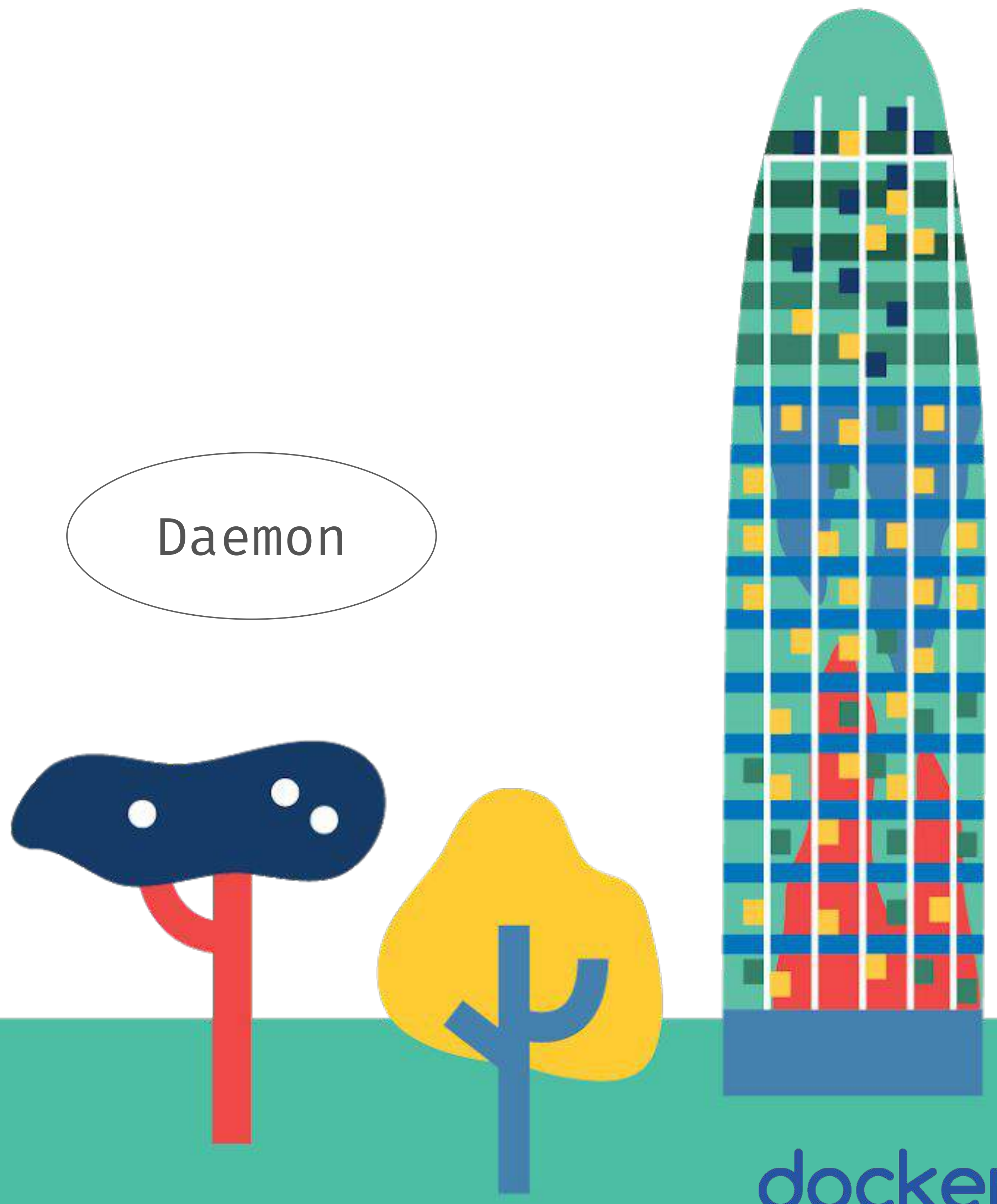


Legacy



BuildKit





New Storage

```
» docker system df
TYPE                TOTAL               ACTIVE              SIZE                RECLAIMABLE
Images              124                 2                   17.19GB             17.11GB (99%)
Containers          2                   2                   204.8kB             0B (0%)
Local Volumes       60                  2                   2.13GB              2.088GB (98%)
Build Cache         84                  0                   3.058GB             3.058GB
```

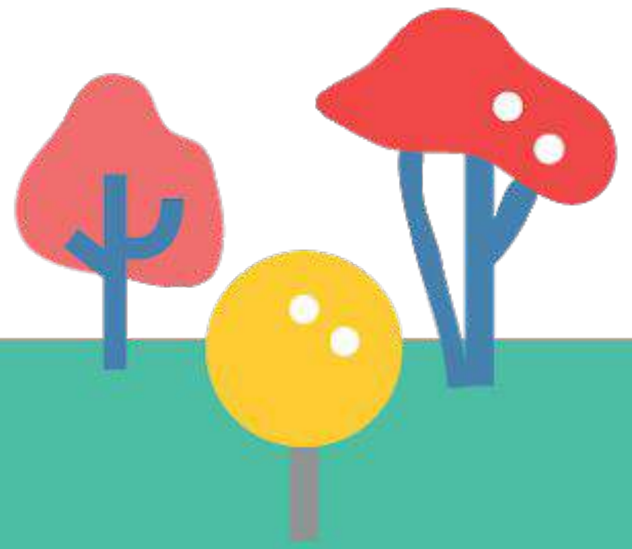
```
» docker builder prune
» docker builder prune --filter unused-for=24h --keep-storage 5GB
```

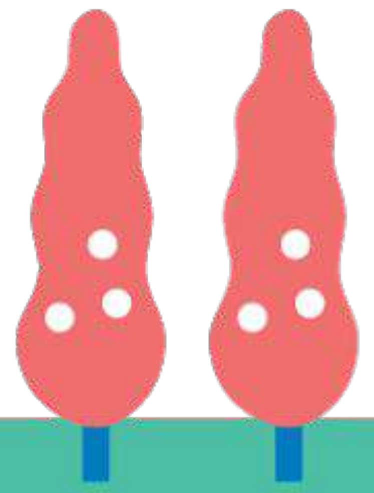
+ **Optional automatic garbage collector**

Dockerfiles

BuildKit is fully compatible with all old Dockerfiles

... but there is more





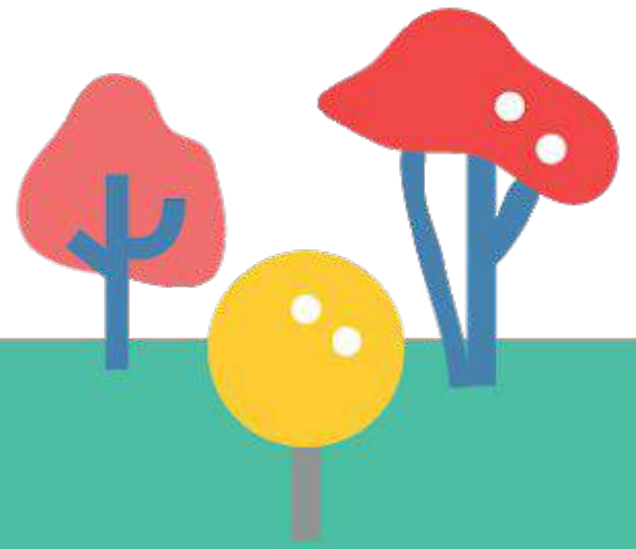
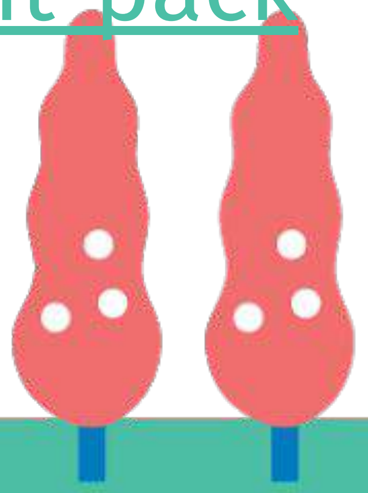
Two levels of Build definitions

LLB

- ❏ Binary DAG
- ❏ For implementers
- ❏ Efficient execution
- ❏ Efficient caching

Frontends

- ❏ Convert any user format to LLB
- ❏ Distributable as image
- ❏ Run inside container sandbox
- ❏ **Dockerfiles**
- ❏ **Buildpacks** github.com/tonistiigi/buildkit-pack
- ❏ **docker assemble**

Frontends

BuildKit can not only build Dockerfiles but pretty much anything you can imagine using a **custom frontend**

Loaded as a **container image** from registry and run in a **secure sandbox** just as your regular containers.

External Frontends

- More than only Dockerfiles
- Easily add new features to Dockerfile implementation
- Bugfixes without updating daemon
- Ensure all versions of daemon use same implementation

New Dockerfile features in v18.09

docker con 18 EUROPE

new in v18.09

```
FROM ... AS ...
ENV ...
WORKDIR ...
COPY ...
RUN --mount=[key=value] ...
USER ...

FROM ...
COPY --from= …
```

new in v18.09

Context mounts

Legacy

```
# syntax=docker/dockerfile:1
FROM alpine
COPY . files/
RUN tar -cf files.tar files
RUN rm -rf
```

BuildKit

```
# syntax=docker/dockerfile:experimental
OM alpine
N --mount=target=./files \
  tar -cf files.tar files
```

Unnecessary data copy

Single command
No extra data usage

actually
release data

new in v18.09

Secrets

```
FROM ubuntu
COPY id_rsa /root/.ssh/
RUN git clone git@github.com:myproject.git
```

new in v18.09

Secrets

Securely expose secrets to the build processes

```
# syntax=docker/dockerfile:experimental
FROM ...
RUN --mount=type=secret,id=key [cmd]
```

Pass the values from the client

```
docker build --secret id=key,src=./to/key.pem .
```

new in v18.09

Secrets: Example

Access private S3 bucket
from docker build

```
docker build --secret=...
```

type=secret
id=aws
src=path/to/secret
id=name
target=mount/path
required

new in v18.09

SSH forwarding

Clone private repositories without transferring private keys

```
# syntax=docker/dockerfile:experimental
FROM ...
RUN --mount=type=ssh git clone git@github.com:repo
```

Allow forwarding of default SSH agent or custom keys

```
docker build --ssh default .
```

new in v18.09

SSH: Example

Cloning a private SSH repository in Dockerfile

```
docker build --ssh=...
```

```
type=ssh
[id]=$SSH_AUTH_SOCK
[id]=path/to/key.pem
required
```

new in v18.09

Cache mounts

Persistent writable mountpoint across repeated builds

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

Significantly speed up actions like:

go build, **go.mod**, **ccache**, **npm install**, **maven**, **gradle**, **apt-get**, **bazel** etc.

Cache Mounts Example

WithOUT cache

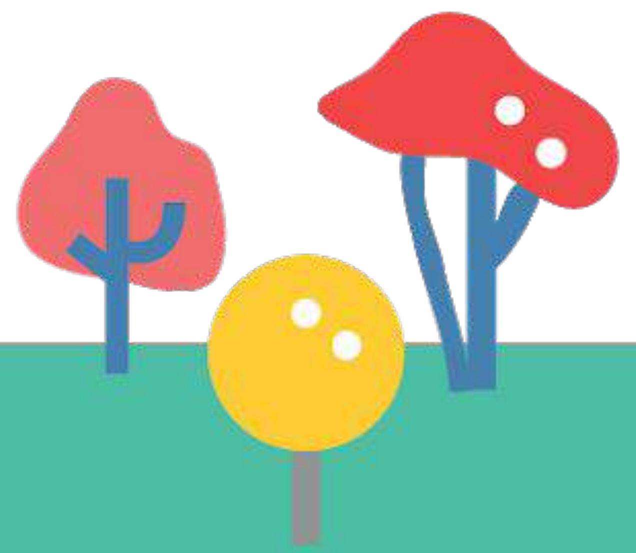
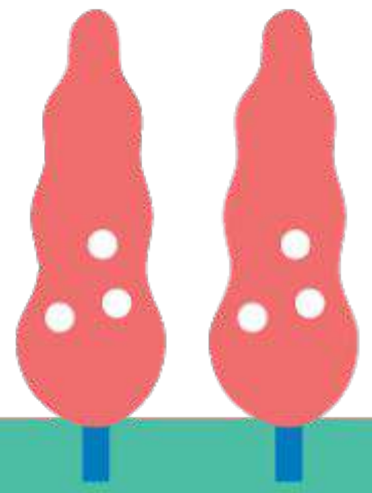
WITH cache

Syntax directive

```
# syntax = repo/image:tag
```

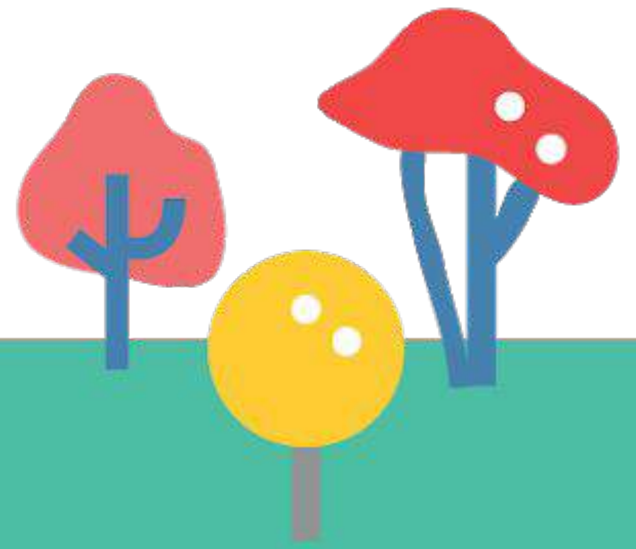
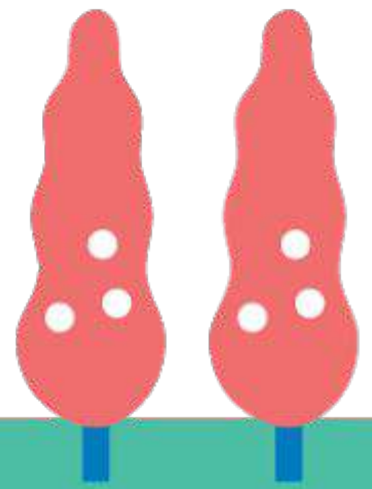
Allows switching to any custom frontend directly from docker build

- Easily add new features to Dockerfile implementation
- Automatic bugfixes without updating daemon

Official Dockerfile builder images

Stable channel	docker/dockerfile:1.0.0 docker/dockerfile:1 docker/dockerfile:latest
Experimental channel	docker/dockerfile: 1.0.0-experimental docker/dockerfile:experimental

Dockerfile frontend source

Temporarily in:
https://github.com/moby/buildkit/tree/master/frontend/dockerfile

Experimental features built with go build tags:

```
go build -tags secrets ./frontend/dockerfile
```

Building a custom Frontend

Buildkit Control API: Build() & Solve()

- Both accept a Frontend (and its arguments) as input
  - Build() takes the frontend as a function
  - Solve() takes a container image reference to run
- Both accept an Exporter (and its arguments)
  - The result of the Frontend is passed to the exporter

Internal Flow

Client
doFrontend()
Solve
Solve, ReadFile, ...
gRPC
Controller
Gateway API
Gateway API
Solve, ReadFile, ...
doFrontend()
Runs in a Container
LLB
Solver
runc
containerd

Custom Frontend

- Is passed the arguments from the client
- Plus a handle to the **Gateway API**

- Produces a **Result** which contains one or more **References** to **Snapshots** and some associated **Metadata**.

LLB Client API

The **LLB Client API** chains **Source** and **Run** operations together to describe a desired output **State**.

LLB Sources

- `st := llb.Image(“alpine:latest”)`
- `st := llb.Git(“https://github.com/.../”)`
- `st := llb.HTTP(“https://static.com/...”)`
- `st := llb.Scratch()`
- `st := llb.Local(“my-name”)`

LLB Run

- `run := st.Run(...)`

```
llb.Args([]string{...})
```

```
llb.AddMount("/src", srcst)
```

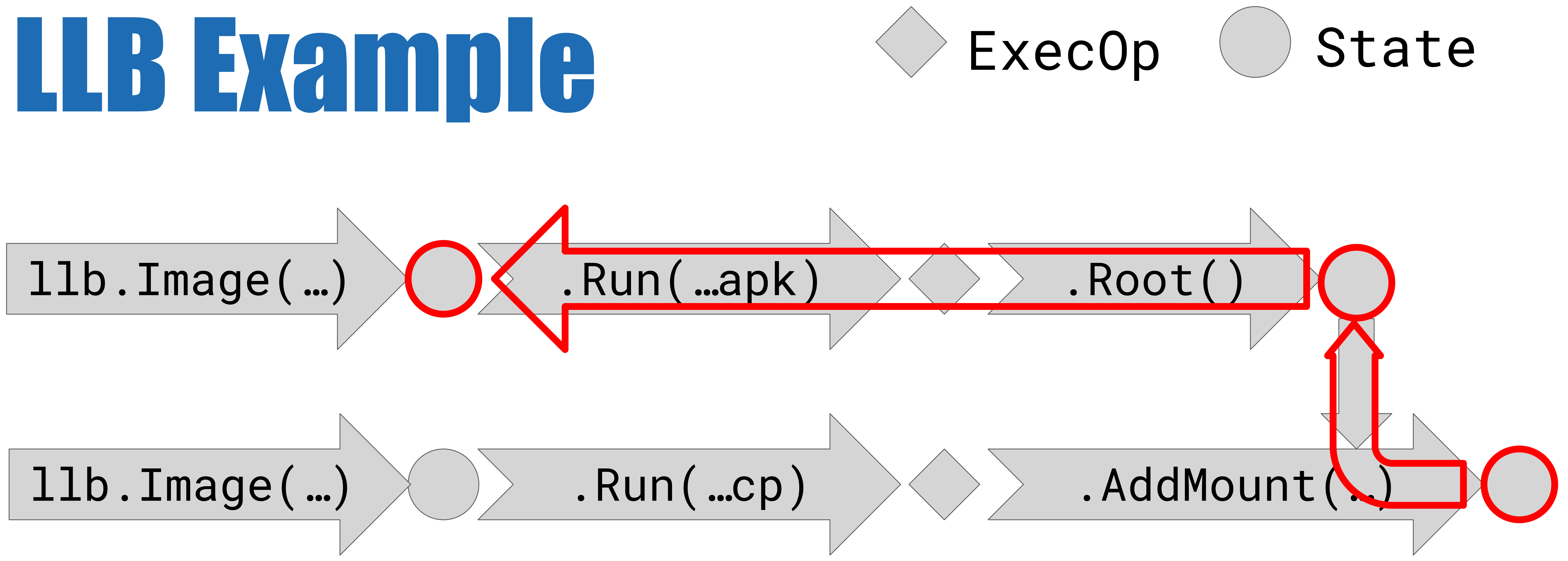
```
Dir, User, AddSecret,
Network, AddEnv, etc.
```

Output State from a st.Run()

```
○ st := run.Root()
```

```
○ st := run.AddMount(“/output”, inState)
```

LLB Example


◆ ExecOp
● State
llb.Image(…)
.Run(…apk)
.Root()
llb.Image(…)
.Run(…cp)
.AddMount(…)

Gateway API: Solve() Method

This is the main mechanism by which a **Frontend** can obtain a **References** to return.

Input is an **LLB Graph** describing a desired **State**.

Returns a **Result** containing one or more **References** to **Snapshots** and accompanying **Metadata.**

Gateway API: References

A **Frontend**’s main use for a **Reference** (the result of a Solve) is to construct its own **Result** to return.

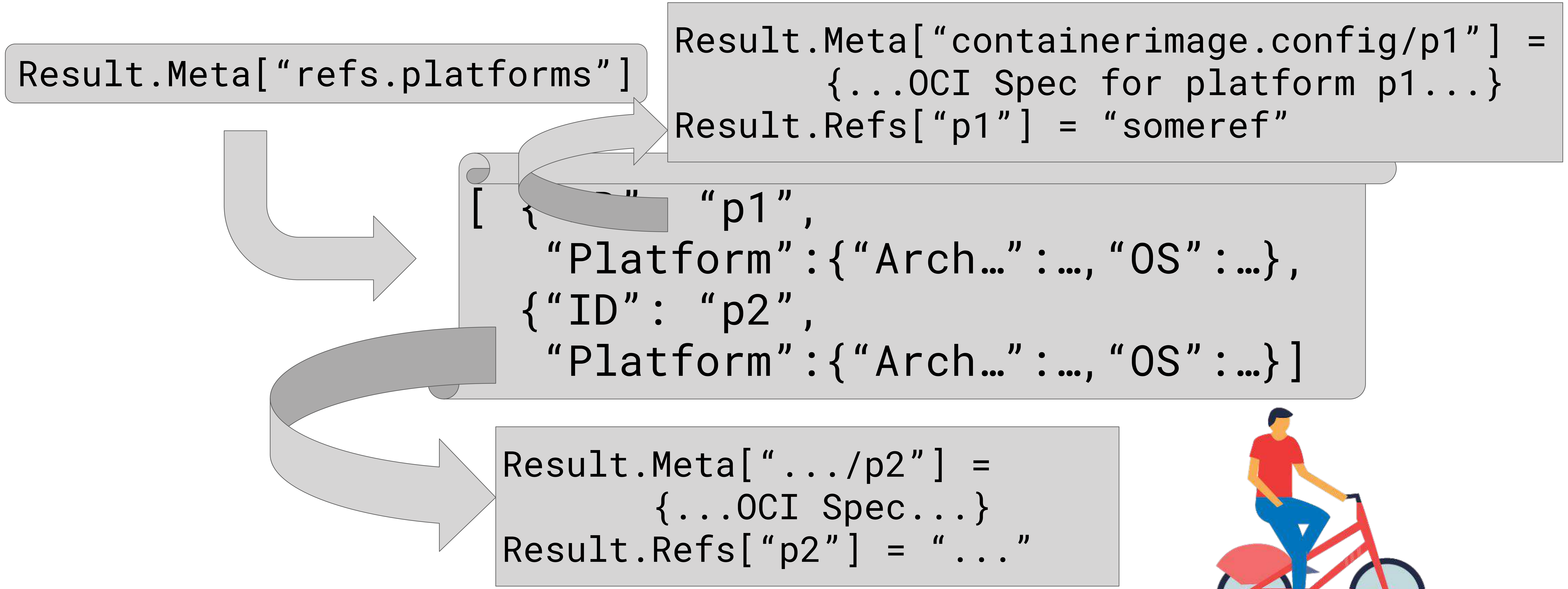
But it can also **ReadFile**, **StatFile** or **ListDir** within the referenced **Snapshot**.

Top-Level Result

```
type Result struct {
        Ref      Reference
        Refs     map[string]Reference
        Metadata map[string][]byte
}
```

“container.image”: «OCI spec (JSON)»

Multiplatform Top-Level Result



```
Result.Meta["refs.platforms"]
```

```
Result.Meta["containerimage.config/p1"] =
        {...OCI Spec for platform p1...}
Result.Refs["p1"] = "someref"
```

```
[ {   D":  "p1",
   "Platform":{"Arch…":…, "OS":…},
  {"ID": "p2",
   "Platform":{"Arch…":…, "OS":…}]
```

```
Result.Meta[".../p2"] =
        {...OCI Spec...}
Result.Refs["p2"] = "..."
```

Demo Outline

- Build a custom frontend.
    - Frontend must be usable with **Docker 18.09**
    - Container should print some **user supplied** text
    - Be **On Brand**.

Docker Assemble

- New tool announced during DockerCon EU
- **Fast** builds with **zero configuration**
- Built using the same **tools** and **techniques** discussed here today

... Just like the **Dockerfile** frontends

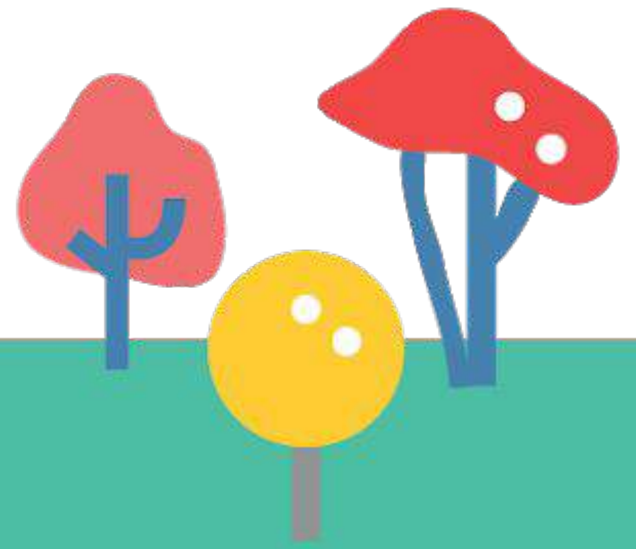
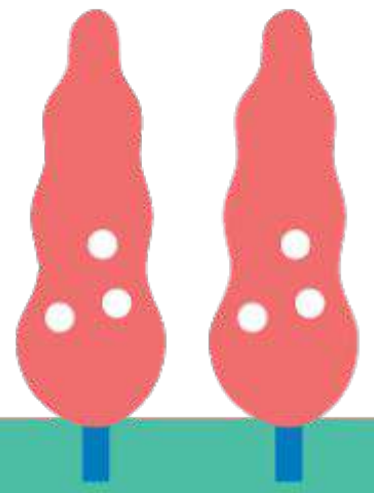
Contributing features to Dockerfile frontend

Upstream in:

https://**github.com/moby/buildkit**/tree/master/frontend/dockerfile

Experimental features built with go build tags:

```
docker build --build-arg BUILDTAGS=”dfrunmount dfssh” \
    ./frontend/dockerfile/cmd/dockerfile-frontend
```

Popular proposals now possible externally

Heredoc #34423

INCLUDE #12749

Multiple contexts #37129

ENVFILE #28617

COPY --exclude #37333

IMPORT/EXPORT #32100

Multiple .dockerignore #12886

The End

Questions? Or find us in the hallway track…

#buildkit on Docker Community Slack

https://github.com/docker/dceu18-build-demo

Docker is looking for Engineers in Cambridge, UK and Paris

https://www.docker.com/careers