Atlassian

Be a better developer with Docker

Tricks of the trade

NICOLA PAOLUCCI • DEVELOPER INSTIGATOR • Atlassian • @DURDN

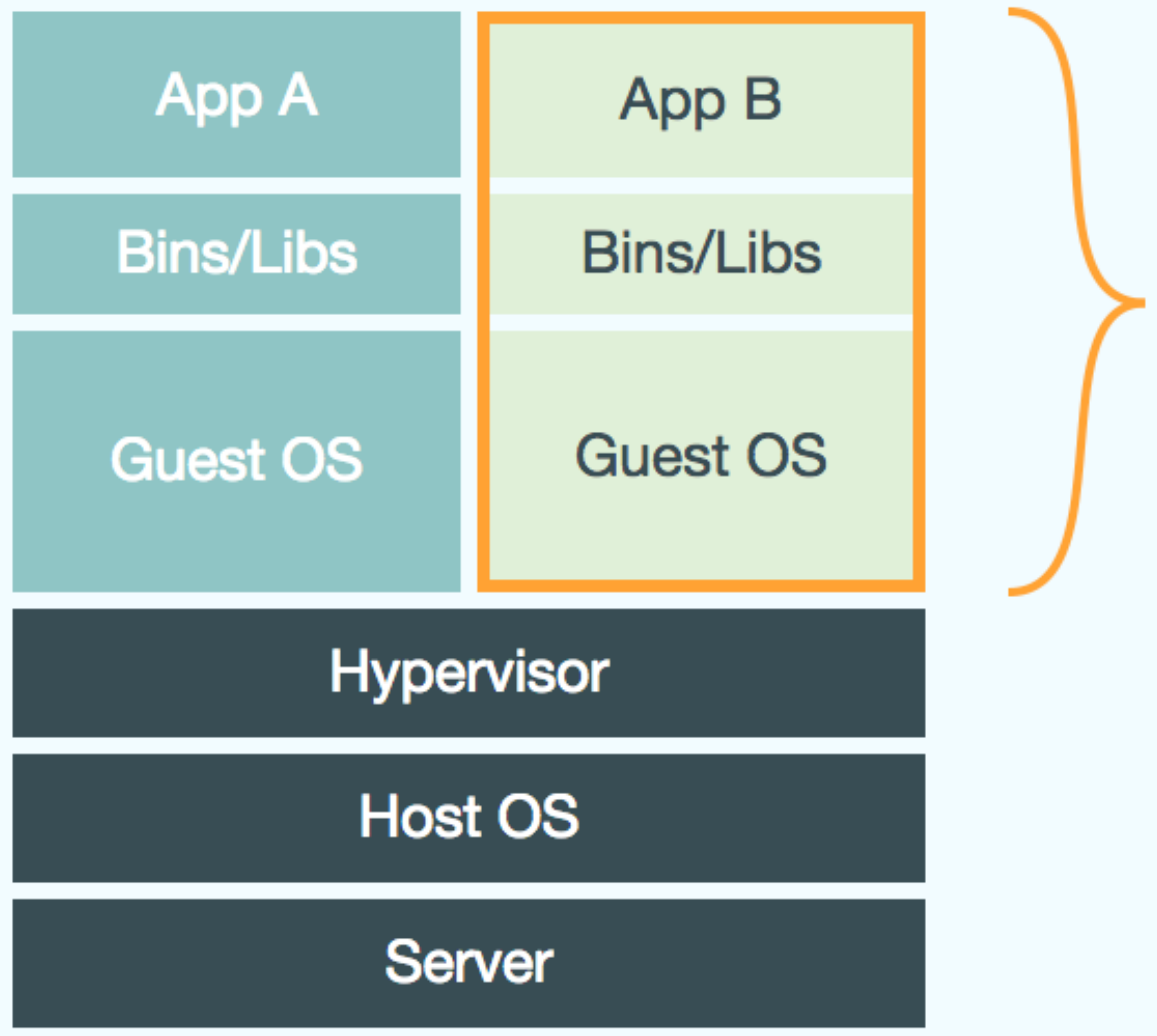
6A
1671
ZOO
The wildest place in town
Nørreport st.
Gl. Torv

6

3 minute Docker intro?

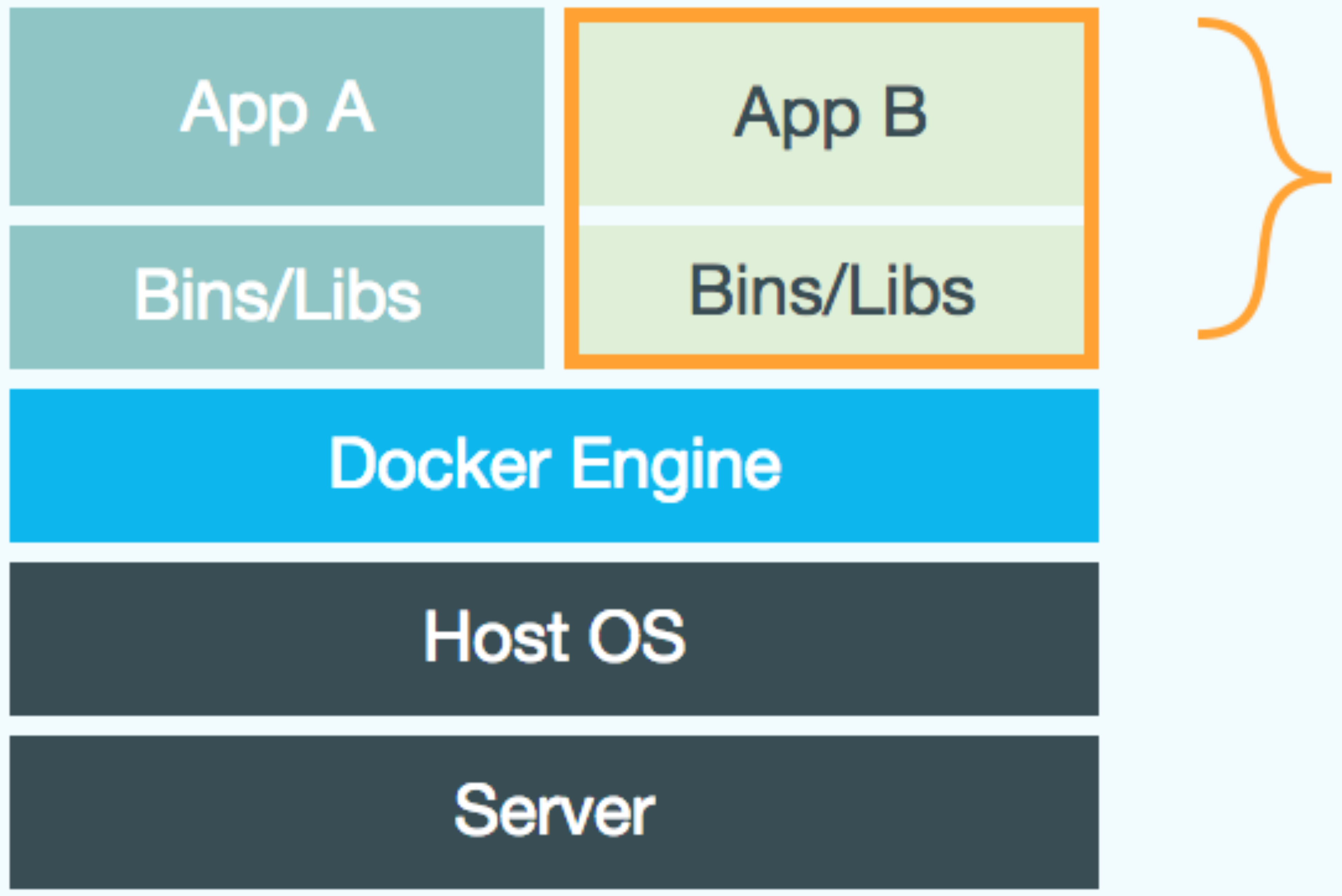
Deploying code to the cloud is hard

docker



Virtual Machines

Each virtualized application includes not only the application - which may be only 10s of MB - and the necessary binaries and libraries, but also an entire guest operating system - which may weigh 10s of GB.



Docker

The Docker Engine container comprises just the application and its dependencies. It runs as an isolated process in userspace on the host operating system, sharing the kernel with other containers. Thus, it enjoys the resource isolation and allocation benefits of VMs but is much more portable and efficient.

The plan for this session:

① Why does Docker make Developers happy?

② Notes on Workflows and Techniques

③ Tools, Tips and Hacks

Why?

Compulsion to have clean and perfect environments

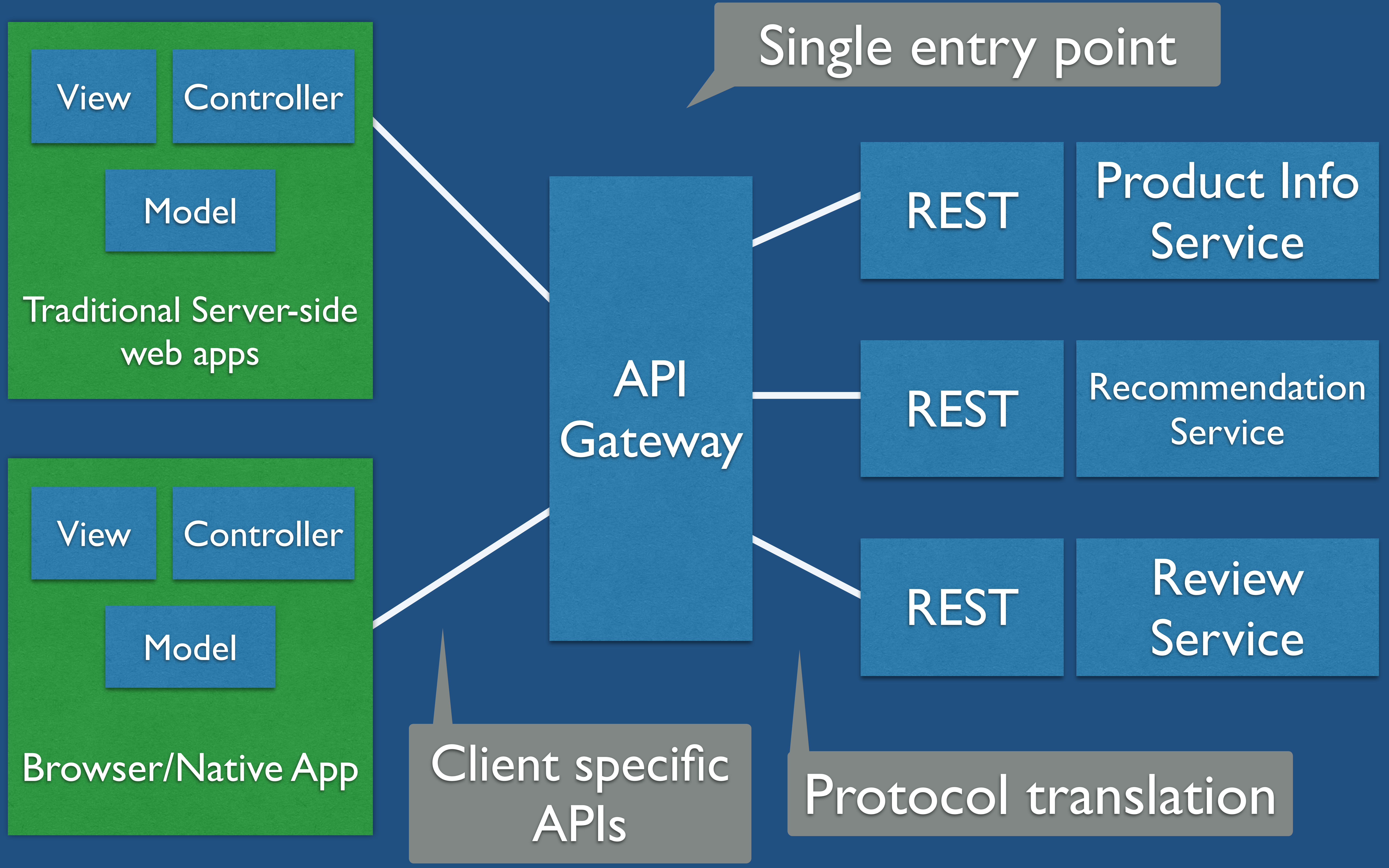
The need for speed of every developer with an idea

Fast application mobility, Real repeatability

Cooperate smoothly with Ops or run our own DevOps

Building blocks for your Micro-services Architecture

Micro-services Architecture Blueprint



@crichardson

Development Workflows
or
“What can I do with it?”

Development workflows on Docker:

① Package your releases, push, run in PaaS

② Create your “base” Container

③ Shared Volume Dev/Debug Container

Development workflows on Docker:

④ Test different versions of your tools

⑤ The Installation Container

⑥ Default-Official-Services-In-A-Box

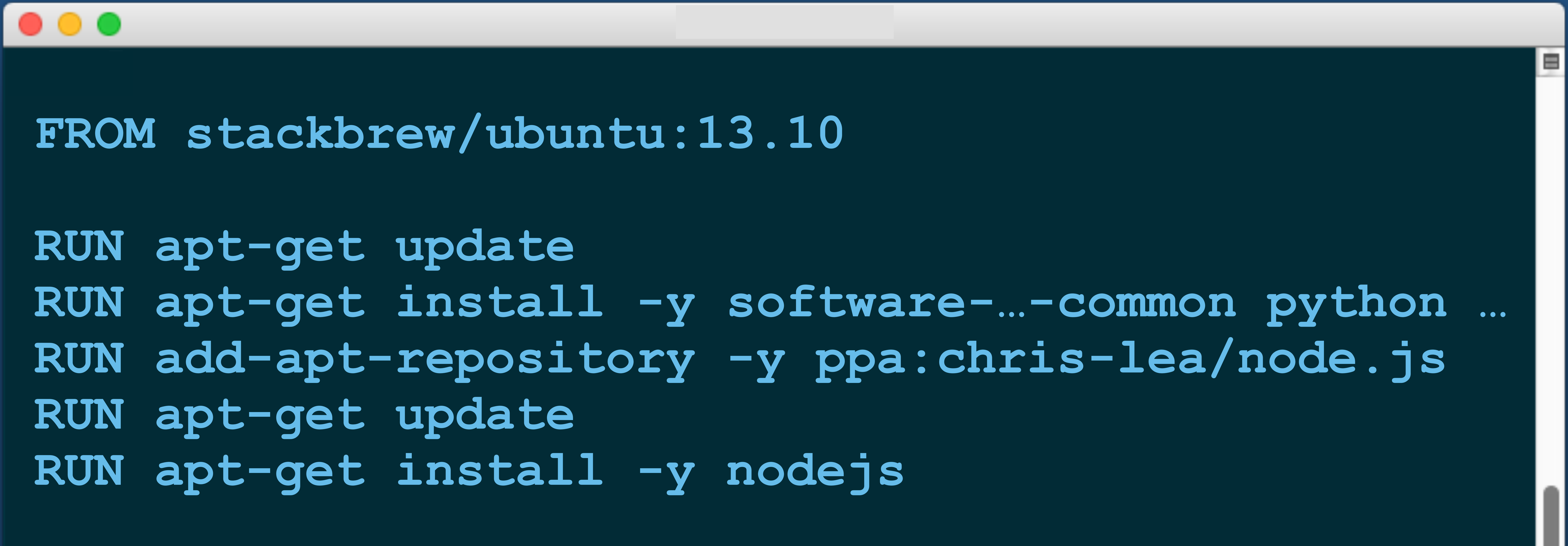
Basic Techniques

Pre-requisite: Dockerfiles

A list of instructions to automate building your image. The steps are cached along the way for fast re-use.

Dockerfiles: Repeatable Magic

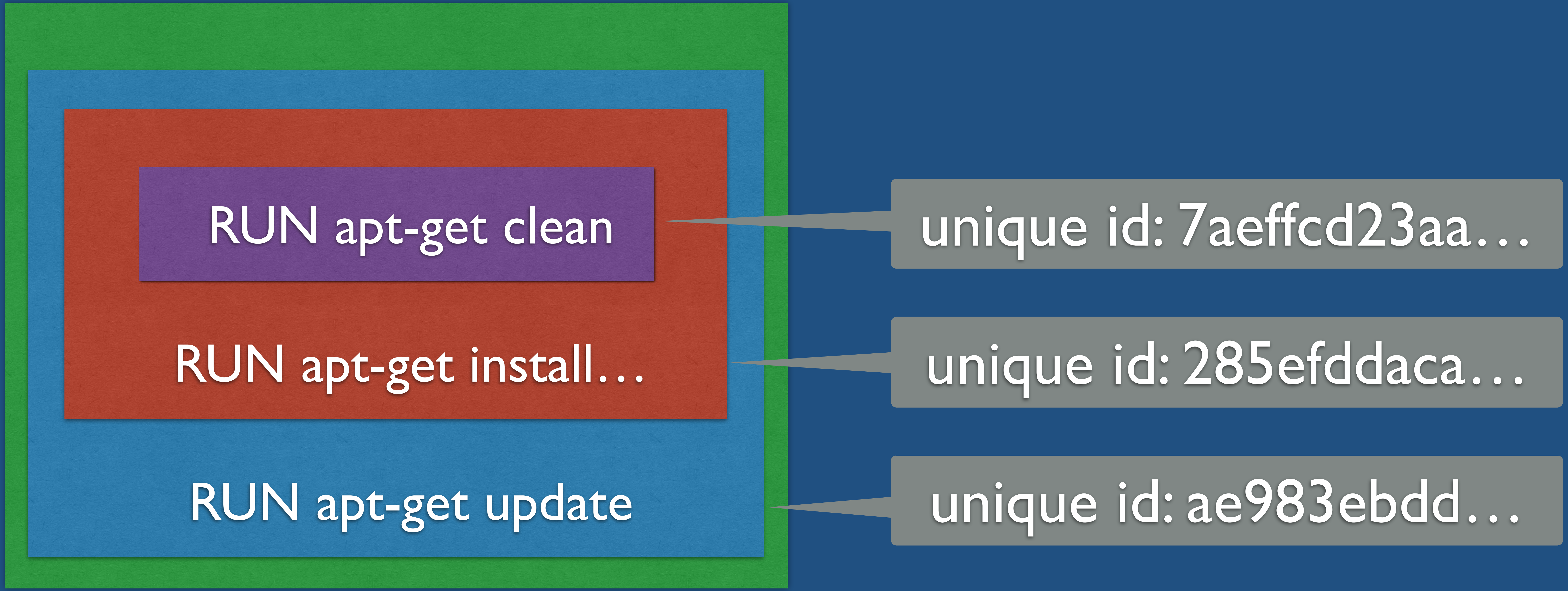
Reads like a shell script but each step is cached



```
FROM stackbrew/ubuntu:13.10

RUN apt-get update
RUN apt-get install -y software-…-common python …
RUN add-apt-repository -y ppa:chris-lea/node.js
RUN apt-get update
RUN apt-get install -y nodejs
```

Every RUN command creates a layer

copy-on-write filesystem



“add + build” routine magic

```
docker add <src> <dest>
```

`docker add <src> <dest>`

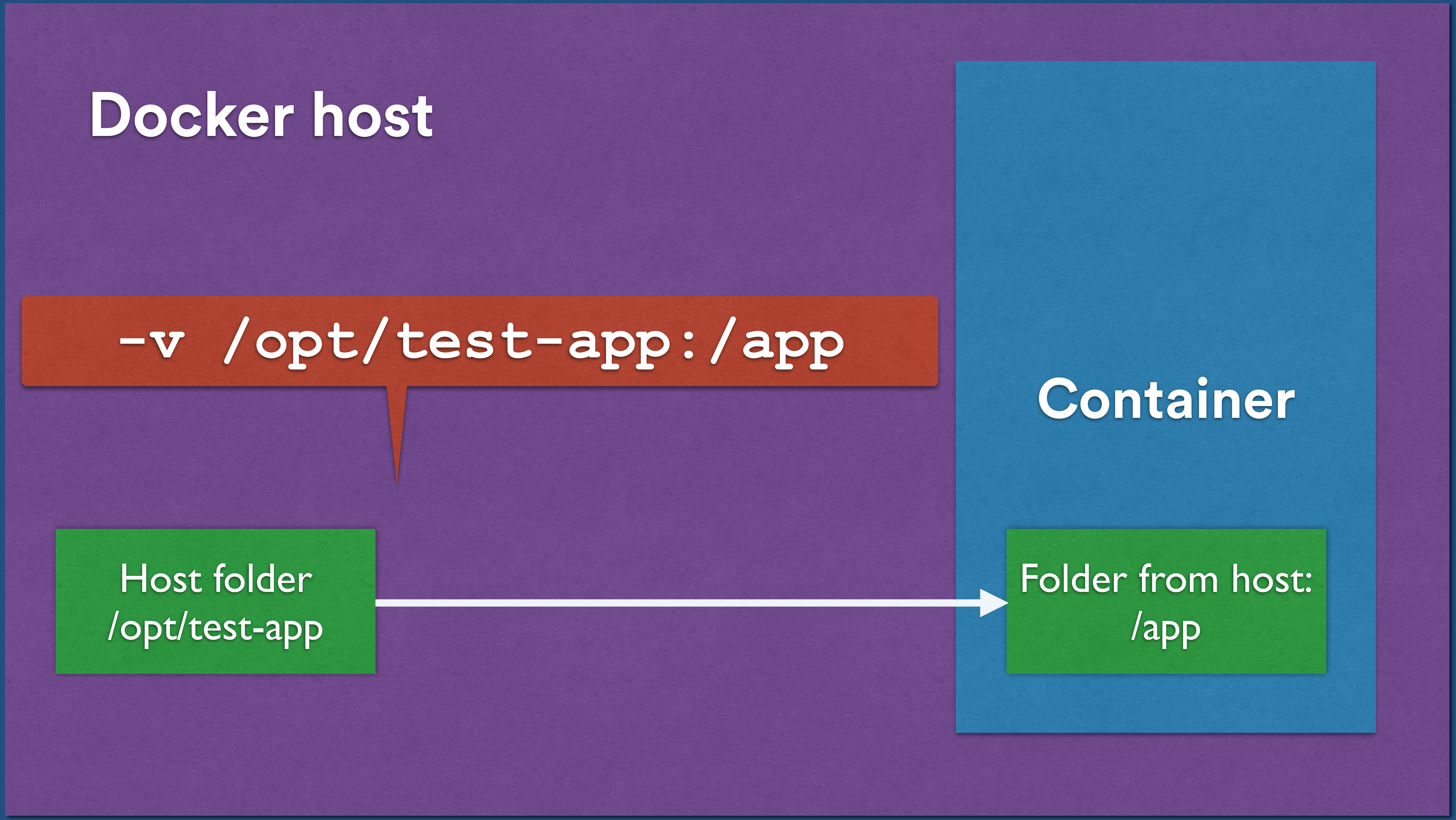
The **ADD** instruction copies new files from host’s <src> to container’s <dest>

“add + build” routine magic?!

1. Update code in local app folder (git pull?)
2. docker build your image with updated code
3. Distribute and profit!

Sharing data in containers

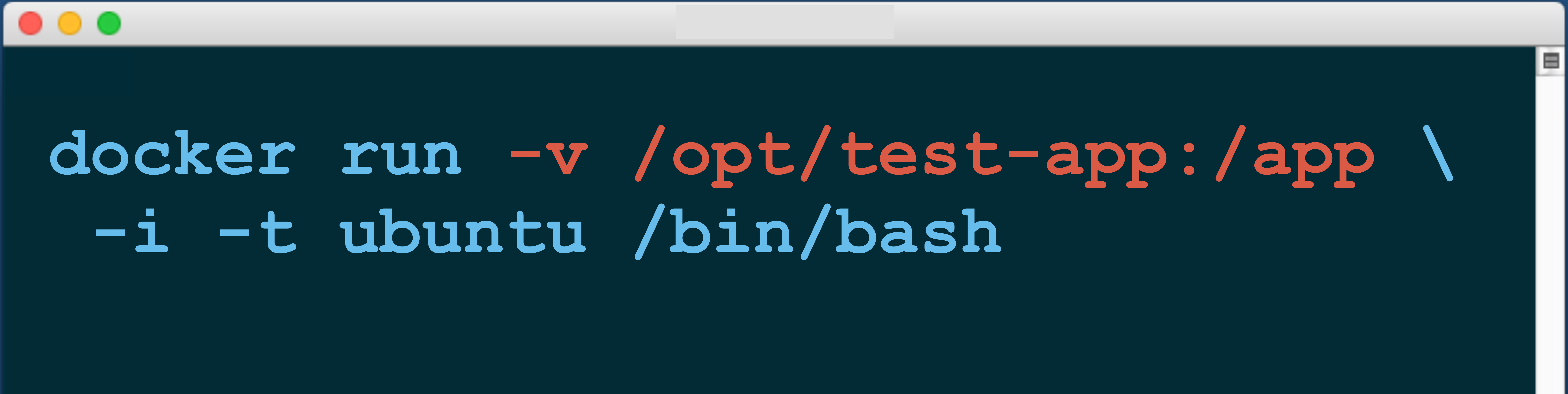
share folder from host to containers



```
-v /opt/test-app:/app
```

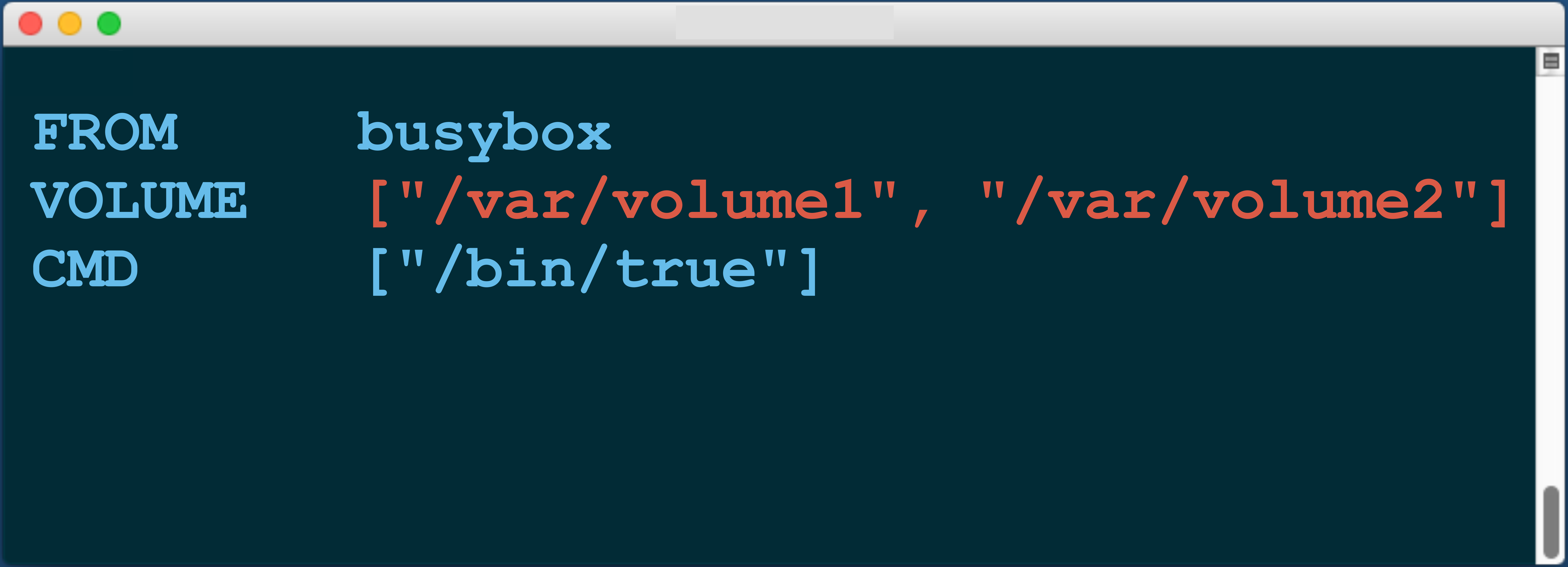
From host to containers is simple

Use the run -v (volume option) to specify host/ container folder to be synced



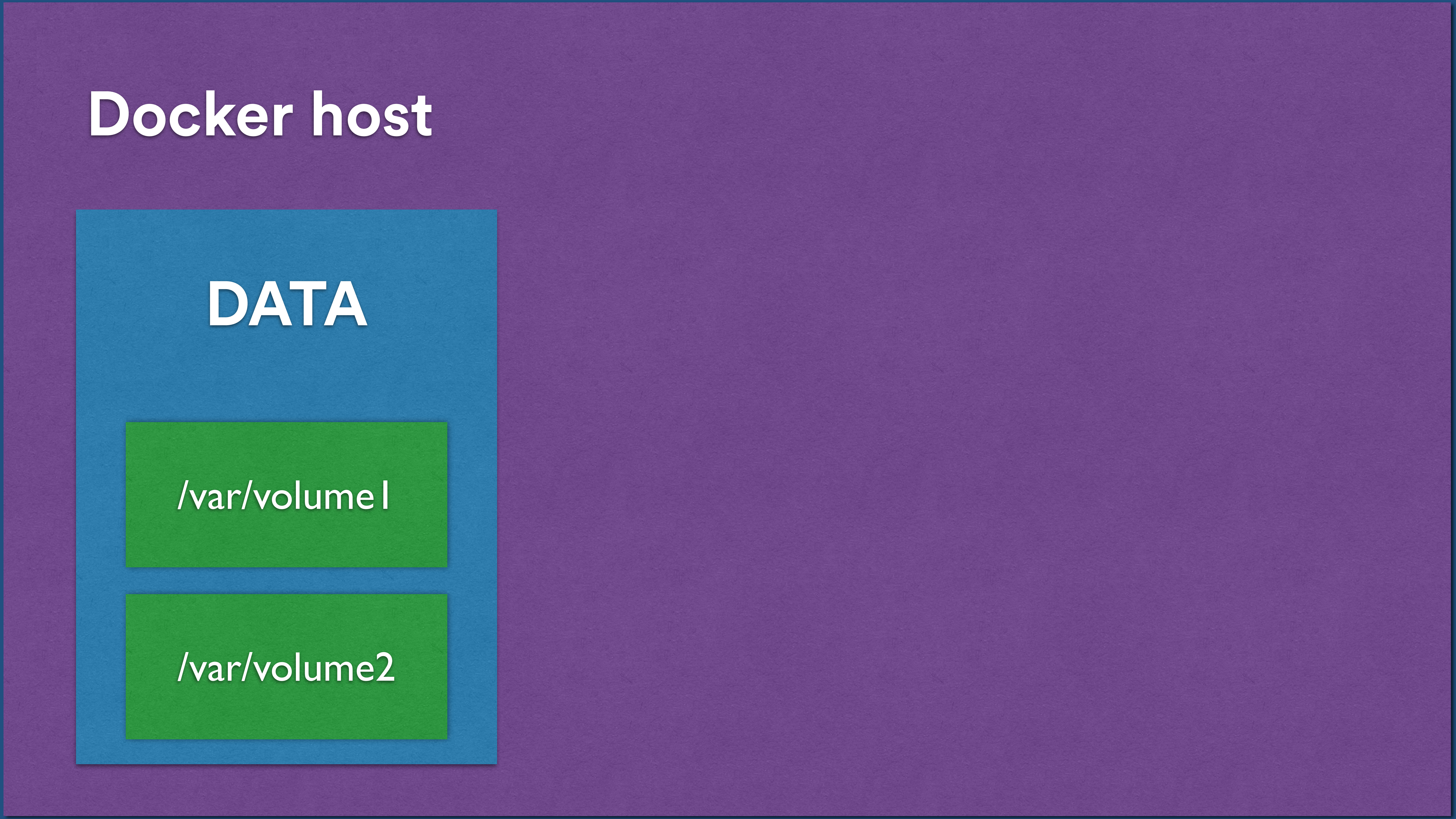
```
docker run -v /opt/test-app:/app \
 -i -t ubuntu /bin/bash
```

Same pattern using Dockerfile



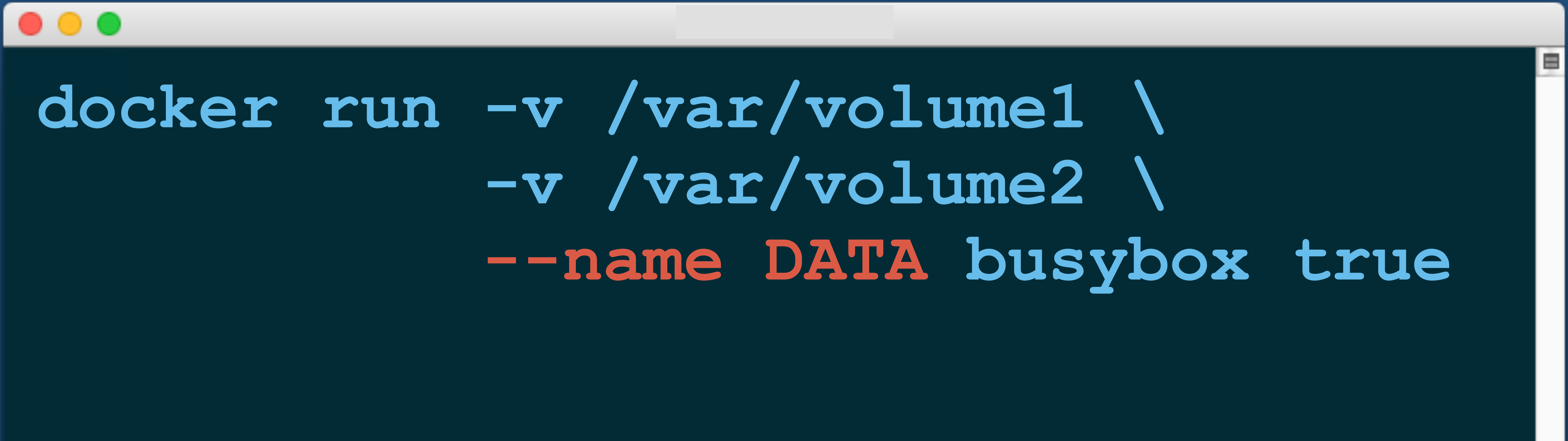
```
FROM      busybox
VOLUME    ["/var/volume1", "/var/volume2"]
CMD       ["/bin/true"]
```

Common pattern: Data in containers



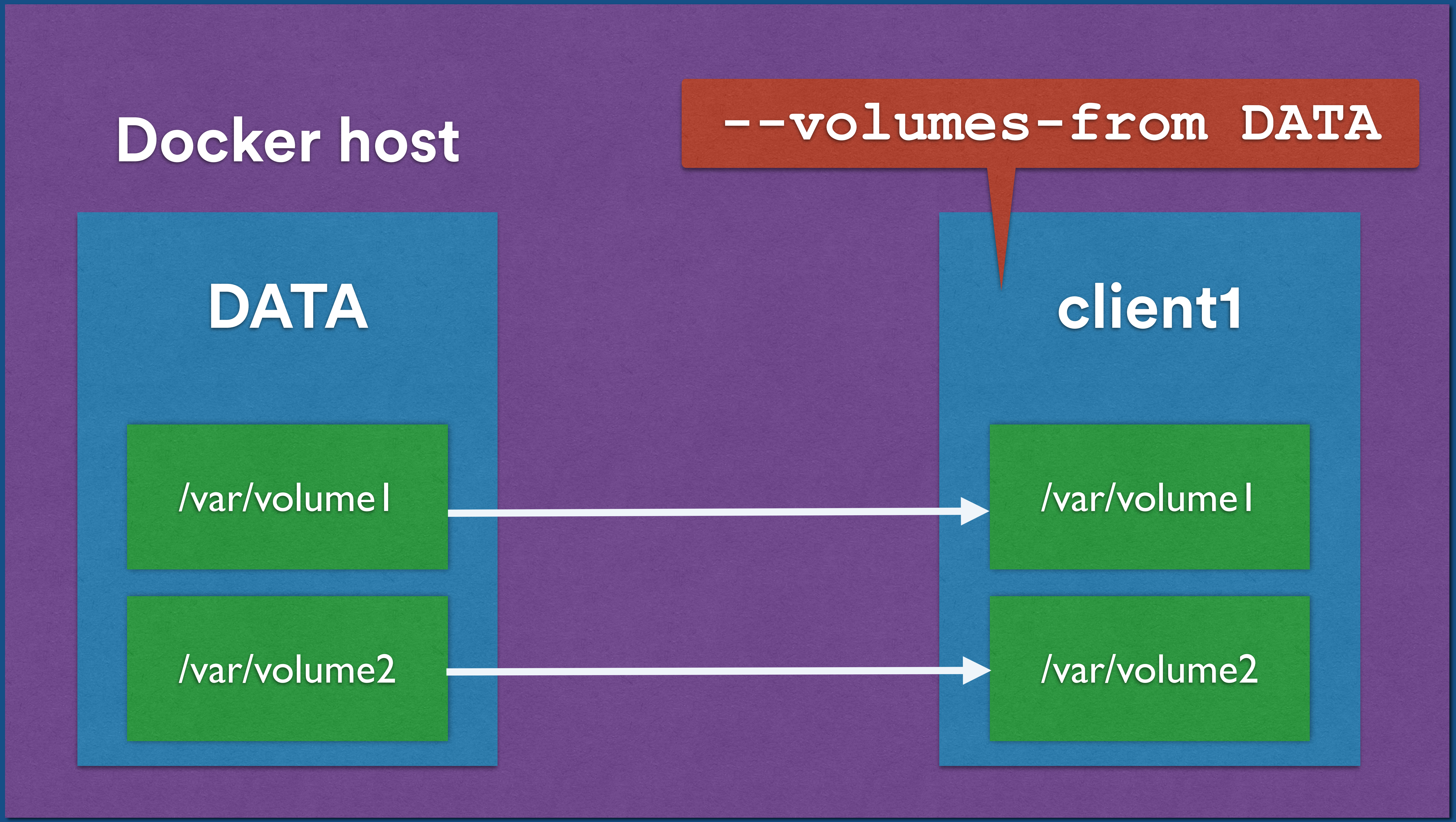
Common pattern: data in containers

Switched off, named, data container which exposes a folder



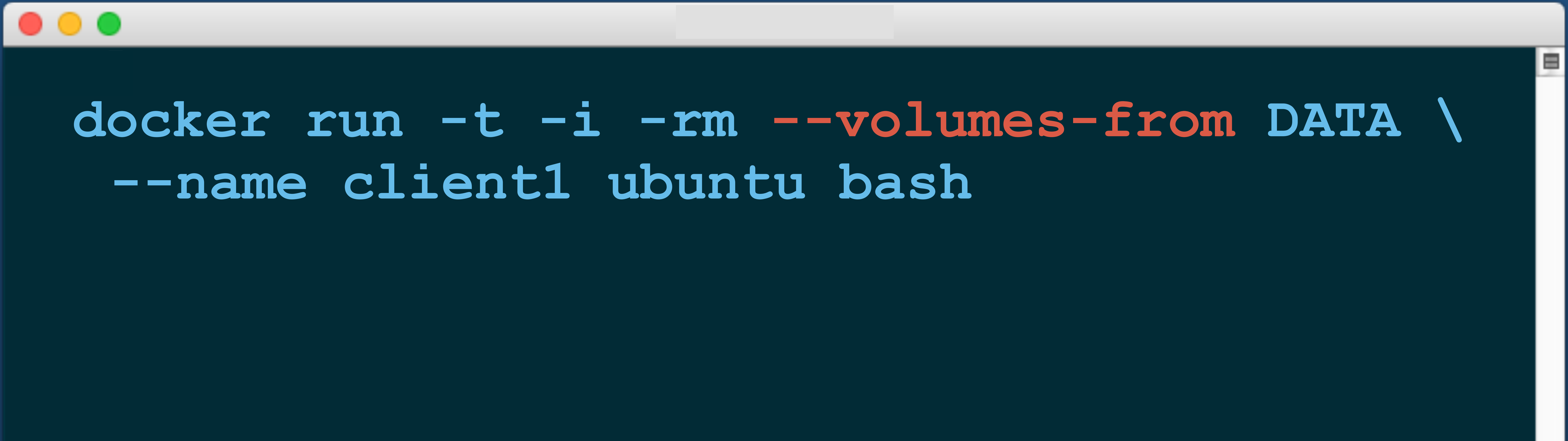
```
docker run -v /var/volume1 \
           -v /var/volume2 \
           --name DATA busybox true
```

Volumes

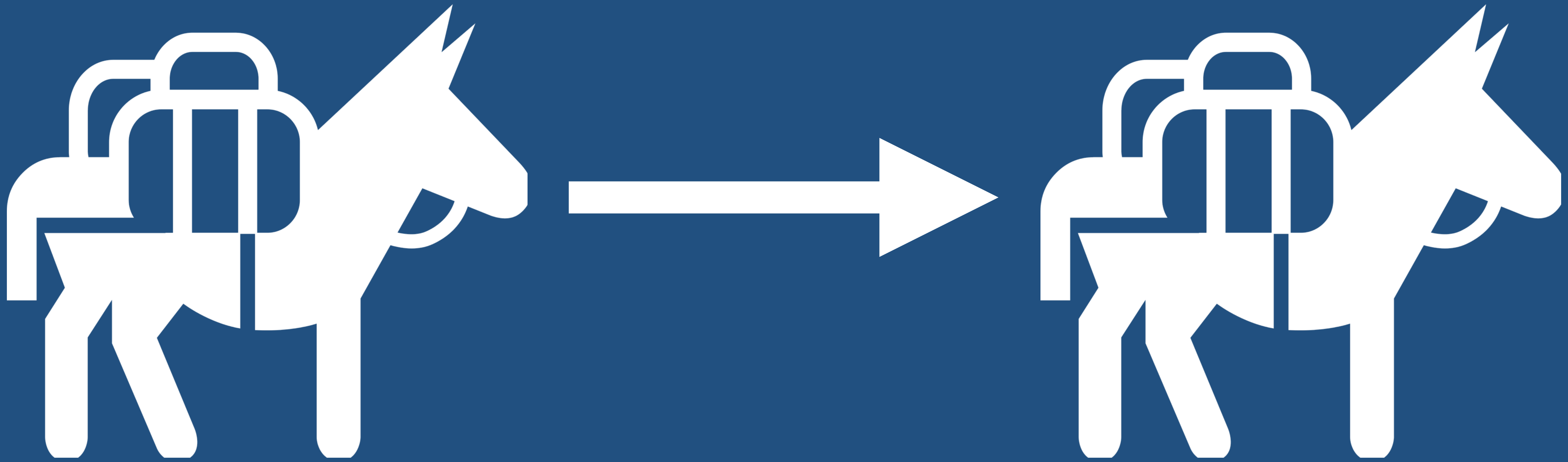

Docker host
--volumes-from DATA
DATA
client1
/var/volume1
/var/volume2
/var/volume1
/var/volume2

Common pattern: data in containers

Then mount the data container in your application containers

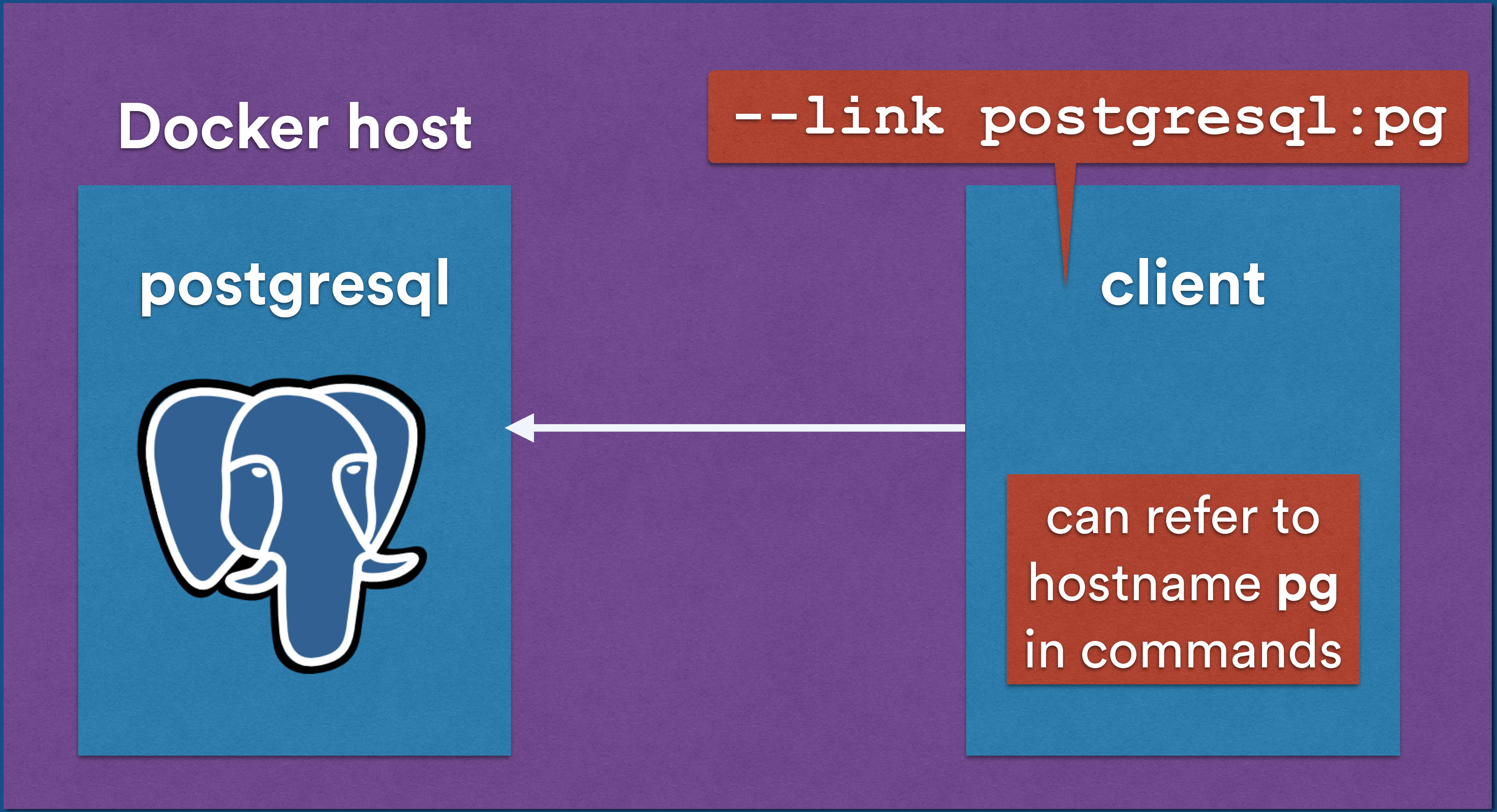


```
docker run -t -i -rm --volumes-from DATA \
  --name client1 ubuntu bash
```



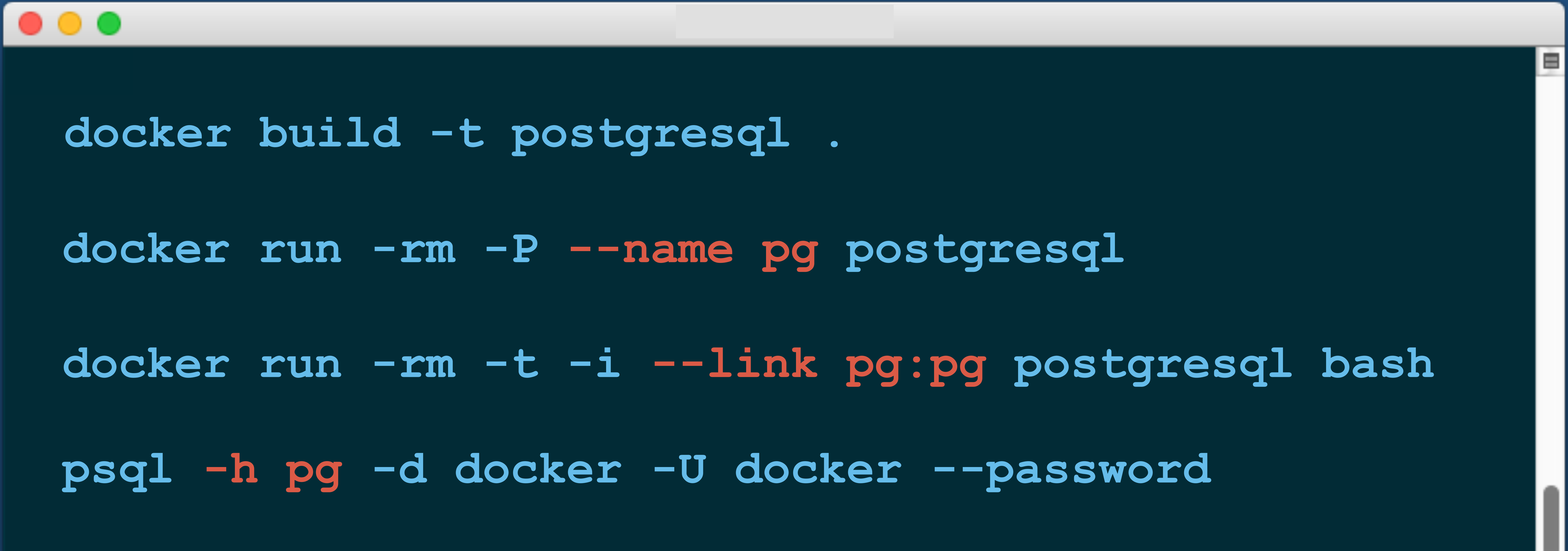
`--links`: simple service connections for docker

Linked containers



Sample access to linked container

Build the image, run it with a name,
link in child container



```
docker build -t postgresql .

docker run -rm -P --name pg postgresql

docker run -rm -t -i --link pg:pg postgresql bash

psql -h pg -d docker -U docker --password
```

Tips & Hacks

Embrace Reusability in Dockerfiles

Write general requirements early, commit and name relevant checkpoints, leave customisations last

Base images off of Debian

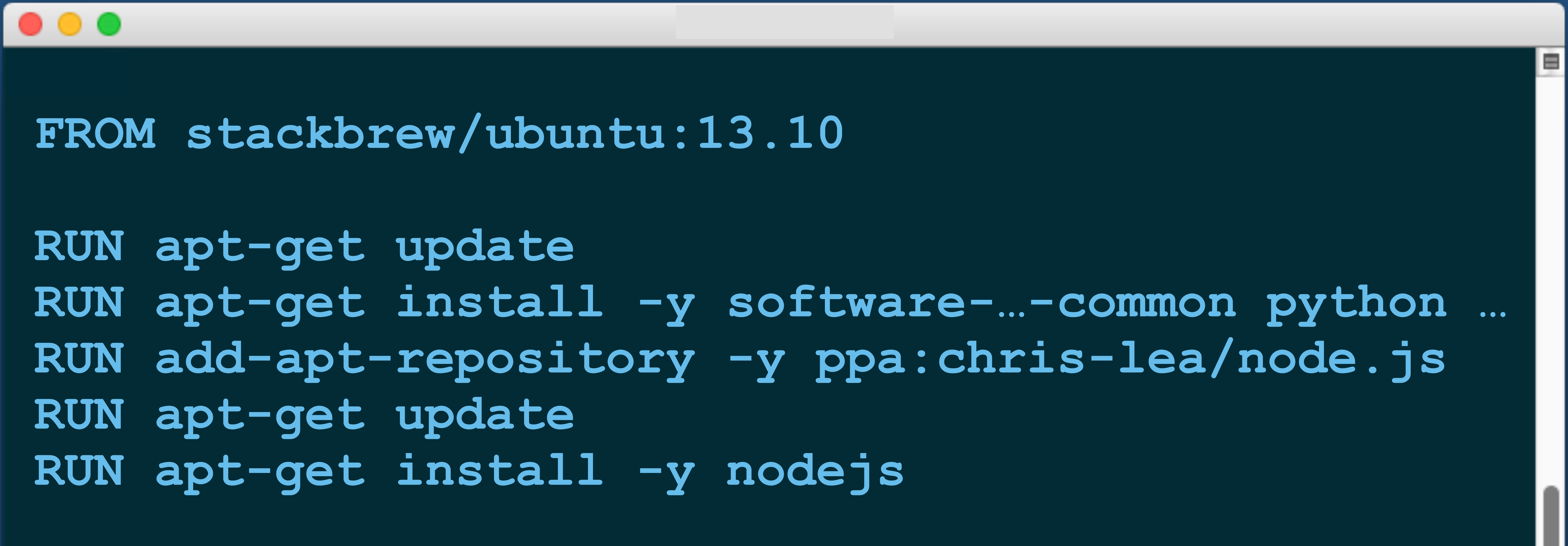
Avoid temporary files

Clean up after the package manager

Combine commands

Combine commands when logical

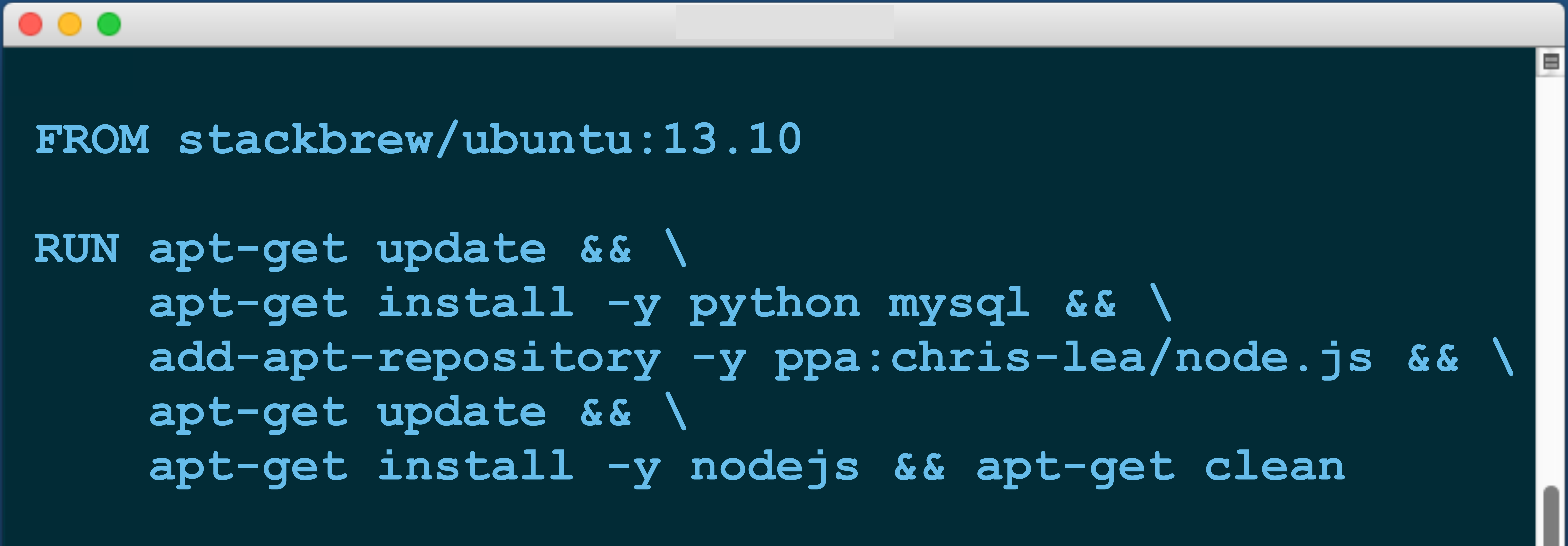
This will cache every step …



```
FROM stackbrew/ubuntu:13.10

RUN apt-get update
RUN apt-get install -y software-…-common python …
RUN add-apt-repository -y ppa:chris-lea/node.js
RUN apt-get update
RUN apt-get install -y nodejs
```

Combine commands when logical

This will use one step for the dependencies setup!

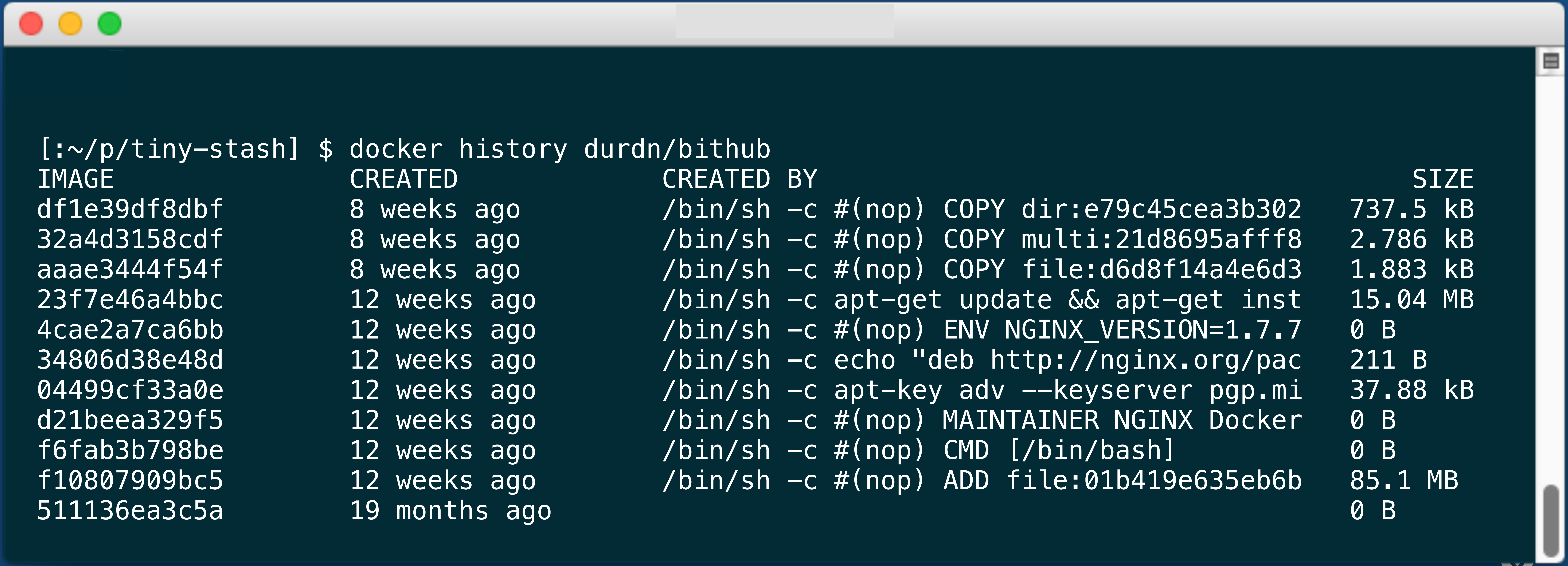


```
FROM stackbrew/ubuntu:13.10

RUN apt-get update && \
    apt-get install -y python mysql && \
    add-apt-repository -y ppa:chris-lea/node.js && \
    apt-get update && \
    apt-get install -y nodejs && apt-get clean
```

Inspect space used by your cached images

for more read http://bit.do/optimize-docker-images by Brian De Hamer

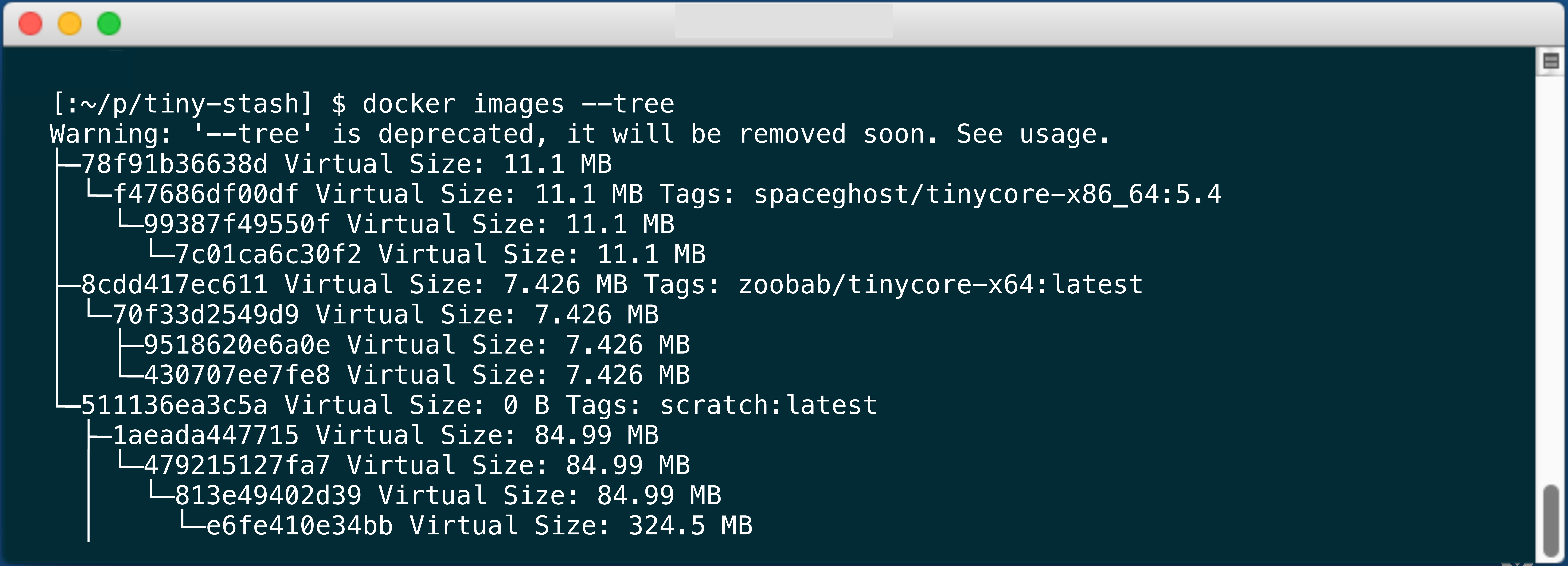
Use docker history to inspect the cost of each cache point



```
[:~/p/tiny-stash] $ docker history durdn/bithub
IMAGE                CREATED              CREATED BY                                        SIZE
df1e39df8dbf         8 weeks ago          /bin/sh -c #(nop) COPY dir:e79c45cea3b302   737.5 kB
32a4d3158cdf         8 weeks ago          /bin/sh -c #(nop) COPY multi:21d8695afff8   2.786 kB
aaae3444f54f         8 weeks ago          /bin/sh -c #(nop) COPY file:d6d8f14a4e6d3   1.883 kB
23f7e46a4bbc         12 weeks ago         /bin/sh -c apt-get update && apt-get inst   15.04 MB
4cae2a7ca6bb         12 weeks ago         /bin/sh -c #(nop) ENV NGINX_VERSION=1.7.7   0 B
34806d38e48d         12 weeks ago         /bin/sh -c echo "deb http://nginx.org/pac   211 B
04499cf33a0e         12 weeks ago         /bin/sh -c apt-key adv --keyserver pgp.mi   37.88 kB
d21beea329f5         12 weeks ago         /bin/sh -c #(nop) MAINTAINER NGINX Docker   0 B
f6fab3b798be         12 weeks ago         /bin/sh -c #(nop) CMD [/bin/bash]           0 B
f10807909bc5         12 weeks ago         /bin/sh -c #(nop) ADD file:01b419e635eb6b   85.1 MB
511136ea3c5a         19 months ago                                                    0 B
```

docker images --tree

Use docker images --tree to get a hierarchy graph of images



```
[:~/p/tiny-stash] $ docker images --tree
Warning: '--tree' is deprecated, it will be removed soon. See usage.
├─78f91b36638d Virtual Size: 11.1 MB
│ └─f47686df00df Virtual Size: 11.1 MB Tags: spaceghost/tinycore-x86_64:5.4
│   └─99387f49550f Virtual Size: 11.1 MB
│     └─7c01ca6c30f2 Virtual Size: 11.1 MB
├─8cdd417ec611 Virtual Size: 7.426 MB Tags: zoobab/tinycore-x64:latest
│ └─70f33d2549d9 Virtual Size: 7.426 MB
│   ├─9518620e6a0e Virtual Size: 7.426 MB
│   └─430707ee7fe8 Virtual Size: 7.426 MB
└─511136ea3c5a Virtual Size: 0 B Tags: scratch:latest
  ├─1aeada447715 Virtual Size: 84.99 MB
  │ └─479215127fa7 Virtual Size: 84.99 MB
  │   └─813e49402d39 Virtual Size: 84.99 MB
  │     └─e6fe410e34bb Virtual Size: 324.5 MB
```

Sometimes it’s nice to flatten an image

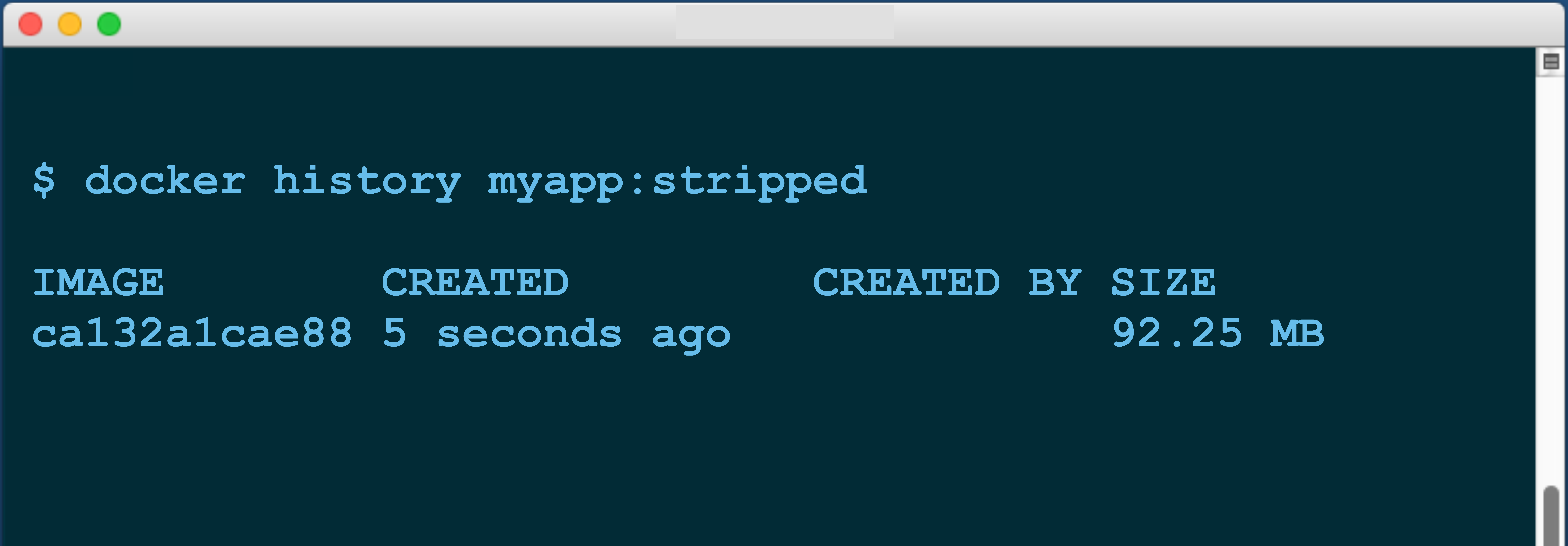
Export and re-import the image

This has the useful effect to flatten it to a single image

```
docker export aa3f12cc | docker import - myapp:stripped
```

Export and re-import the image

This has the useful effect to flatten it to a single image

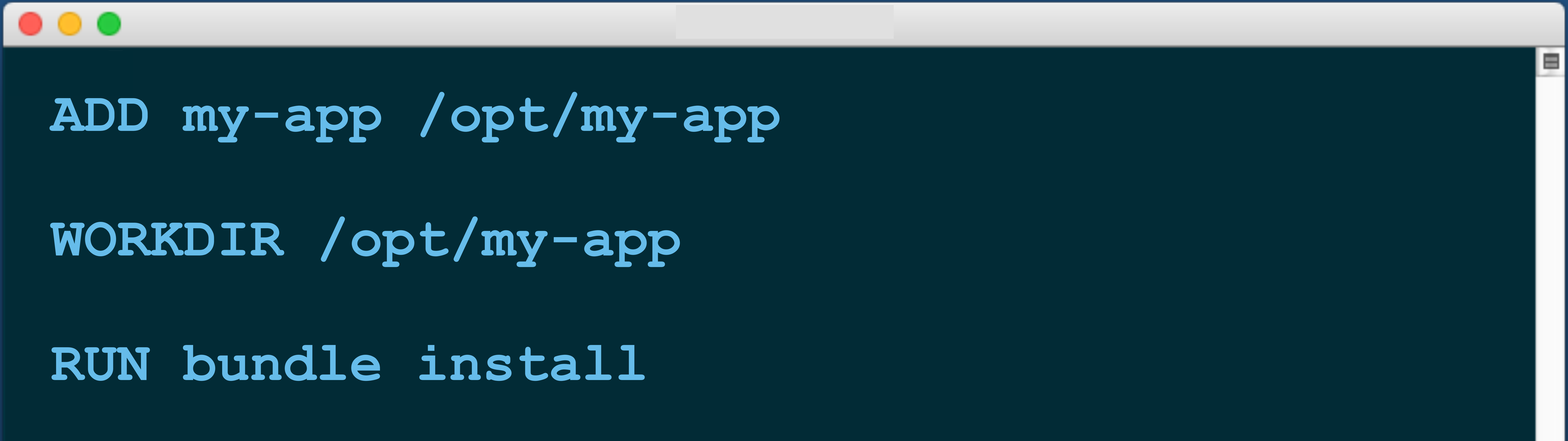


```
$ docker history myapp:stripped

IMAGE        CREATED        CREATED BY SIZE
ca132a1cae88 5 seconds ago             92.25 MB
```

Cache your build dependencies

How do I cache “bundle install” ?

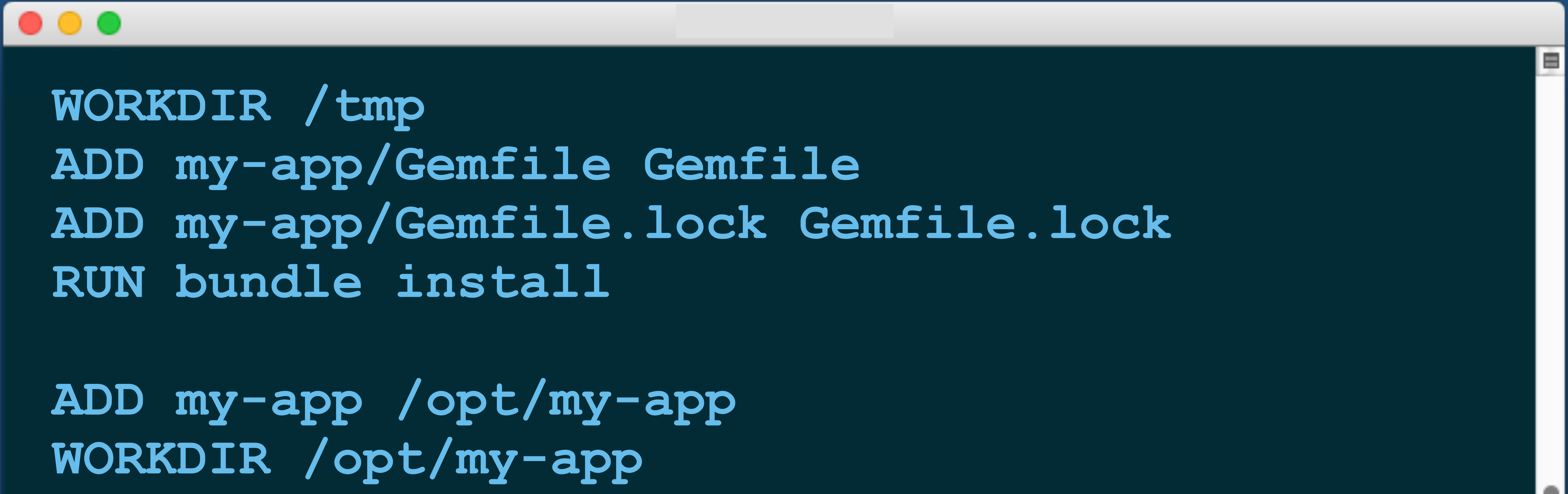
Split the dependency builder from the rest of the source code addition



```
ADD my-app /opt/my-app

WORKDIR /opt/my-app

RUN bundle install
```

How do I cache “bundle install” ?

Split the dependency builder from the rest of the source code addition



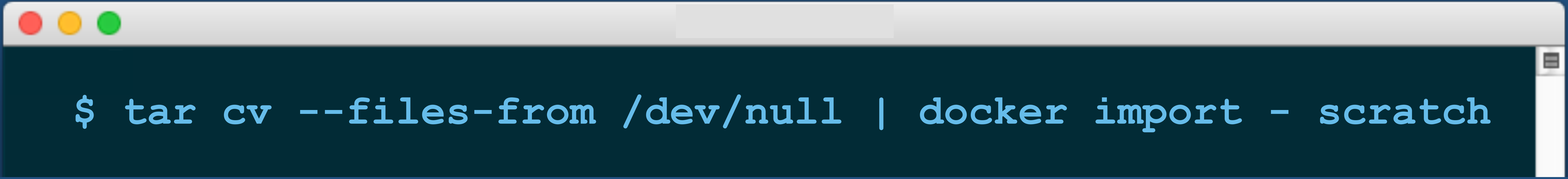
```
WORKDIR /tmp
ADD my-app/Gemfile Gemfile
ADD my-app/Gemfile.lock Gemfile.lock
RUN bundle install

ADD my-app /opt/my-app
WORKDIR /opt/my-app
```

Statically linked minimal apps running in containers? try Golang!

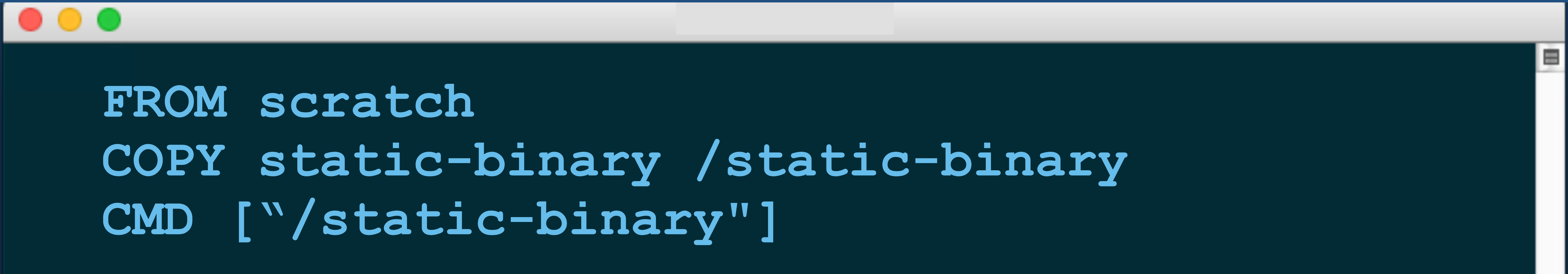
for more read http://bit.do/go-static by Adriaan De Jonge

The “secret” scratch image

Add static binary to the smallest container ever



```
$ tar cv --files-from /dev/null | docker import - scratch
```



```
FROM scratch
COPY static-binary /static-binary
CMD [“/static-binary"]
```

The Docker ecosystem is moving fast!

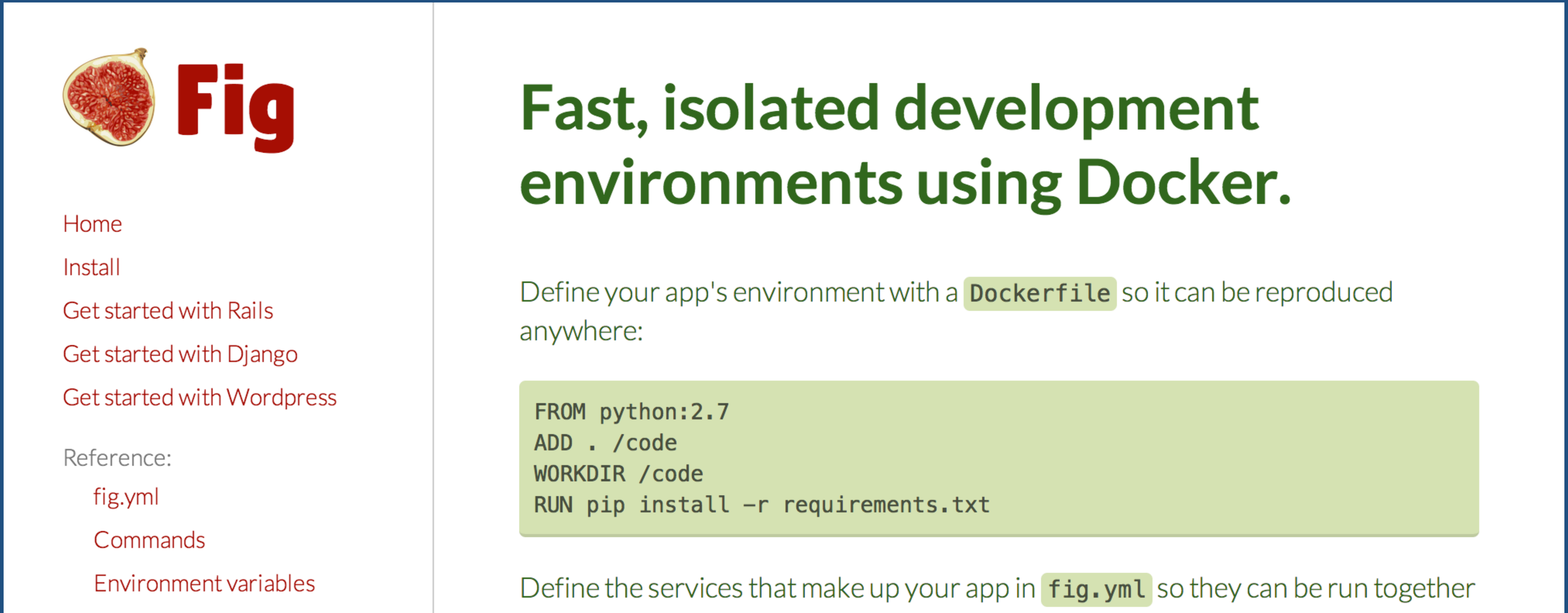
docker machine

docker swarm

docker compose

Isolated dev environments

http://orchardup.github.io/fig/



Fig

Home

Install

Get started with Rails

Get started with Django

Get started with Wordpress

Reference:

fig.yml

Commands

Environment variables

Fast, isolated development environments using Docker.

Define your app's environment with a `Dockerfile` so it can be reproduced anywhere:

```
FROM python:2.7
ADD . /code
WORKDIR /code
RUN pip install -r requirements.txt
```

Define the services that make up your app in `fig.yml` so they can be run together

The elevator is going up!

Atlassian

Thank you!

@durdn

NICOLA PAOLUCCI • DEVELOPER INSTIGATOR • ATLASSIAN • @DURDN