Docker:
an insider view

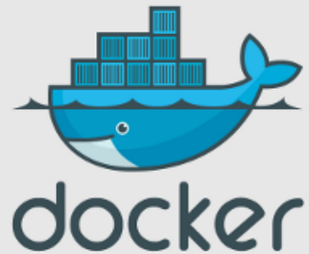
Google Developers Group Meetup
November 2013, Google West Campus 2

Michael Crosby — @crosbymichael
Jérôme Petazzoni — @jpetazzo
Victor Vieux — @vieux

docker

Outline

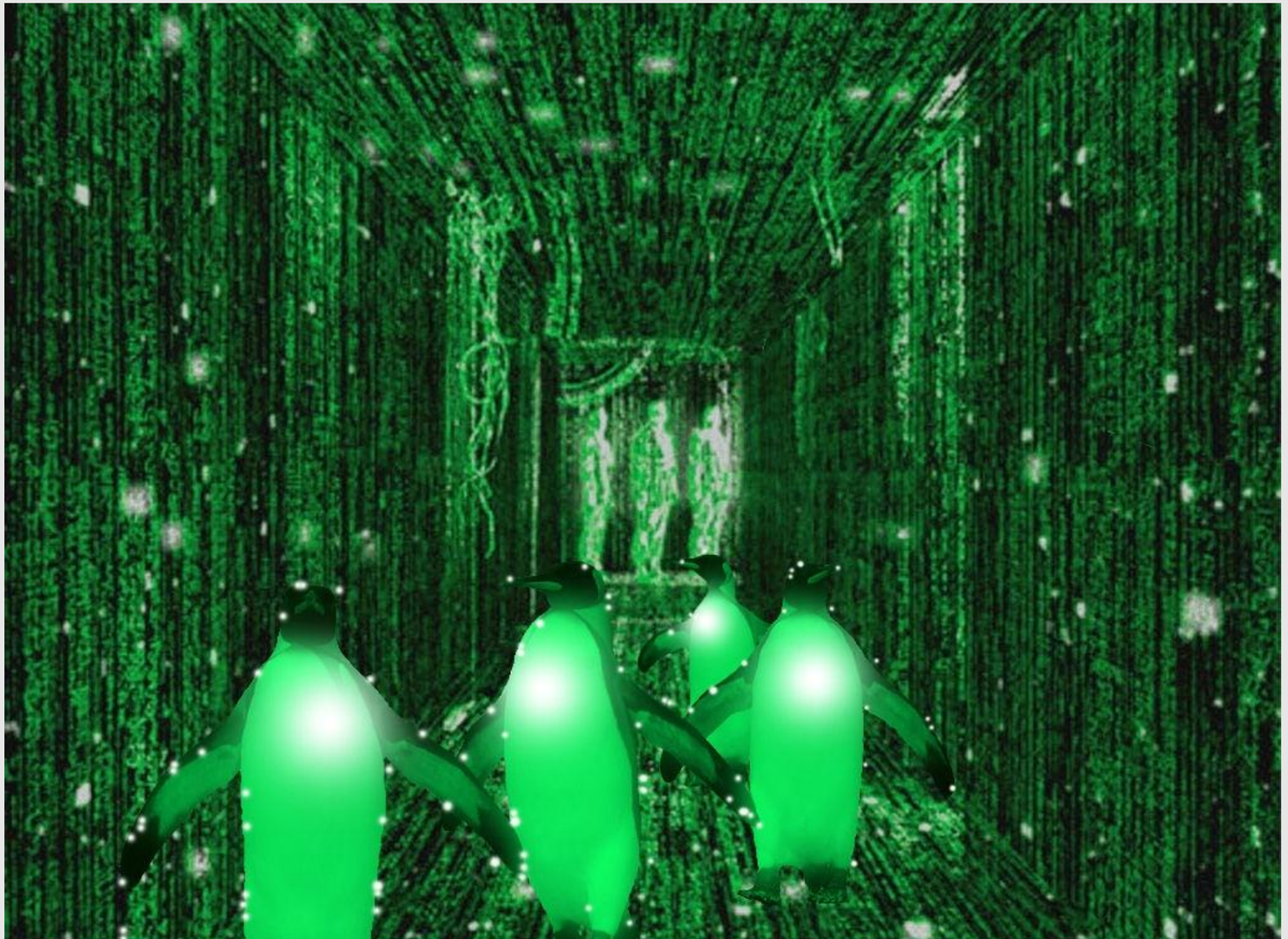
- what is Docker
- why it is written in Go
- some implementation details
- drawbacks
- more drawbacks
- discussion



What’s Docker?

Docker solves the “Matrix from Hell”

Docker solves the “Matrix from Hell”



docker

Docker solves the “Matrix from Hell”

static website	?	?	?	?	?	?
web frontend	?	?	?	?	?	?
background workers	?	?	?	?	?	?
user DB	?	?	?	?	?	?
analytics DB	?	?	?	?	?	?
queue	?	?	?	?	?	?
	dev VM	QA server	single prod server	on-site cluster	public cloud	contribu-tor laptop

Real-world analogy: containers

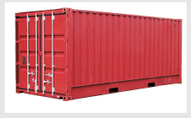
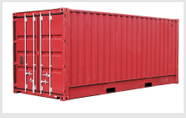
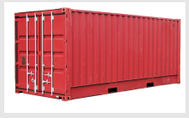
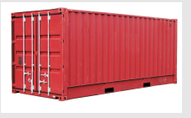
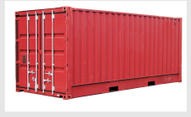
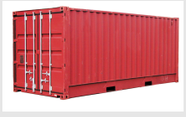
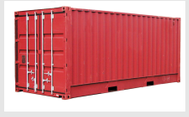
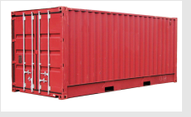
docker

Worldwide shipping: another Matrix from Hell

clothes	?	?	?	?
wine	?	?	?	?
coffee	?	?	?	?
electronics	?	?	?	?
cars	?	?	?	?
raw materials	?	?	?	?
	ships	trains	trucks	storage

docker

Solution:
the *intermodal shipping container*

clothes				
wine				
coffee				
electronics				
cars				
raw materials				
	ships	trains	trucks	storage

Solution to the deployment problem: the Linux container

docker

What's a Linux container?
High level approach

Chroot on steroids

- normal processes, but isolated
- share kernel with the host
- no device emulation (neither PV nor HVM)
- doesn't need a /sbin/init
  (runs the app/DB/whatever directly)

"Application Container"

docker

What's a Linux container?
Low level approach

Lightweight Virtual Machine

- own process space
- own network interface
- can run stuff as root
- can have its own /sbin/init (different from the host)

"Machine Container"

docker

Separation of concerns: Dave, from the dev team

- my code
- my framework
- my libraries
- my system dependencies
- my packaging system
- my distro
- my data

I don’t care where it’s running or how.

docker

Separation of concerns: Oscar, from the ops team

- logs
- backups
- remote access
- monitoring
- uptime

I don’t care what’s running in it.

docker

OK, so what’s Docker?

Runtime for Linux containers

```
jpetazzo@tarrasque:~$ sudo docker run -t -i ubuntu bash
root@092ee318746f:/#

jpetazzo@tarrasque:~$ sudo docker run -d johncosta/redis
c6000fa5ddc6

jpetazzo@tarrasque:~$ sudo docker inspect c6000fa5ddc6
...
    "NetworkSettings": {
        "IPAddress": "172.17.0.8",
        "Ports": {
            "6379/tcp": null
        }
    ...
```

docker

Standard format for containers, and a place to share them

- fetch an image from the registry with “docker pull”
- enter the image with “docker run”, do some changes
- record those changes with “docker commit”, repeat as many time as needed
- then share the result with “docker push”, on the public registry or on a private one

docker

Why Go?

The Five Reasons Why We Used Go

docker

Why Go?
1) static compilation

- “go build” will embed everything you need
  (no more “install this in order to run my stuff”)
- … except dynamic libraries if you use cgo
  (cgo lets you use any C library)
- and … except libc
  (but who doesn’t have libc?)
- you can have a real static binary
  (if you hack the build process a bit…)

docker

Why Go?
1) static compilation

- easier to install, easier to test, easier to adopt

- good candidate for bootstrap
  (i.e. the first installed thing,
  which will install the other things)

- “no dependencies? me gusta!”
  -- Anonymous BOFH

docker

Why Go?
2) neutral

- it’s not C++
- it’s not Python
- it’s not Ruby
- it’s not Java

docker

Why Go?
3) it has what we need

- good asynchronous primitives
  (wait for I/O, wait for processes…)
- low-level interfaces
  (manage processes, syscalls…)
- extensive standard library and data types
- strong duck typing

docker

Why Go?
4) full development environment

Go addresses multiple issues of the development workflow.

- go doc (see documentation for any package)
- go get (fetch dependencies on github etc.)
- go fmt (solve “tabs vs. spaces” once for all)
- go test (runs all Test* functions in *_test.go)
- go run (rapid script-like prototyping)

docker

Why Go?
5) multi-arch build

...without pre-processors

_linux.go
_darwin.go

docker

Drawbacks

Because that’s the
most interesting thing
in Amazon reviews

“Go doesn’t solve any problem”

Jonn Mostovoy @podmostom29 Sep

@benoitc @arnaudsj in not solving any problems. Everything they claim to solve is better solved in Dv2. 0, Erlang or will be solved in Rust..

But...

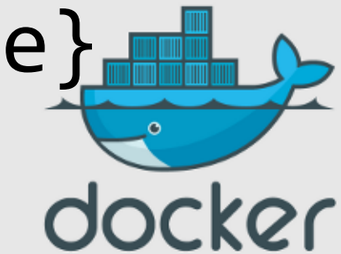
- Go is easier than Erlang
- Go is more real than Rust
- Don’t know about D 2.0 though!

docker

maps aren’t thread-safe

- deliberate decision: they’re *fast*
  (and it’s up to you to ensure that they’re *safe*)
- you have to protect access with sync.Mutex
- or, use channels of channels!

```
m := NewMap()
response := make(chan string)
m<-Get{Key: “fortytwo”, Replyto: response}
value := <-response
```



go get

- can’t pin a particular revision
- Docker had to vendor all dependencies (i.e. import their source code in our repo)
- must deal with private repos manually

docker

go test

- can’t have destructors/cleanups/… in tests
- use a test named “z_final_test.go”
- … which doesn’t work too well when running individual tests!

docker

go build

- it’s painful to build multiple binaries,
  when they share some common code

- each program has to be in its own package,
  and package must be “main”

you *have to* put shared/common stuff aside,
or use named import tricks (bleh)

docker

flag package

- doesn’t handle short/long options
  (-o --option)
- doesn’t handle options grouping
  (-abc -a -b -c)
- seriously just don’t use it
  (use getopt or go-flags instead)
  unless your primary target is Plan9 of course!

docker

no IDE

Let’s ask Samuel Jackson
what he thinks about it

docker

docker

Error handling can be verbose

```
if err := openFile(); err != nil {
    return err
}
if err := readAll(); err != nil {
    return err
}
if err := closeFile(); err != nil {
    return err
}
...
```

docker

Error handling

- panic / recover
  (don’t abuse it; only use it internally!)

- deploy one-off utility types
  http://talks.golang.org/2013/bestpractices.slide#5

docker

can’t select on readers/writers

```
select {
case data = <-inputChannel:
    …
case data = <-otherInputChannel:
    …
case data = connection.Read():
    …
}
```

docker

can’t select on readers/writers

- for readers:
  - start a goroutine with access to an output channel
  - the goroutine do blocking reads on the readers…
  - …and pushes the results on the output channel
- for writers:
  - adaptation is left as an exercise for the reader

→ reduce everything to channels!

docker

Some useful resources:

- The Holy Reference:
  http://golang.org/ref/spec
- Some useful patterns:
  http://golang.org/doc/effective_go.html
- Some best practices:
  http://talks.golang.org/2013/bestpractices.slide
- Embedded documentation:
  `godoc -http=:6060 & chrome http://localhost:6060`

Michael Crosby — @crosbymichael
Jérôme Petazzoni — @jpetazzo
Victor Vieux — @vieux

docker

Thank you!
Questions?

Michael Crosby — @crosbymichael
Jérôme Petazzoni — @jpetazzo
Victor Vieux — @vieux

docker