A million containers isn't cool
You know what’s cool?
A hundred containers
A million containers isn’t cool

You know what’s cool? A hundred containers.

@ChrisSinjo
GOCARDLESS
We aren’t #webscale (#sorrynotsorry)
So why do we care about containers?
POST /cash/monies HTTP/1.1

{ amount: 100 }

💰💰💰
High 💵 per-request
Reliability is 🚭
Deploying software reliably
Deploying software reliably

How containers can help
Deploying software reliably

How containers can help

Other options
First things first: deployment artifacts
Source code

↓

Something you can put on a server
A .jar file

A statically linked binary

An OS package (.deb, .rpm)
Some languages start on the back foot
Capistrano: a typical Ruby flow
On each server:
On each server:
- Clone source
On each server:

- Clone source
- Build dependencies
On each server:
- Clone source
- Build dependencies
- Run schema migrations
On each server:
- Clone source
- Build dependencies
- Run schema migrations
- Build static assets
On each server:
- Clone source
- Build dependencies
- Run schema migrations
- Build static assets
- SIGHUP
What’s wrong here?
Hope
On each server:
- Clone source
- Build dependencies
- Run schema migrations
- Build static assets
- SIGHUP
On each server:
  - Clone source
  - Build dependencies
  - Run schema migrations
  - Build static assets
  - SIGHUP
$ bundle install
...
Building nokogiri using system libraries.

Gem::Ext::BuildError: ERROR: Failed to build gem native extension.
On each server:

- Clone source
- Build dependencies
- Run schema migrations
- Build static assets
- SIGHUP
On each server:
- Clone source
- Build dependencies
- Run schema migrations
- Build static assets
- SIGHUP
On each server:
- Clone source
- Build dependencies
- Run schema migrations
- Build static assets
- SIGHUP
“Hope is not a strategy.”

– Traditional SRE saying

There's something else
Applications don’t run in a vacuum.
Ruby app
Ruby app

Ruby dependencies
Ruby app

Nokogiri

libxml2

Ruby dependencies

Native libraries
Ruby app

Nokogiri

libxml2

Ruby dependencies

Native libraries
Ruby app

- Nokogiri
- libxml2

Ruby dependencies

Native libraries
How do we install software?
Nokogiri

libxml2
$ bundle install
Nokogiri

$ bundle install

libxml2

$ apt-get install libxml2
Nokogiri

libxml2

App’s source repository

Chef or whatever
That seems inconvenient...
Container images: totally a thing
Nokogiri -> App’s source repository

libxml2 -> Chef or whatever
Nokogiri

libxml2

App’s source repository

App’s source repository
This is why most people care about Docker
namespaces
cgroups
images
namespaces
cgroups
images
OH: @ChrisSinjo "Docker is a fat-jar for people not on the JVM"
Deploying software reliably

How containers can help

Other options
Deploying software reliably

How containers can help

Other options
So what did we care about?
Uniform deployment
Uniform deployment

Based around an artifact
Uniform deployment

Based around an artifact

Fail early
And what didn't we care about?
Know what your aims are not
Distributed schedulers
Scheduler

compute compute compute compute compute compute ...
Scheduler

Apps:
- App
- App
- App

Computations:
- compute
- compute
- compute
- compute
- compute
...
Nothing comes for free
Kubernetes means:
Kubernetes means:
– a distributed scheduler
Kubernetes means:

- a distributed scheduler
- cluster DNS
Kubernetes means:
- a distributed scheduler
- cluster DNS
- etcd
Kubernetes means:

– a distributed scheduler
– cluster DNS
– etcd
– ...
Nothing comes for free
We aren't webscale

(#sorrynotsorry)
Distributed schedulers
Distributed schedulers
So what did we build?
3 parts...
Service definitions
A service:
A service:
—an image
A service:

- an image
- environment config
A service:
- an image
- environment config
- command to run
A service:

- an image
- environment config
- command to run
- limits (memory, CPU)
A service:

- an image
- environment config
- command to run
- limits (memory, CPU)
- ...

...
This is config management
So we used Chef
Chef

Service A

Service B

Service C
Chef

Service A
Service B
Service C

Compute 1
Service A
Service B

Compute 2
Service B
Service C

Compute 3

config
Service definitions
Service definitions

Single-node orchestration
Enter Conductor
conductor service upgrade
   --id gocardless_app_production
   --revision 279d903588
conductor service upgrade
   --id gocardless_app_production
   --revision 279d903588
conductor service upgrade
  --id gocardless_app_production
  --revision 279d903588
The flow:
The flow:

— start containers for new version
The flow:

- start containers for new version
- wait for health check
The flow:

— start containers for new version
— wait for health check
— rewrite local nginx config
The flow:

- start containers for new version
- wait for health check
- rewrite local nginx config
- reload nginx
The flow:

— start containers for new version
— wait for health check
— rewrite local nginx config
— reload nginx
— stop old containers
Conductor

nginx

Docker
Conductor

traffic

nginx

traffic

Old

Docker
Conductor

- nginx

Old

New

Docker

API

**traffic**

**traffic**
Conductor

nginx

Old

New

Docker

traffic

health check

traffic

traffic
Conductor

nginx

traffic

Docker

API

Old

New

traffic

Old
What about cron jobs?
conductor cron generate
  --id gocardless_cron_production
  --revision 279d903588
conductor cron generate
   --id gocardless_cron_production
   --revision 279d903588
gocardless/

▼ app/

  payment_stuff.rb

▶ lib/

  generate-cron
# Clean up expired API tokens

*/30  *  *  *  *  scripts/cleanup-api-tokens
# Clean up expired API tokens

*/30 * * * * /usr/local/bin/conductor run
   --id gocardless_cron_production
   --revision 279d903588
   scripts/cleanup-api-tokens
Service definitions

Single-node orchestration
Service definitions

Single-node orchestration

A way to trigger deploys
Keep it boring
Keep it in Capistrano
Capistrano

deploy

Legacy infra

deploy

New infra
Help developers do their job
1 thing missing
“Hey, this process died.”

– a computer
Supervisor

start

Process

Process

Process
Some supervisors:
Some supervisors:

— Upstart
Some supervisors:

- Upstart
- systemd
Some supervisors:

- Upstart
- systemd
- runit
Those didn’t play well with Docker
Docker restart policies
We didn’t get along well
Hard to stop
Hard to stop
or
Gave up entirely
We built a process supervisor 😬
conductor supervise
Specifically:
Specifically:

— check number of containers
Specifically:

— check number of containers
— health check each container
Specifically:

- check number of containers
- health check each container
Specifically:

- check number of containers
- health check each container
- restart if either fails
Specifically:

- check number of containers
- health check each container
- restart if either fails
- at most every 5 seconds
# service conductor-supervise stop
We don’t want this piece of software
Deploying software reliably

How containers can help

Other options
Deploying software reliably

How containers can help

Other options
systemd + rkt
Supervisor: systemd
Supervisor: systemd

Containers: rkt
To fit our usage:
To fit our usage:

– Conductor generates systemd config
To fit our usage:

- Conductor generates systemd config
- systemd manages processes
To fit our usage:

– Conductor generates systemd config
– systemd manages processes
– Delete conductor supervise
To fit our usage:

- Conductor generates systemd config
- systemd manages processes
- Delete conductor supervise
- HTTP health checks???
systemd + rkt

or

VMs + autoscaling
Supervisor: autoscaling
Supervisor: autoscaling

Containers → VMs
Meta-thoughts
Meta-thoughts
Some reckons 🤔
Introduce new infrastructure where failure is survivable.
Non-critical batch jobs

↓

Background workers

↓

API servers
Goal state is what matters
Everything might change before your next method call.
The system isn’t interesting without context.
Start with why
Thank you❤️

@ChrisSinjo
@GoCardlessEng
We’re hiring
Questions?

@ChrisSinjo
@GoCardlessEng