Building a PaaS in Clojure

Allen Rohner

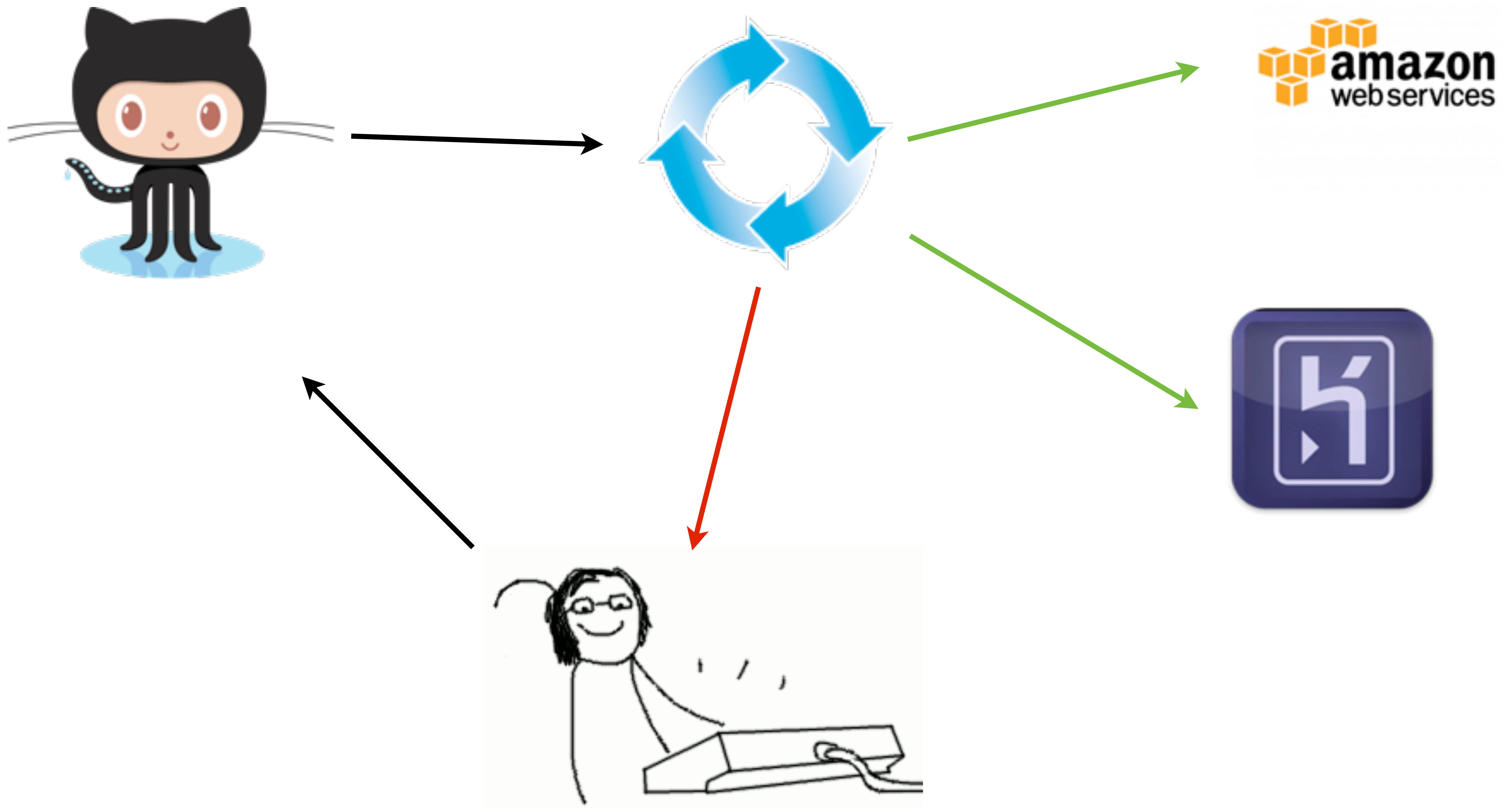
About Me

- Started using Clojure in 2008
- Using professionally since 2009
- Two startups in Clojure

Second Annual
Allen Talks about Circle

Mostly a Success Story

CircleCI



Scale

- 300 companies
- 1000 developers
- 4k builds/day
- 6k VMs
- 350/builds/hour @ peak

What does PaaS mean?

- Auto-scale EC2
- Create and Manage VM
  - all the languages
  - all the DBs
  - all the developer tools
- Fast

Systems

- AWS
  - ec2, s3, elb, ebs, r53
- LXC
- Clojure!

LXC

- Chroot on Steroids
- Largest AWS instance *
- 16 cores
- 12 LXC instances

Low level examples

- modify firewall rules at runtime
- install packages
- create & mount filesystems
- tweak FS args
- parse dmesg for OOM

No Bash!

Anywhere!

Libraries

- clj-aws (need to OSS)
- clj-ssh
- pallet
- stevedore
- wait-for
- clj-http
- clojure.core.memoize
- clj-time

Advantages

- Higher level abstractions
- exceptions
- concurrency
    - future
    - delay
- syntax

Advantages, cont.

- Code as data!
- single configuration
  - single place for logging
  - single place for DB

Stevedore

- Parenscript
- Scriptjure
- Stevedore

https://github.com/pallet/stevedore

Syntax

```
(sh/q (foo) (bar))
=>
“foo; bar”
```

Quasiquoting

```
(sh/q (git log -1 ~commit))
```

Variations

```
(sh/q-chain (foo) (bar))
=>
“foo && bar”
```

Executing

```
(sh/sh “hostname”)
=>
{:exit 0, :out "bahamut\n", :err ""}
```

Executing

```
(sh/shq (hostname))
=>
{:exit 0, :out "bahamut\n", :err ""}
```

exceptions!

```
(sh/shq! (bogus))
=> *e
#<ExceptionInfo clojure.lang.ExceptionInfo:
   {:object {:exit 127,
             :out "",
             :err "bash: line 1: bogus: command not found\n"}
```

Pallet

- Library of existing fns
- (mostly) Declarative VM specification

```
(postgres/settings
  (postgres/settings-map
    {:version "9.1"
     :options {}
     :permissions
       [{:connection-type "local" :auth-method "trust"}
        {:connection-type "host" :ip-mask "127.0.0.1/32" :auth-method "trust"}
        {:connection-type "host" :ip-mask "::1/128" :auth-method "trust"}]}))
(postgres/initdb)
(postgres/hba-conf)
(replace-line "/etc/postgresql/9.1/main/postgresql.conf" "^#fsync" "fsync = off")
```

Timeouts

```
(bash (bundle exec rspec spec) {:keepalive (time/minutes 3)
                                :timeout (time/hours 1)})
```

Streams

Side-Effects

- Nearly everything we do is side-effecting
- Nearly everything takes wall clock time
  - mounting & umounting
  - copying
  - wait for machine startup

wait-for

```
(wait-for #(foo))
```

https://github.com/circleci/wait-for

wait-for

```
(wait-for {:tries 3} #(foo))
```

wait-for

```
(wait-for {:sleep (time/secs 5)
           :timeout (time/minutes 2)} #(foo))
```

wait-for

```
(wait-for {:catch [:exit 1]} #(foo)) ;; sh! slingshot
```

wait-for

```
(wait-for {:catch [:status 404]} #(foo)) ;; clj-http slingshot
```

Putting It Together

```
(def remote-home
  (memo/memo-ttl
   (fn [node]
     (->> (sh/shq node (echo “REMOTE_HOME=$HOME"))
          :out
          (re-find #"REMOTE_HOME=(.*)")
          (second)
          (str/trim)))
   (-> 60 time/minutes ->millis)))
```

Starting up

```
(wait-for
  {:sleep (time/secs 10)
   :timeout (time/secs 90)
   :catch [java.net.ConnectException com.jcraft.jsch.JSchException]}
  #(ssh/shq node (echo “hello")))
```

Process running?

```
(defn running? [str]
  (let [out (-> (sh/shq (pgrep -f ~str)) :out str/trim)]
    (when (seq out)
      (->int out))))
```

Challenges

JSch

Buggy!

Unmaintained!

Pallet

Futures

```
(safe-future (wait-for #(send-build-results-email)))
```

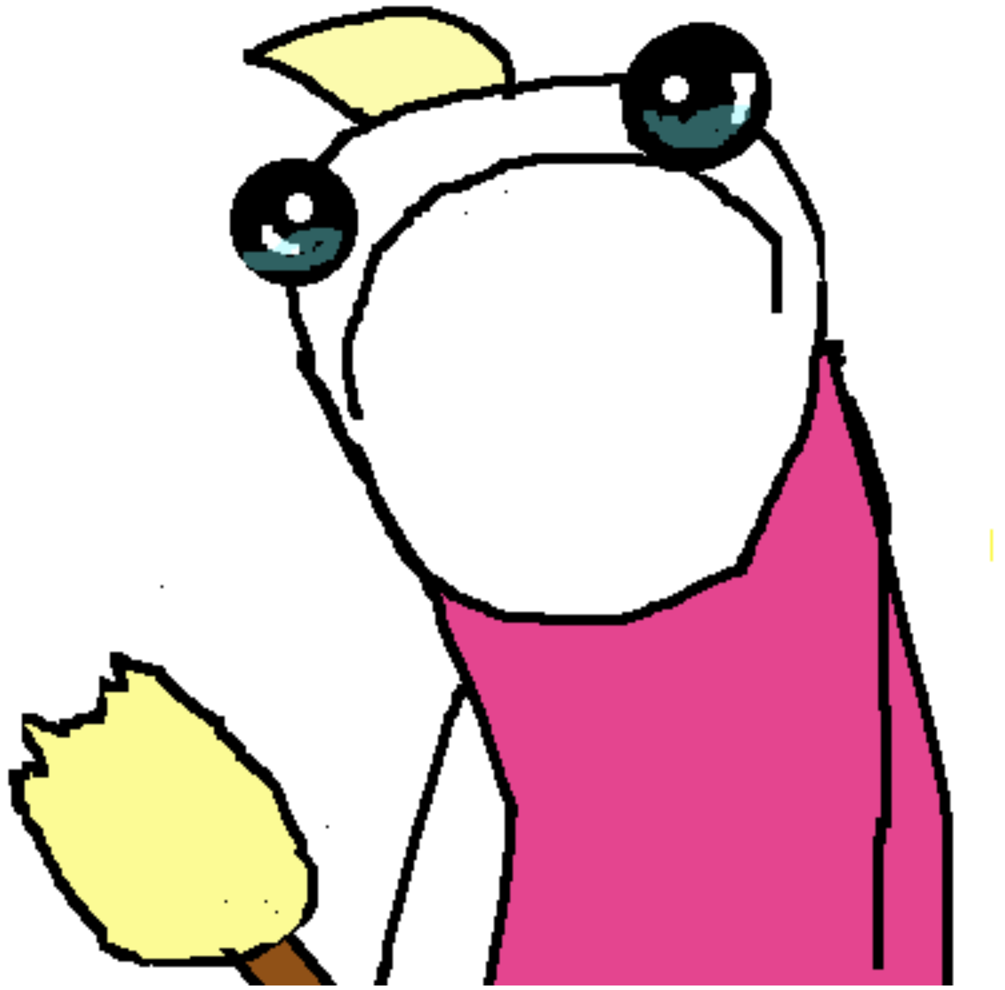
- retries
  - API failure
  - machine failure
- logging
- reporting

That sounds like a queue!

Serialize!

Things you can't Serialize:

- fns
- the stack
- lazy seqs
- bindings
- refs



Dequeuing

- Find a build
- mark queue job in-progress
- Acquire VM
- update DB
- mark queue job finished
- Transaction with side-effects!

Transactional Queue?

Ephemeral Data

- Which build is running on which EC2 instance
- How many concurrent builds a user is running

Redis?

- Non-transactional
- Another API

Redis is a failure in DB Design

Datomic

- {noHistory true}
- {noPersistence true}?

Hiring!

- Scaling problems
- Distributed Systems
- Queing
- Big Data
- jobs@circleci.com

Questions?