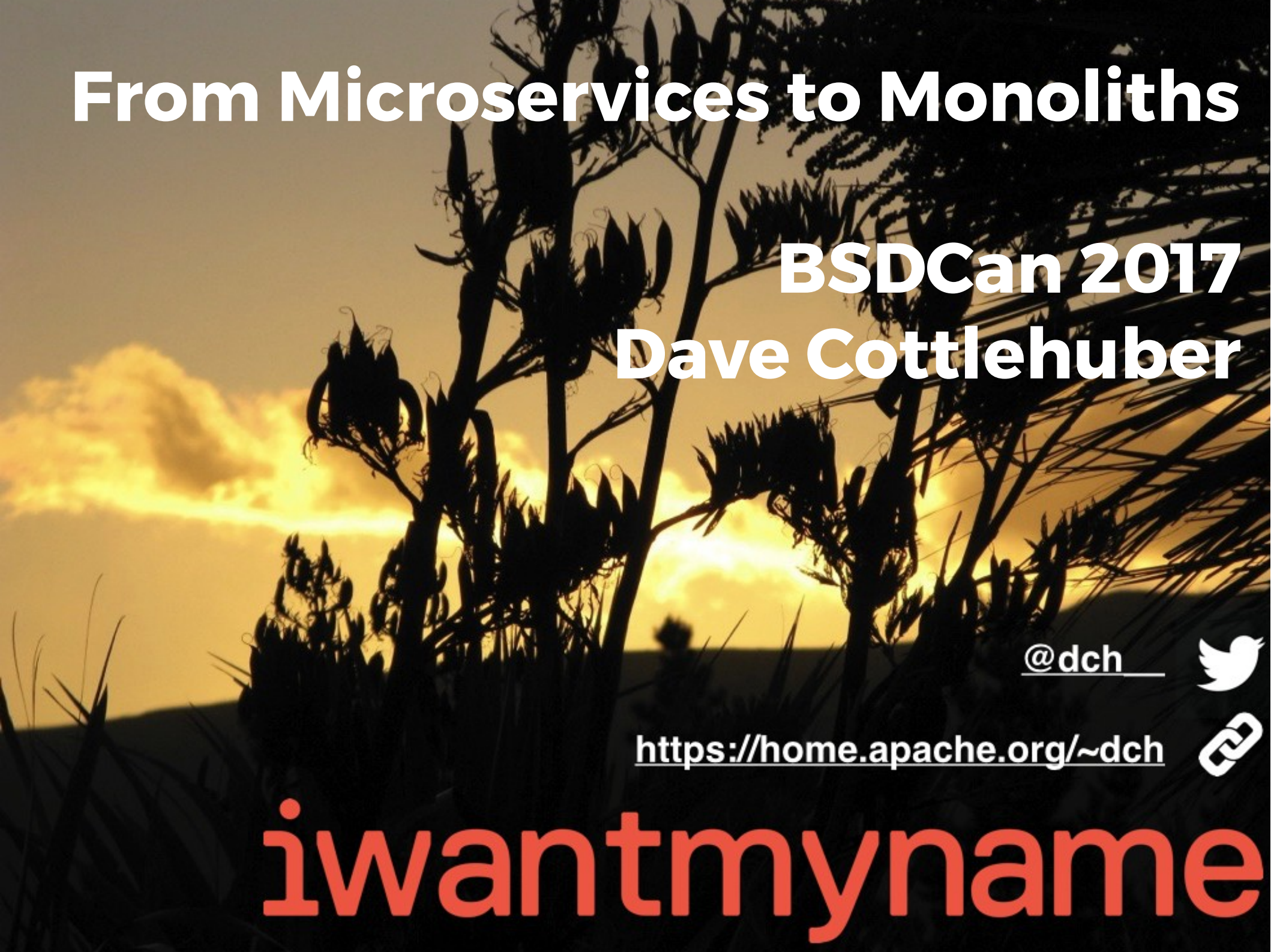

From Microservices to Monoliths
BSDCan 2017
Dave Cottlehuber
@dch
https://home.apache.org/~dch
iwantmyname

iwantmyname

- Hand-Crafted Artisanal Domain Resellers
- Domains for Real People
- Ethical business
- Simple interface
- NZ + global distributed team

- Perl Catalyst front end
- Erlang backend: RabbitMQ, Apache CouchDB, Search App
- Kyoto Tycoon Session Store
- 20 Debian Unstable VMs

This Talk

- why we migrated & to what
- maintaining & building our source
- architecture
- moving the back end
- interlude 1
- the batbelt – vault & ansible
- cluster/jail internals
- interlude 2
- metrics & monitoring
- interlude 3
- Q&A

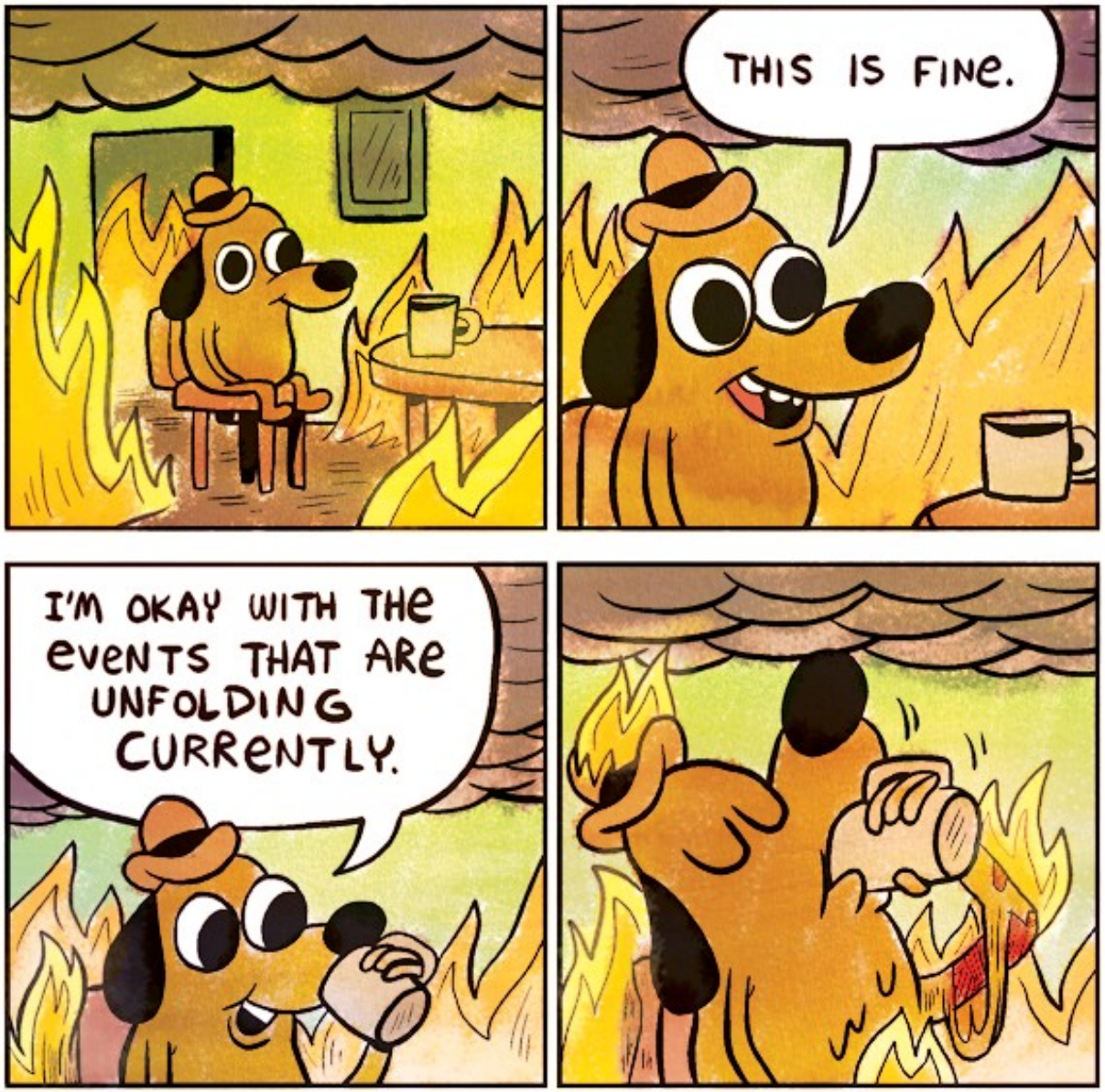
Why Migrate?

- Microservices-induced Latency

- VM-induced network instability

- Debian-induced base package variability – OpenSSL, Net::HTTP

- Upgrade Hell – no turning back



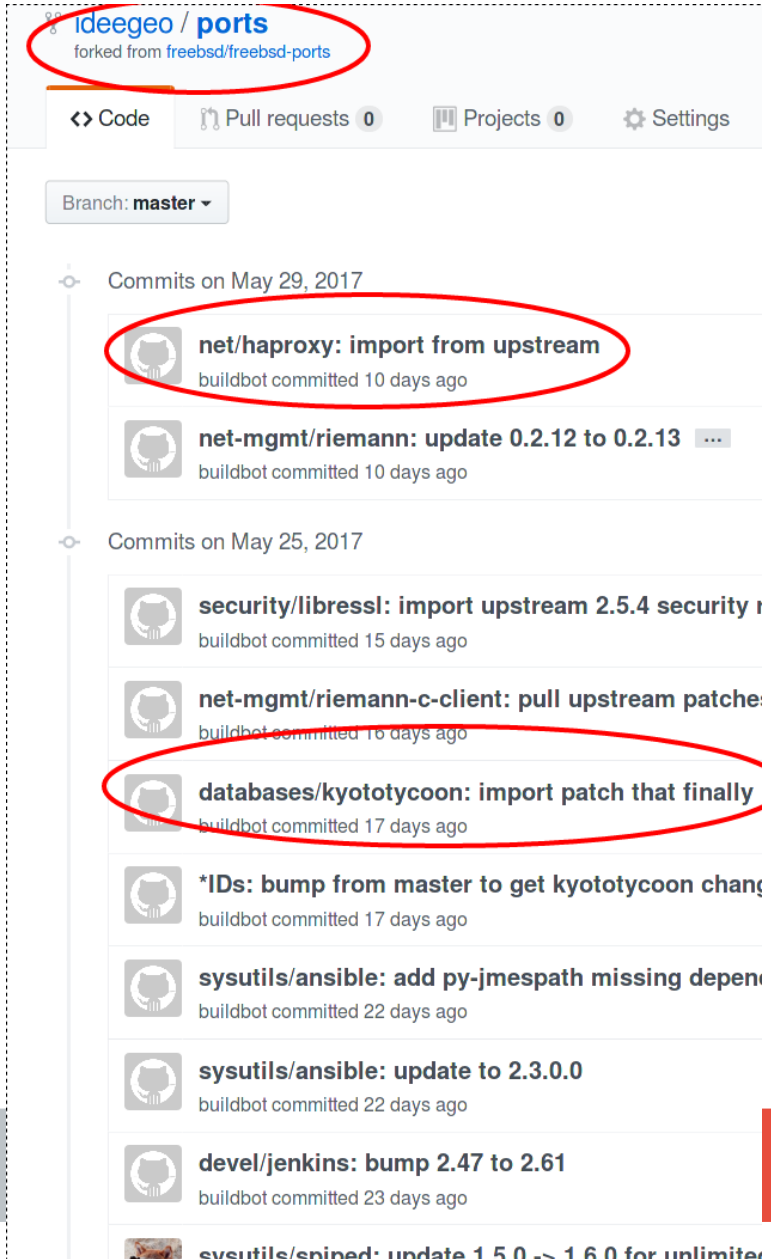
Choose Wisely

The Holy Grail

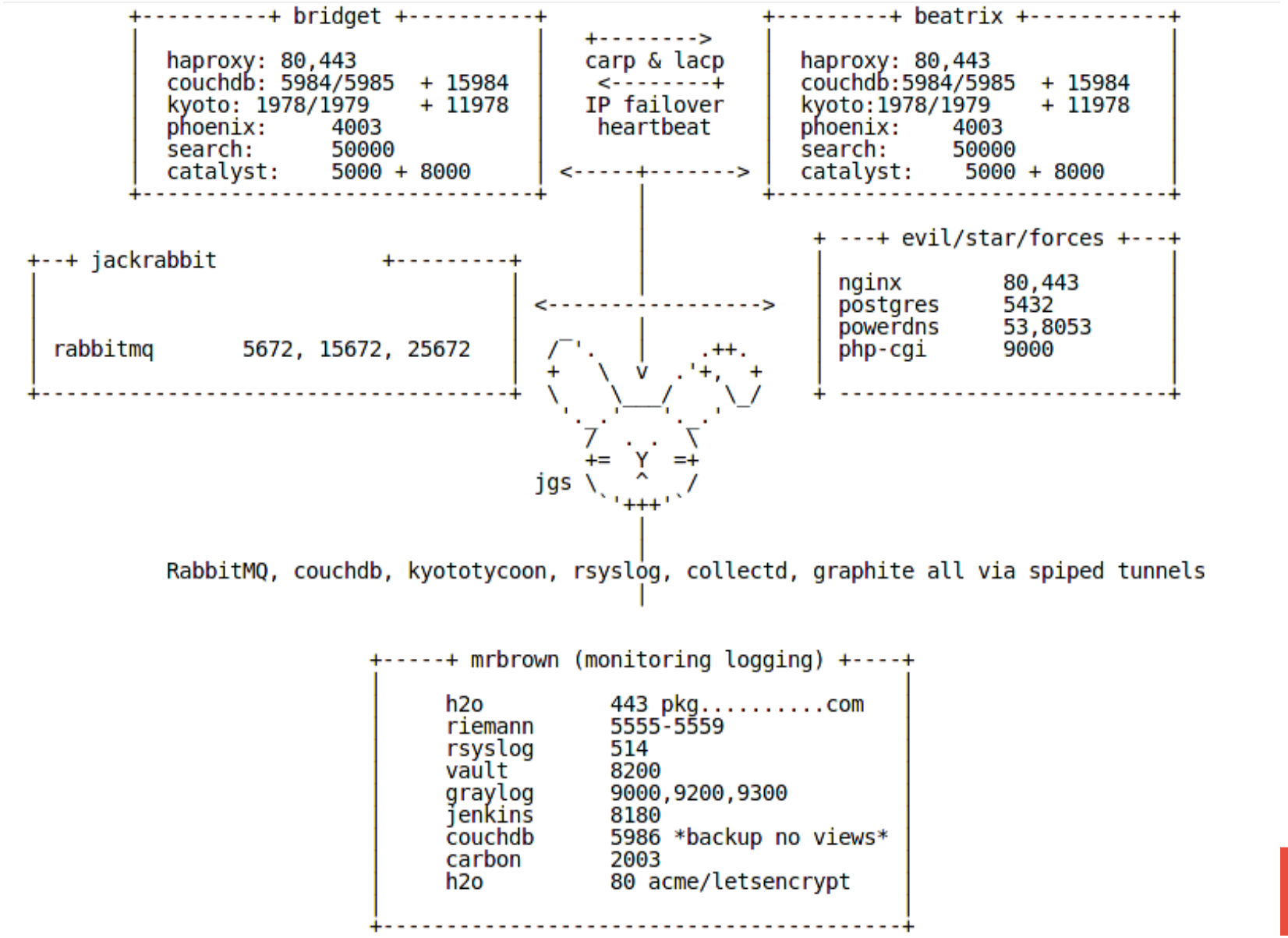
- reliable std & custom packages
- easy rollbacks for Dbs & upgrades
- ansible for lower non-sysadmin learning curve
- robust transparent failover for app & backend
- spiped tunnels for resilience
- stable API for containers – no fuss, no latency
- 3 physical machines (A + B cluster nodes, C sysadmin/backup)
- a few VMs during migration for eventual consolidation

Poudriere – Floating Ports

- custom git fork of github:freebsd-ports
- custom ports + private packages
- best of both worlds
- bump every 2-3 months when it suits
- pull in security fixes only when needed
- no rush for upstream committers ;-)
- guaranteed consistency across dev/prod



Architecture



Backwards First

Multiple nodes all behind 

All distributed, reliable, boring tech:

- Session: Kyoto Tycoon
- Store: Apache CouchDB – Erlang/OTP JSON doc DB
- Queue: RabbitMQ – Erlang/OTP

Interlude the First

- tested load through spiped tunnels based on 2x loading
- started DB migration to FreeBSD cluster
- on Debian systems we switched over spiped tunnels
- perfect until the last system
- roll back and reviewed logs
- tweaked FreeBSD network stack
- tried again → looking good → go to watch Rogue One
- no cell reception in the theatre ...

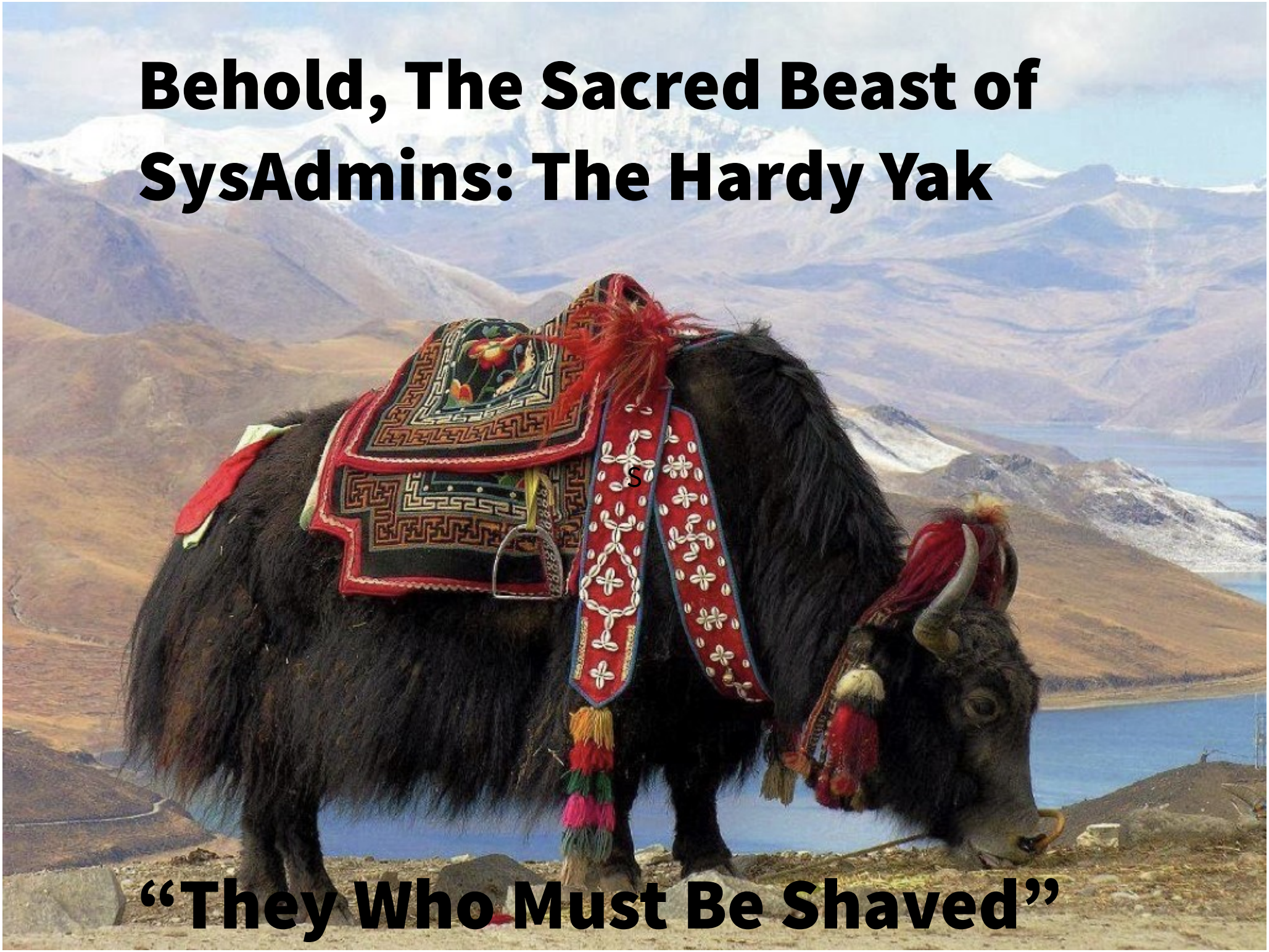
Interlude the First

> - [72288] sonewconn: pcb 0xfffff80d68996cb0: Listen queue overflow: 16
> already in queue awaiting acceptance (64492 occurrences)
>
> 1. Am I using spiped incorrectly by having it handle many short-lived
> connections? There are usually about 100, sometimes up round 200,
> concurrent HTTP connections most of the time, spread across 6 main
> "client" servers that all connect to the same tunnel endpoint.

"... spiped has a default limit of 100 connections. You can increase this up to 500 via the -n option; if you need more than that, we'll need to fix the networking code to use poll instead of select (entirely doable; it just hasn't happened yet)."

– cperciva@

Behold, The Sacred Beast of SysAdmins: The Hardy Yak



“They Who Must Be Shaved”

SysAdmin Batbelt

- vault – secrets management
- ansible – config management
- rsyslog* – centralised log shipping
- graylog* – log aggregation & search
- collectd – host-based metrics agent
- riemann – centralised event correlation & realtime dashboard
- graphite & grafana – pretty pictures and trends
- poudriere & jenkins – custom ports and applications

Hashicorp Vault

- key value store specifically for storing secrets
- Shamir's Secret Sharing algorithm prevents theft
- github auth yields time-limited admin tokens
- mostly pushed out via ansible
- no secrets in git
- no redundancy just S3 backend
- future plans

Hashicorp Vault

```
$ export VAULT_ADDR=https://vault.foo.com:8200/
$ vault auth -method=github token=<your_token>

Successfully authenticated! You are now logged in.
The token below is already saved in the session. You do
not
need to "vault auth" again with the token.
token: 7271d370-aa45-0629-b3b4-e3a7f150974b
token_duration: 7775999
token_policies: [admins, default]

$ cat ~/.vault-token
7271d370-aa45-0629-b3b4-e3a7f150974b
```

Blackadder

```
$ vault write secret/blackadder scarlet_pimpernel="we do not know"

Success! Data written to: secret/blackadder


$ vault write secret/blackadder scarlet_pimpernel="comte de frou frou"

Success! Data written to: secret/blackadder


$ vault read -format=json secret/blackadder

{

"request_id": "36592ad4-b8e0-8a72-e4a5-540bcc6d3a4d",

"lease_id": "",

"lease_duration": 7776000,

"renewable": false,

"data": {

"scarlet_pimpernel": "comte de frou frou"

},

"warnings": null

}
```

```
$ vault read -format=yaml secret/blackadder

data:

  scarlet_pimpernel: comte de frou frou

lease_duration: 7776000

lease_id: ""

renewable: false

request_id: 901a598f-3b6b-e0a9-9fbd-eba625f2aa3f

warnings: null


$ vault read -field=scarlet_pimpernel secret/blackadder

comte de frou frou


$ vault delete secret/blackadder

Success! Deleted 'secret/blackadder' if it existed.
```

Ansible

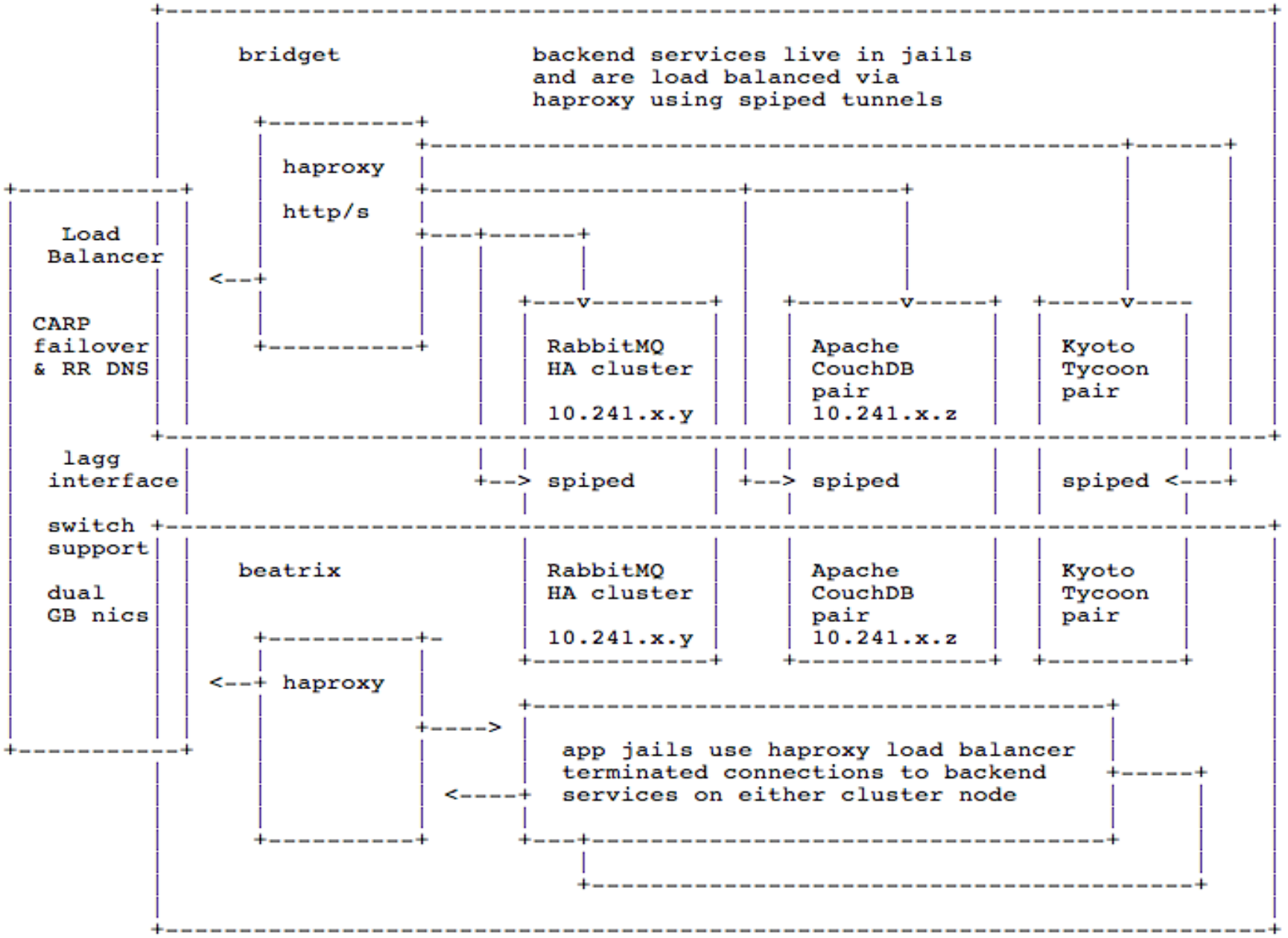
- simpler on-ramp for non-sysadmins
- jail support is *local only*
- proper jail & vault support via plugins
- thanks xmj@ for both moral support and fractal cells
- 50+ roles: pf, zfs, spiped, iocell jail creation, patch & reboot

Ansible

```
# jail definition
---
config:
  iocell:
    instances:
      iwmnbase:
        ipv4: "{{ net.private.ip.iwmnbase }}"
        packages: |
          lang/python2
          security/sudo
          devel/gmake
          devel/git
          devel/p5-App-cpanminus
          www/p5-LWP-Protocol-https
          dns/p5-Net-LibIDN
          textproc/p5-XML-SAX-Expat
        properties:
          allow_raw_sockets:    1
```

```
# jail app config
---
config:
  iwmnbase:
    catalyst:
      primary_port: 5000
      root_path:    /usr/local/d8o/iwmnbase/...
      debug:        0          # 0|1 starman ...
      testmode:     0          # catalyst test ...
      app:          debug      # (app|debug).psgi ...
    log:
      debug:        0          # enable debug
      level:        info       # debug only in dev
    nginx:
      root_path:    /usr/local/d8o/iwmn-frontend/templates
    hipchat:
      token:   "{{ lookup('vault',
                   'secret/iwmnbase',
                   'hipchat_token') }}"
```

Cluster


bridget
backend services live in jails
and are load balanced via
haproxy using spiped tunnels
haproxy
http/s
Load
Balancer
CARP
failover
& RR DNS
RabbitMQ
HA cluster
10.241.x.y
Apache
CouchDB
pair
10.241.x.z
Kyoto
Tycoon
pair
lagg
interface
spiped
spiped
spiped
switch
support
dual
GB nics
beatrix
RabbitMQ
HA cluster
10.241.x.y
Apache
CouchDB
pair
10.241.x.z
Kyoto
Tycoon
pair
haproxy
app jails use haproxy load balancer
terminated connections to backend
services on either cluster node

Interlude the Second

- DBs migrated & first apps behind haproxy
- one morning, *everything* wedged completely
- different everything - networks, h/w vs VMs, service providers
- 3 other people reported same event, same time
- FreeBSD 11.x only, not 10.x, different haproxy versions
- gremlins? Aliens? Port scanners? Ping'o'death?

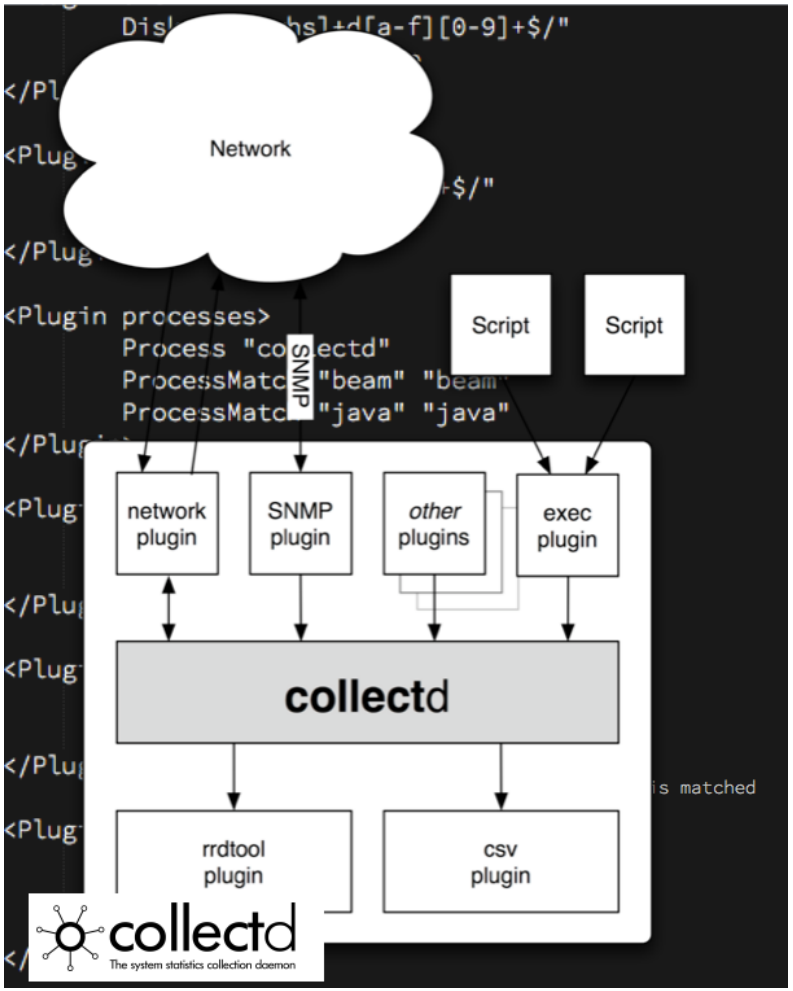
Interlude the Second

"Haproxy relies on signed integer wraparound on overflow, however this is really an undefined behavior, so the C compiler is allowed to do whatever it wants, and clang does exactly that, and that causes problems when the timer goes from <= INT_MAX to > INT_MAX, and explains the various hangs reported on FreeBSD every 49.7 days. To make sure we get the intended behavior, use -fwrapv for now. [....]

Many thanks to David King, Mark S, Dave Cottlehuber, Slawa Olhovchenkov, Piotr Pawel Stefaniak, and any other I may have forgotten for reporting that and investigating."

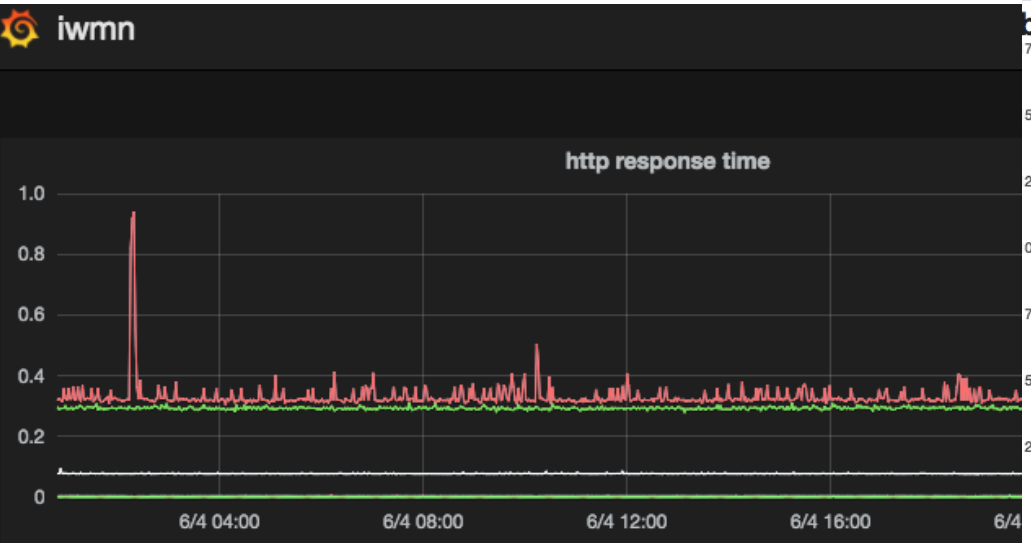
Pro Ops Yak Herder Jedi

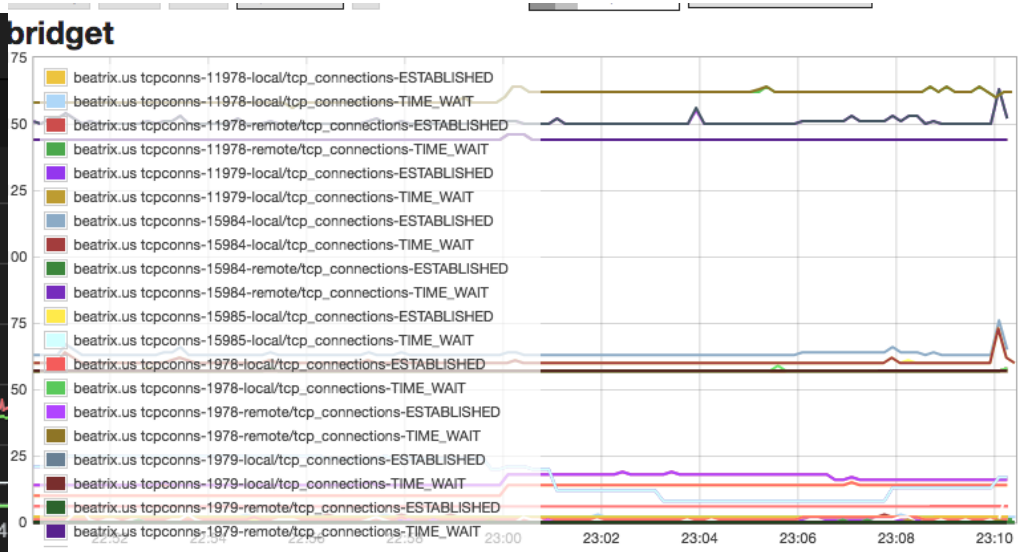
OS via CollectD

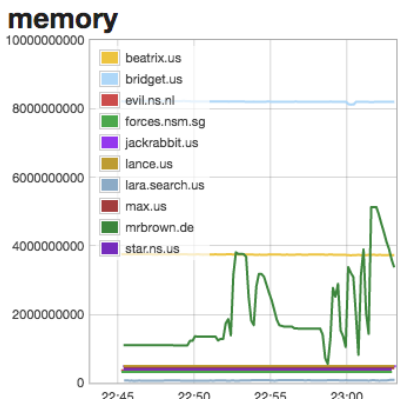


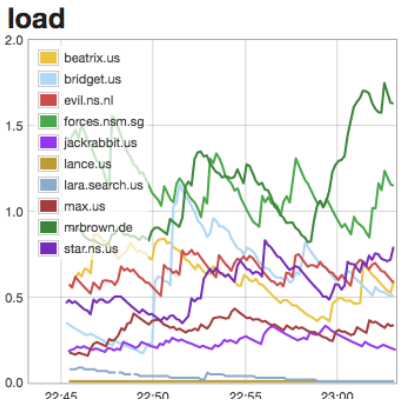
- zfs, cpu, disk, memory
- per-process stats
- tcp stats per port
- rabbitmq queue length
- basic thresholds
- no rrd, snmp
- output to riemann & grafana

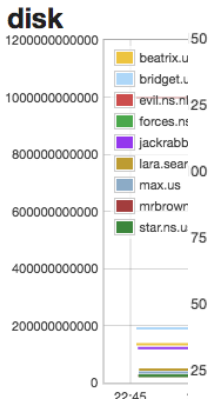
Riemann – Real Time

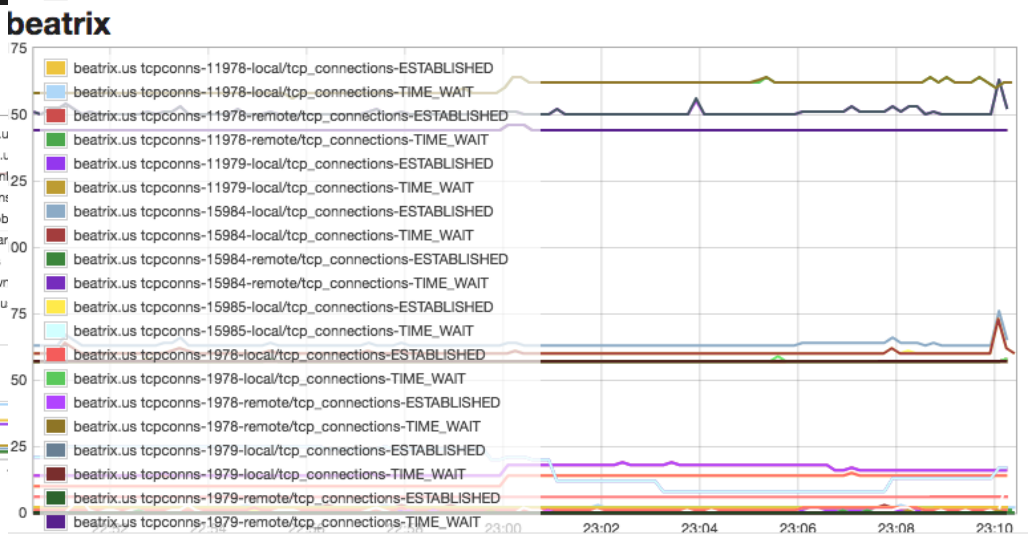




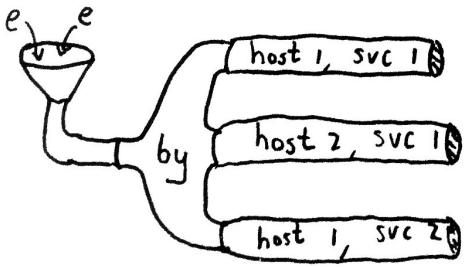


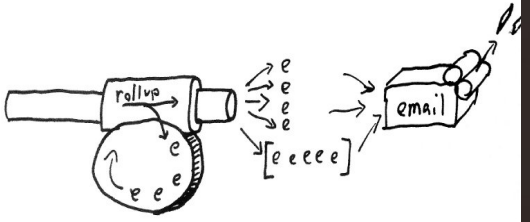






Riemann - Streams

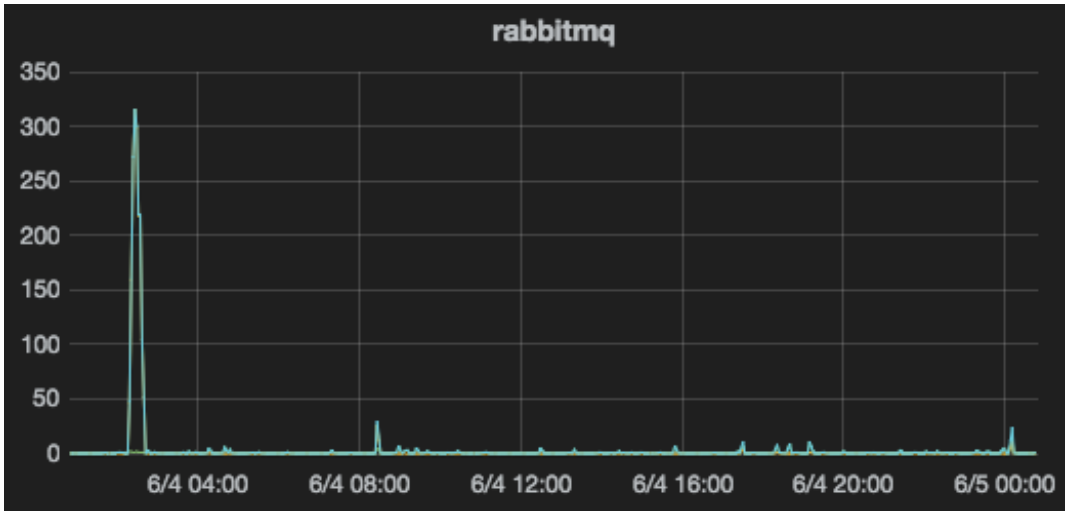





```
(streams
  (default  {:ttl        300
             :state      "ok"
             :service    "undefined"
             :host       "riemann.{{ config.domain }}"}

    ; index all events immediately before any further transformation
    index
    ; events explicitly tagged for pagerduty or pushover always get sent
    (where (tagged "push") push)
    (where (tagged "page") pager)
    ; flip state if queue backlog grows unreasonably
    (where (service "curl_json-rabbitmq/gauge-queue_totals-messages")
           (with {:ttl      120
                  :service  "rabbitmq backlog"
                  :tags     ["notify"]}
             (where (or (> metric 300) (state "expired"))
                    (with :state "critical" reinject)
                    (else
                      (with :state "ok" reinject))
                    )))
    (where (service "rabbitmq backlog") pager)
    ; send to local graphite instance
    graph
    ; Calculate an overall rate of events and insert into index
    ; clear tags and description first
    (with {:metric       1
           :service      "riemann.events.second"
           :host         "riemann.{{ config.domain }}"
           :tags         "riemann"
           :description  nil}
      (rate 5 index))
```

Riemann – App Events



jackrabbit.us rabbitmq monitoring **critical** Mon Jun 05 2017 01:06:44 GMT+0200 (CEST) ttl 120 tags: rabbitmq

jackrabbit.us rabbitmq monitoring **ok** Mon Jun 05 2017 01:03:08 GMT+0200 (CEST) ttl 120 tags: rabbitmq

rabbit

Monitoring operational

	jackrabbit.us
rabbitmq monitoring	ok
rabbitmq.object_totals.channels	0
rabbitmq.object_totals.connections	0
rabbitmq.object_totals.exchanges	7
rabbitmq.object_totals.queues	116
rabbitmq.queue_totals.messages	1
rabbitmq.queue_totals.messages_ready	0
rabbitmq.queue_totals.messages_unacknowledged	1

	beatrix.us	bridget.us
curl-kyoto-primary/gauge-connections	3	3
curl-kyoto-primary/gauge-records	50958	51410
curl-kyoto-primary/gauge-repl_delay	0	0
curl-kyoto-primary/gauge-repl_interval	0	0
curl-kyoto-primary/gauge-sys_mem_peak	428040192	411619328
curl-kyoto-primary/gauge-sys_mem_rss	428040192	411619328
curl-kyoto-primary/gauge-sys_mem_size	428040192	411619328

Interlude the Third

- databases, message queues, catalyst app all migrated
- lets move the job queue processors
- I plan 2 weeks holiday in Sardinia starting Saturday
- major backend problems – but site is still usable**
- some connections to rabbitmq are decidedly flakey
- tweak all the things
- un-tweak all the things
- check spiped tunnels
- slow panic sets in
- I finally work it out after tcpdump/ngrep at 23h00 Friday

My Bad

- haproxy connections are tuned for HTTP keep-alive
- aggressive timeouts on connections to avoid spiped blowout
- rabbitmq reverse proxied connections inherit these defaults
- queues with low or no message volume have their TCP sessions dropped
- perl is not fast enough on restarting workers to keeppm

- enable heartbeats in all rabbit libraries that support them
- perl ofc doesnt but erlang / elixir do
- change haproxy idle connection time to

- I make the holiday trip, albeit tired & with laptop


Question Time

Thanks

- emaste
- cperciva
- xmj
- swills
- koobs
- mdexter

- everybody@ports
- everybody@re
- everybody@docs
- BSDNow & Techsnap ‘casters


Thanks!
Questions?
@dch__
https://home.apache.org/~dch
iwantmyname