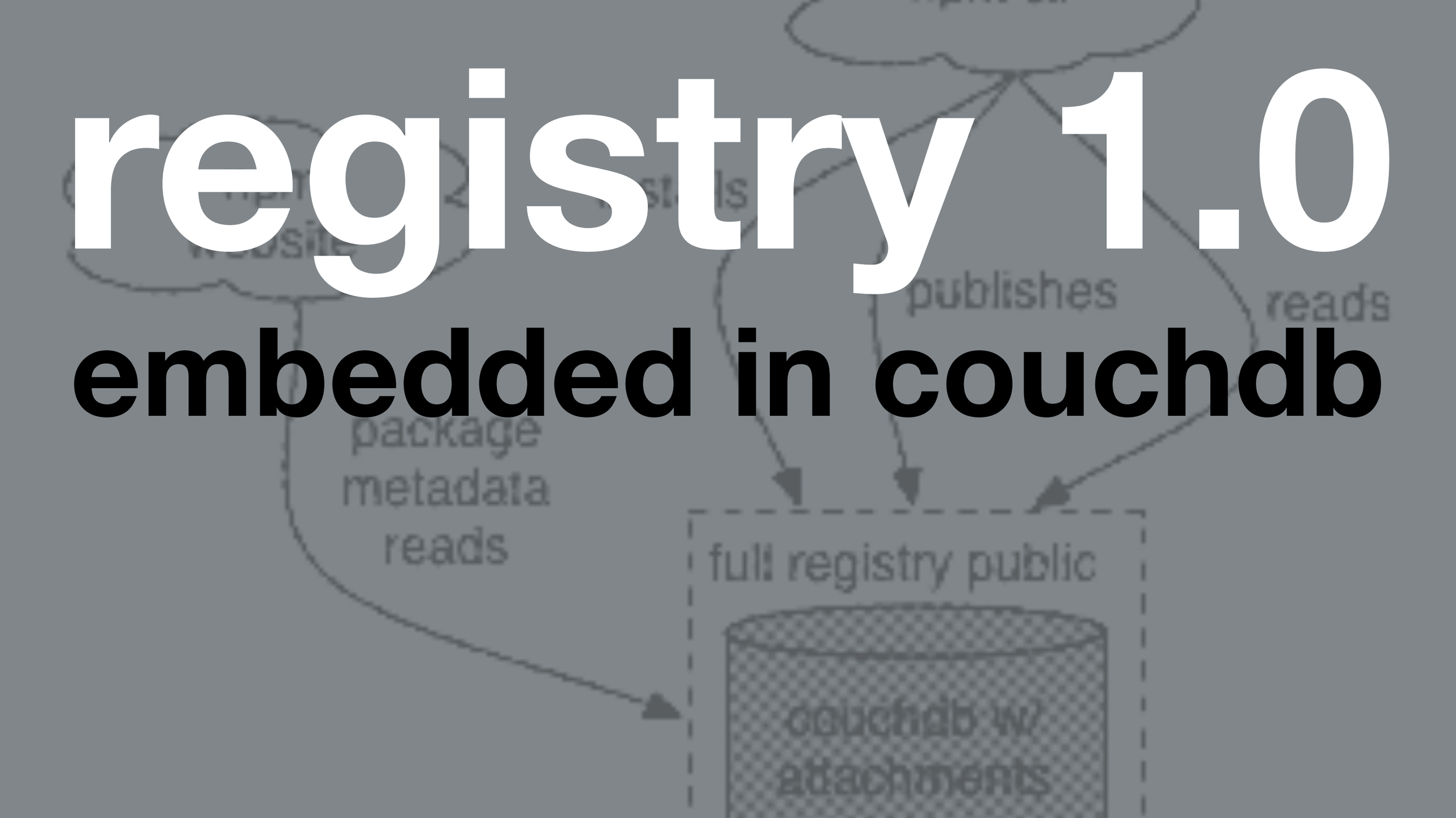
npm registry

dev-ops deep-dive

C J Silverio

director of engineering, npm

@ceejbot


registry 1.0
embedded in couchdb
publishes
reads
package
metadata
reads
full registry public
couchdb w/
attachments

javascript

but not node

the shame, the shame

advantages

- hey! it was a simple working system
- couchdb's replication made mirrors easy
- didn't have to implement auth
- got away with storing package tarballs as couch attachments
- worked for a longer time than we deserved

disadvantages

— all of this fell over at scale
— tarballs fell over first
— we aren't erlang experts
— not modular; hard to work on

late 2013: stay up

— pulled out tarballs into Joyent Manta
— put varnish in front of everything
— fastly CDN for geolocality

early 2014: stability

- tarballs onto a file system
- found & stomped problems with our couchdb installation
- load-balanced everything
- operational maturity
- big sign of success: many mirrors shut down

now we're stable!

npm's next goal:

be self-sustaining

end 2014: rewrite

— we are node experts!
— microservices: node's natural architecture
— future scaling
— ability to add features easily
— scoped modules!

scoped modules aka namespaces

- `hyperfs`: **the famous module**
- `@mikeal/hyperfs`: **super-hip fork**
- `@ceejbot/hyperfs`: **my completely unrelated private module**

Everybody can make public scoped modules. $7/month and you can create private scoped modules.

team

- 3 engineers on the registry & operations
- 2 engineers on the website
- 2 engineers on the command-line client

shipped the core of it

as npm-enterprise

"npm in a box" service

(our other way to make $)

had a working registry in node

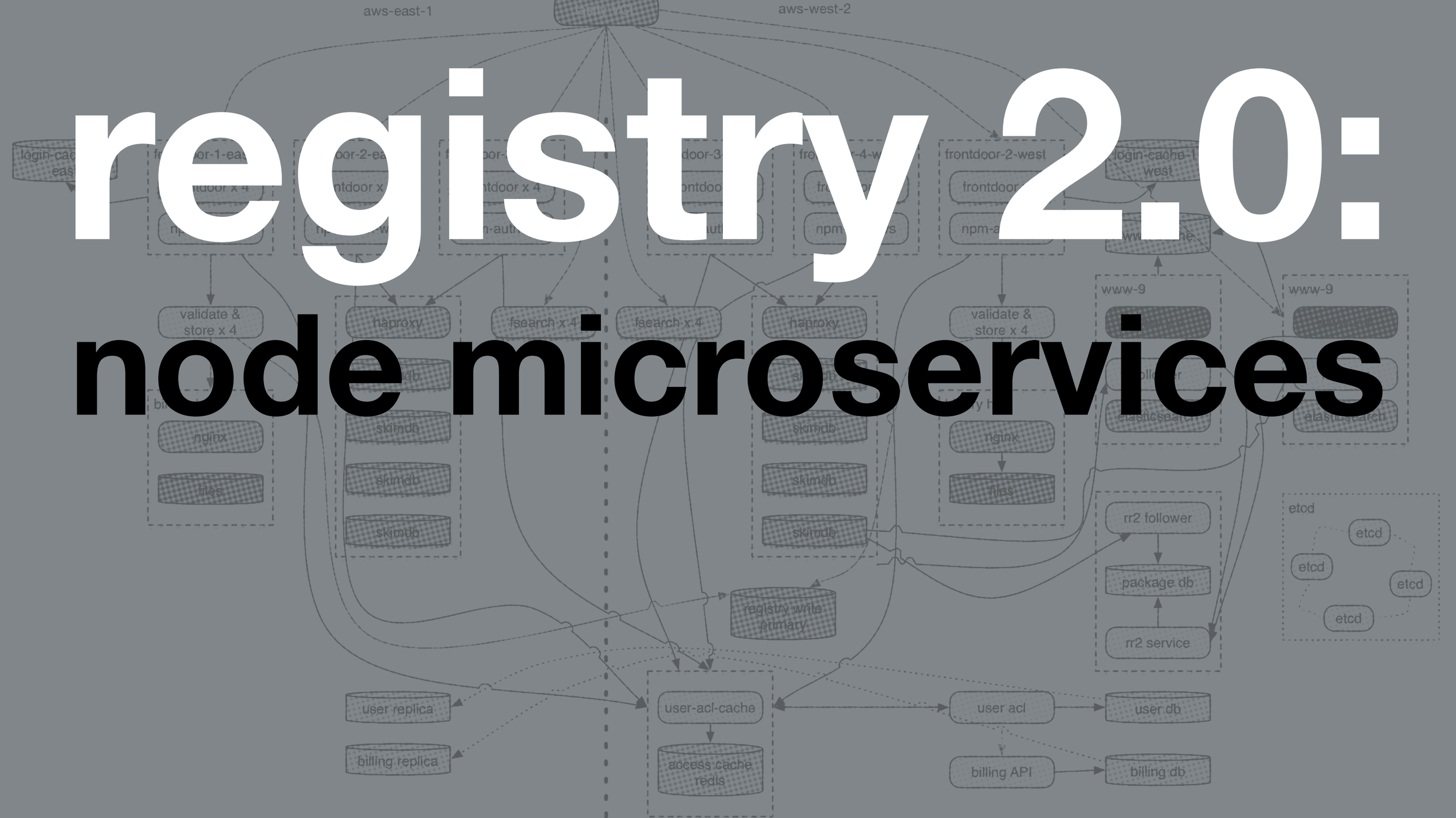
before we migrated the

public registry to it

in production April 2015

scoped modules

were a feature flip



registry 2.0:
node microservices

the stack (top)

— Fastly as our CDN (faster in Europe!)
— AWS EC2
— Ubuntu Trusty
— nagios + PagerDuty
— Github hosts our code
— TravisCI for public & private repos

the stack (middle)

— haproxy for load balancing & tls termination
— a couple instances of pound for tls (legacy)
— nginx for static files
— redis for caching

the databases

- couchdb for package data storage
- postgres for users, billing, access control lists
- replica of the package data in postgres to drive website

big node modules

— web site only: hapi
— everything else: restify
— knex to help with postgres

restify

— barely a framework
— trivial to get a json api running
— observable
— sinatra/express routing
— we like the connect middleware style

conventions across services

- monitoring endpoints same for all
- every process has a repl
- json logging
- config mostly through cmd-line arguments
- some environment variable passing

configuration via etcd

https://github.com/coreos/etcd

A highly available key/value store intended for config & service discovery. We recursively store & extract json blobs from it using `renv`.

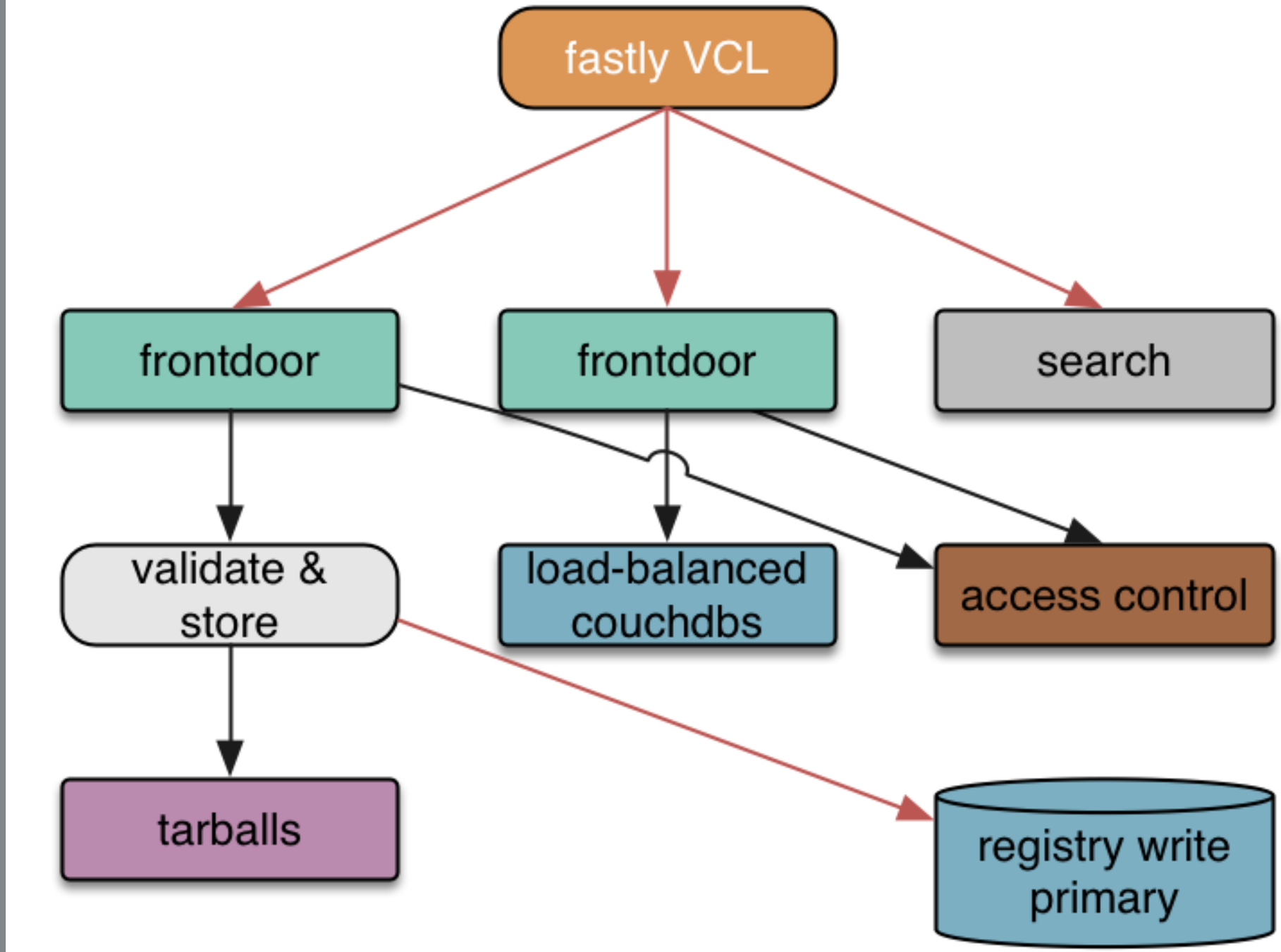
`ndm` tool transforms json into command-line options in an upstart script.

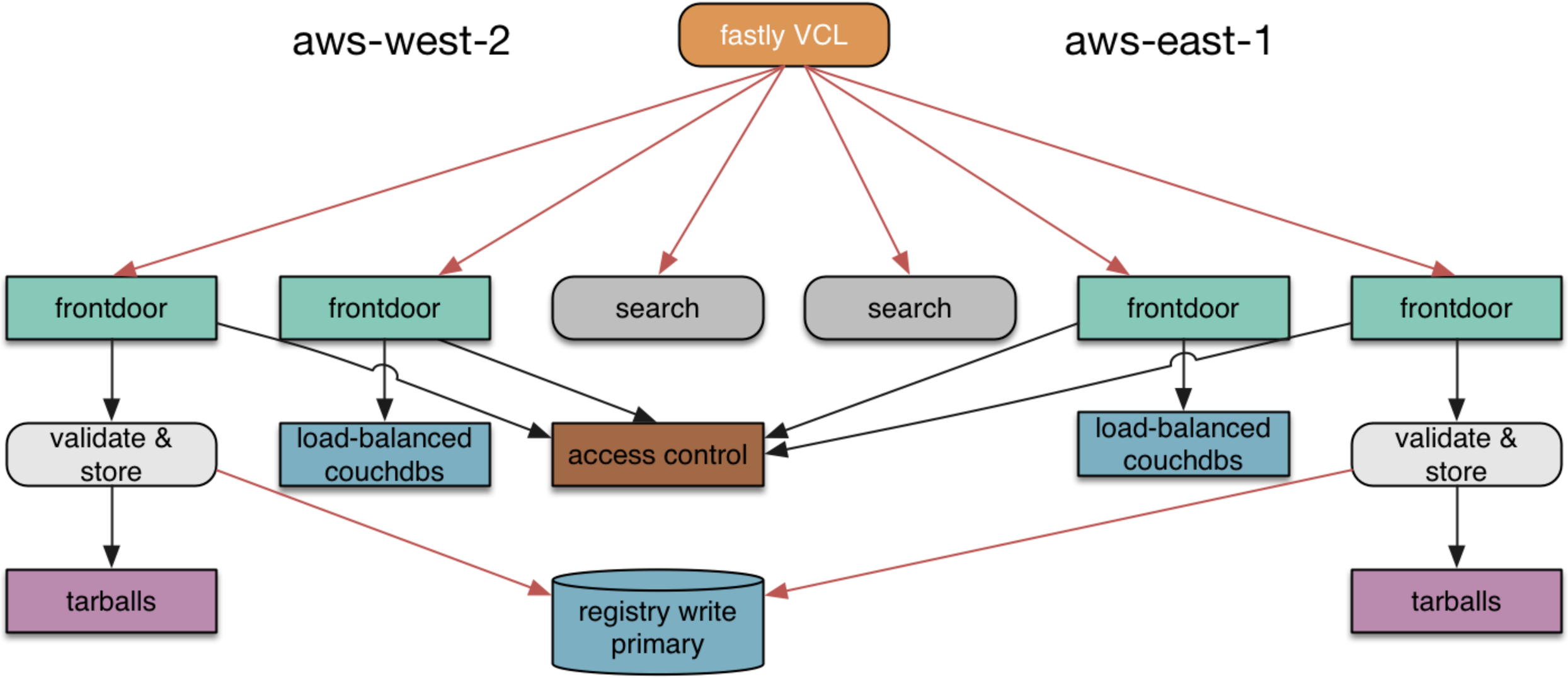
automation via ansible

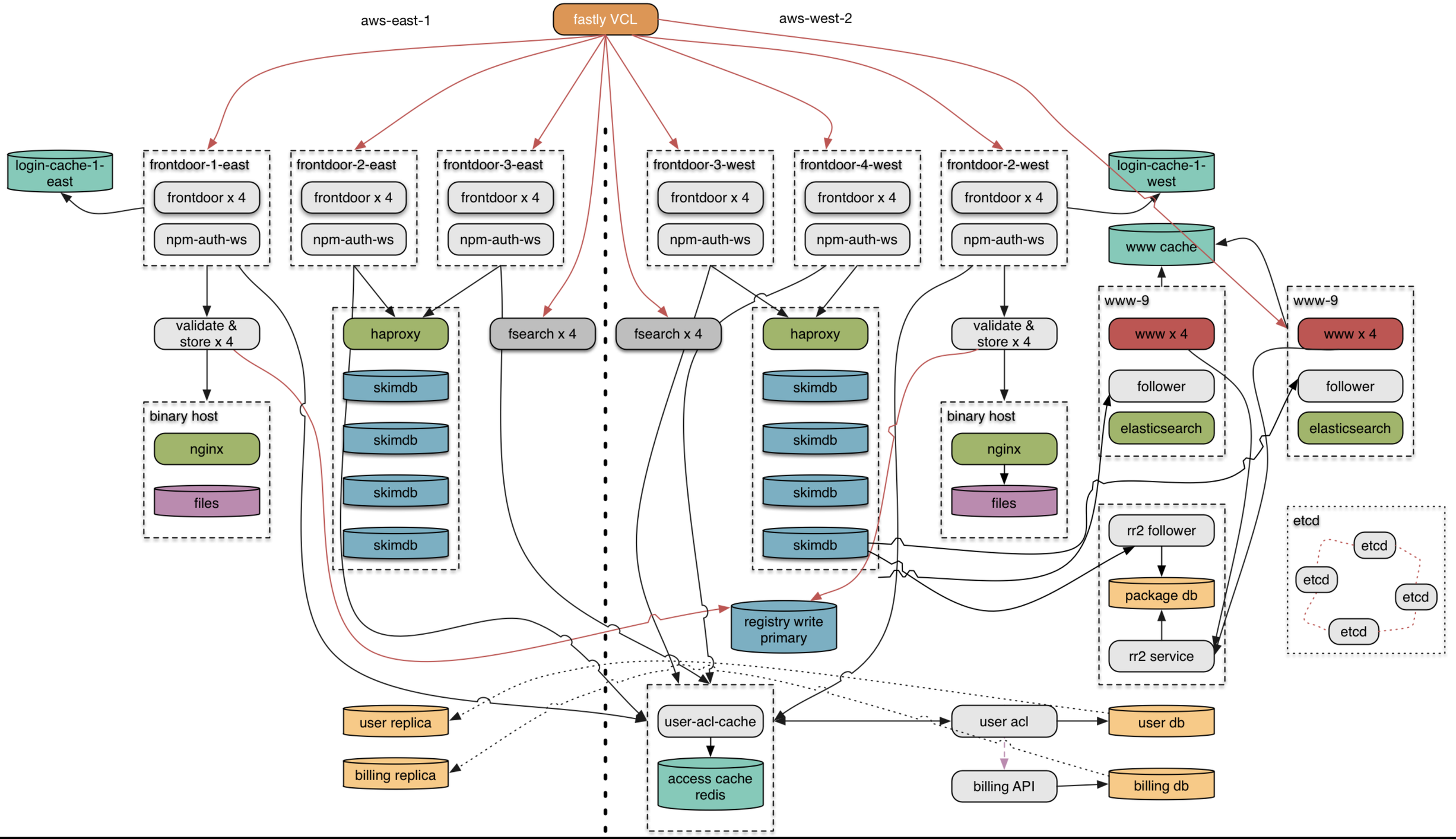
any box can be replaced

by running an ansible play

brace yourselves
diagrams incoming


fastly VCL
frontdoor
frontdoor
search
validate &
store
load-balanced
couchdbs
access control
tarballs
registry write
primary


aws-west-2
fastly VCL
aws-east-1
frontdoor
frontdoor
search
search
frontdoor
frontdoor
validate & store
load-balanced couchdbs
access control
load-balanced couchdbs
validate & store
tarballs
registry write primary
tarballs


aws-east-1
fastly VCL
aws-west-2
login-cache-1-east
frontdoor-1-east
frontdoor x 4
npm-auth-ws
frontdoor-2-east
frontdoor x 4
npm-auth-ws
frontdoor-3-east
frontdoor x 4
npm-auth-ws
frontdoor-3-west
frontdoor x 4
npm-auth-ws
frontdoor-4-west
frontdoor x 4
npm-auth-ws
frontdoor-2-west
frontdoor x 4
npm-auth-ws
login-cache-1-west
www cache
validate & store x 4
haproxy
fsearch x 4
fsearch x 4
haproxy
validate & store x 4
www-9
www x 4
follower
elasticsearch
www-9
www x 4
follower
elasticsearch
binary host
nginx
files
skimdb
skimdb
skimdb
skimdb
skimdb
skimdb
skimdb
skimdb
binary host
nginx
files
rr2 follower
package db
rr2 service
etcd
etcd
etcd
etcd
etcd
registry write primary
user replica
billing replica
user-acl-cache
access cache redis
user acl
billing API
user db
billing db

lots of complexity, but

- each piece has a well-defined responsibility
- each piece can be redundant
- exceptions: db write primaries
- each service can be worked on in isolation

downsides

— yay distributed systems
— pretty sure a message queue is in our future
— some single points of failure: db primaries
— metrics & log handling is poor
— everything is hand-rolled

conservatism won with node

- we're mostly on node 0.10.38
- memory leaks, some networking trouble with early iojs
- will try again with iojs 1.8.x
- or with node now that iojs took over :)

git deploy

This was a pain until we wrote a bunch of tools. Ansible to set it up once. Git to deploy. (Not the @mafintosh future!)

```
git push origin +master:deploy-production
git push origin +master:deploy-staging
```

Each interested host will report in Slack when it's done. You've deployed!

A git-deployable service

— haproxy load-balancing & monitoring
— webhooks server
— github webhooks trigger a bash script
— any server can have many apps git-deployed to it
— generally 1 process per core

open sourced parts

- jthooks: set up github web hooks from the command line
- jthoober: a server that listens for webhook pushes from github & runs scripts in response
- rderby: rolling restarts for servers behind haproxy
- renv: recursively manages json blobs with `etcd`.
- ndm: generate upstart/whatever scripts from a service.json config

metrics

All open-source. InfluxDB ➜ Grafana for dashboards.

- numbat-emitter - client to emit metrics from any node service
- numbat-collector - service to collect & redirect to many outputs

150,000 modules

~400GB tarballs

68 million dls/day peak

5800 req/sec peak

future work

- organizations for private modules! already in progress
- make web site search a lot better
- make the relational package data available via public api
- more public replication points (all public packages, including scoped)

npm loves you

```
npm install -g npm@latest
```