Scaling GitHub

Scaling Startups

Scaling GitHub

SCALING GITHUB

Scaling GitHub

Scaling GitHub

B======D~~~~

Scaling GitHub

Scalin' githubs

Scaling GitHub

GitHub

Scaling GitHub

githubs and shit

Scaling GitHub

Scaling GitHub

Scaling GitHub

Scaling github

Scaling GitHub

Scaling GitHub

Scaling GitHub

Scaling GitHub

Scaling GitHub

“Scale”

Two problems.

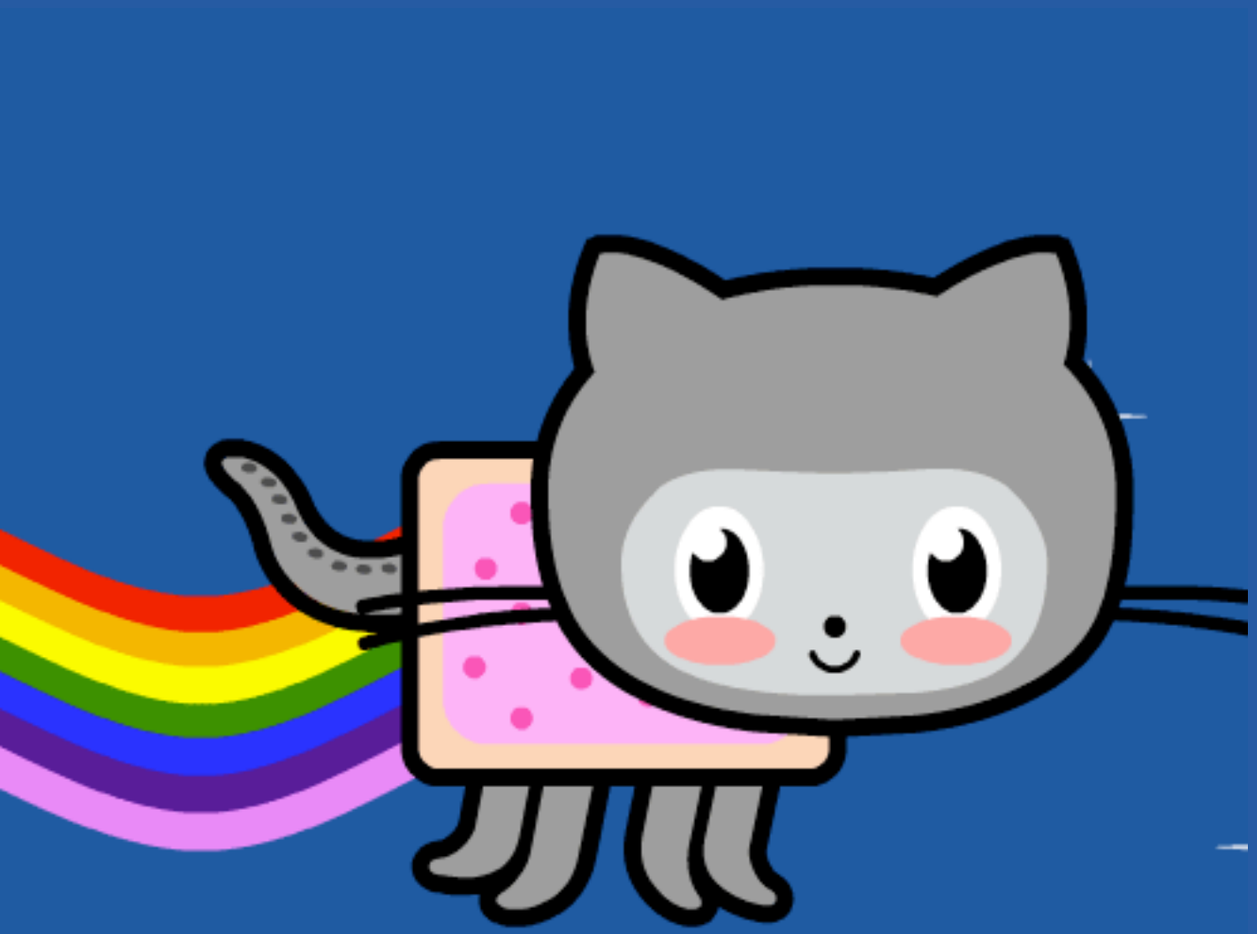
SyntaxError: compile error.

TECHNICAL ORGANIZATIONAL

I’m too hungover to work.

Scaling is people + technology


@holman



github

SOCIAL CODING

Organizational

jeez humans are so finicky

Happiness.

Happiness vs Productivity

happy employees are productive employees

productive employees

are happy employees

This isn’t a “management problem”.

Everyone needs to worry about this.

Hiring an employee is the most **TOXIC** thing you can do to your startup.

worse culture work slower

Hiring an employee is the most TOXIC thing you can do to your startup.

more bugs less features

better culture work faster

Hiring an employee is the most EXCITING thing you can do to your startup.

fewer bugs more features

so how can you score excitement and avoid the toxic?

yeah, i know...

TOXIC EXCITEMENT

would be a great name for a rock band

Keep your employees happy.
Really happy.

Worry about your coworkers.

Your servers, offices, and ideas are bullshit.

EMPLOYEES

Know your codebase

Know your process

Know your mistakes

Know your mission

Know your jokes

Know your priorities

NEW HIRES

Don’t know jack

Imprison your employees with happiness and nice things and cuddly work practices.

GitHub Jail

work whenever you want

work however you want

work on what you want

a product people love

four beers on tap

health, dental, vision

paid conference trips

retirement plans

solid salaries

stock

get out of the way

NO PLANNING SESSIONS

NO MEETINGS

NO NEED TO BE IN THE OFFICE

chat, pull requests, email

FASTER

MORE DIRECT

ALWAYS RECORDED

This is designed to retain people.

We're at 56 employees. We haven't lost one.

This is a huge, massive competitive advantage.

It justifies the extra expense.

Communication.

Don’t have the server guy who knows everything.

the billing girl

the testing dude

the performance czar

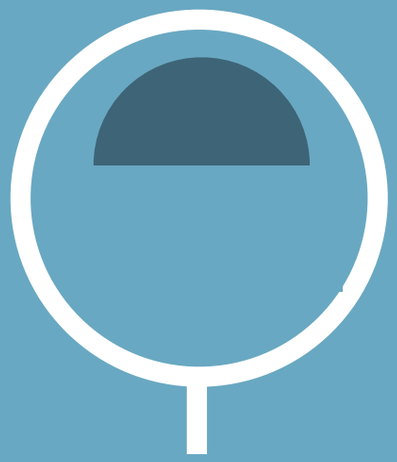
the customer support maven

the software licensing file hoarder

Don’t have the person who knows everything.

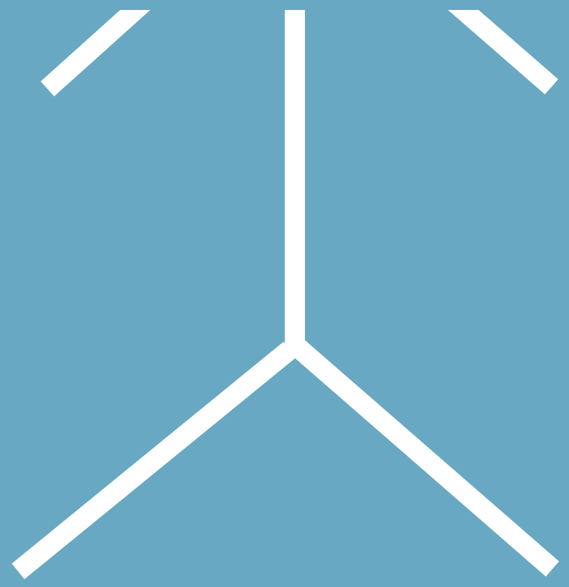
Specialization is great,
but only having one person
is a synchronous bottleneck.

Reduce institutional knowledge.



wikis
issues
chat logs
pull requests

Reduce institutional knowledge.



Every internal GitHub talk is automatically recorded, uploaded, and viewable to every future employee.

...on a Kinect-powered Arduino-based motion-detecting portable video recording platform.

Your new hire is stoked to dive in, start reading, and start contributing

...so don't get in their way.

Hire well.

Hiring poorly is just as bad as losing people.

Aim for really great people.

WE ♥ SELF-STARTERS

less babysitting, more code

(future!)

Keep your employees happy.

Really happy.

Don’t just market your product; market your team and company too.

Always think about attracting good people, even if you're not hiring.

OPEN SOURCE

CONFERENCES

TECHNICAL POSTS

SPONSORSHIPS

MEETUPS

TALKS

Technical

robots can be pretty finicky too

Automate.

hubot deploy github to production

COMPILATION

CoffeeScript
SCSS and SASS
bundles assets
caches Python dependencies
compiles Erlang changes
compiles C changes
builds static pages

APP SETUP

installs gems
symlink directories
14 rolling app server restarts

NOTIFY

Campfire
New Relic
graphite

deploys
current process overview
multi-server shell commands
new employee setup
app bootstrap

Automating now will save you way more time down the road.

Ship.

5x-30x

deploys per day

Ship early, ship often.

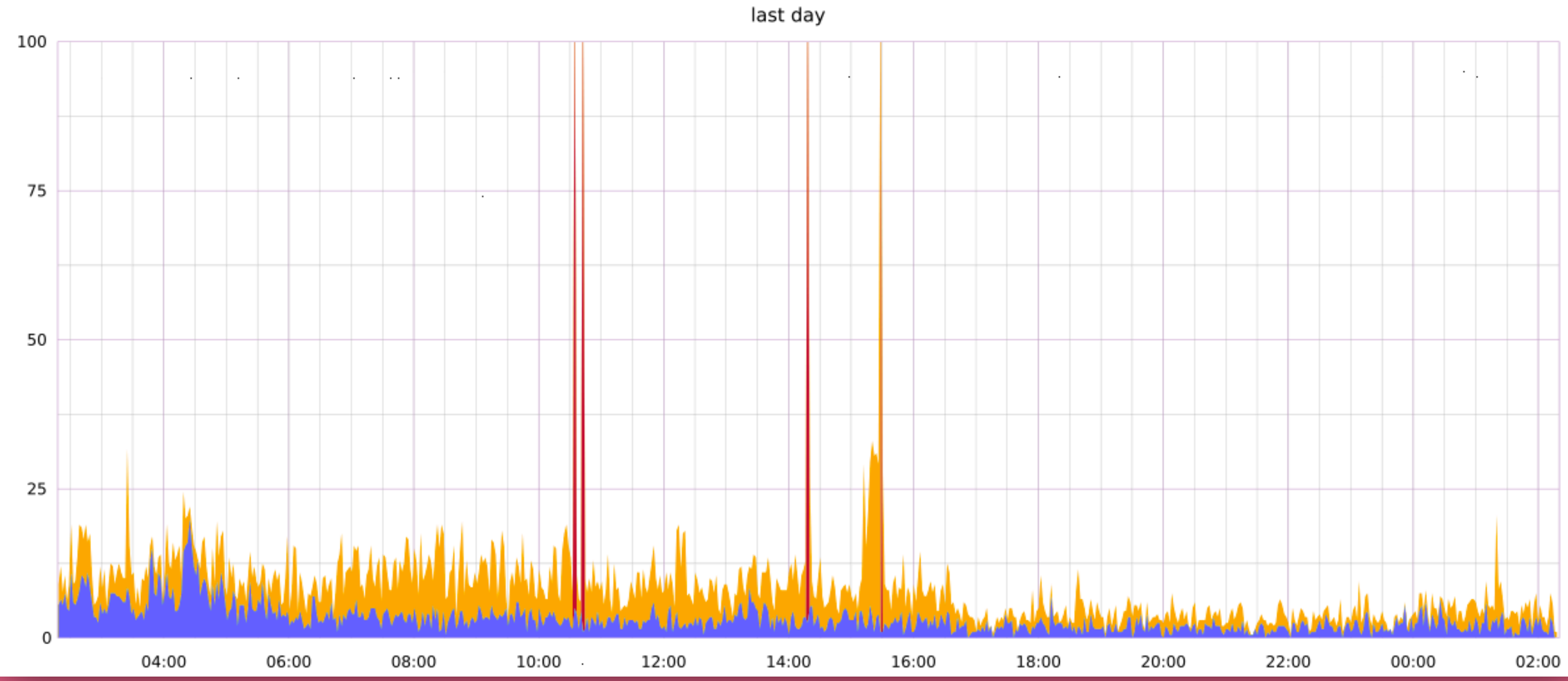
master = always deployable

always green tests

always a safe rollback

Limit your deployments

to staff-only
to beta users only
to one server only
to one app process on one server only



@github tweets

exceptions

deploys

deploys

Graph.

everyone loves fancy graphs

quickly see trends

quickly see problems

historical data as basis for alerts

METRICS ARE GREAT

But use them wisely.

162ms

average **overall** response time

Valueless metric.

59ms

average API response time
with 4x throughput of web

23ms

average raw response time
with 2x throughput of web

The responsiveness is a lie.

199ms

average **browser** response time

16,000

requests in the last week over 4.5s

Needed to look at the right stuff.

Browser Requests

Top-level page loads by logged in users in real web browsers. Also, pjax.

Week

199 ms

average response time

337,035 requests over **987** ms

16,852 requests over **4.49** s

	Response Time	Throughput	CPU Burn
Browser	**199** ms	**1,671** rpm	**5.6** cpus
Public	**103** ms	**1,994** rpm	**3.4** cpus
Ajax	**43** ms	**2,345** rpm	**1.7** cpus
API	**59** ms	**4,490** rpm	**4.4** cpus
Googlebot	**162** ms	**322** rpm	**0.9** cpus
Raw	**23** ms	**3,612** rpm	**1.4** cpus
Feed	**51** ms	**1,105** rpm	**0.9** cpus
Other	**83** ms	**1,254** rpm	**1.6** cpus
		16,795 rpm	**20.0** cpus

googlebot had

2-3x throughput

3-4x CPU usage

compared to web requests

Collect a lot of metrics, but make sure they're important metrics.

GitHub scale.

Everyone has different growth patterns.

GitHub has had three.

major github infrastructure milestones

Launch
2008

Bare metal servers
2009

net-shard
2010

Launch
2008

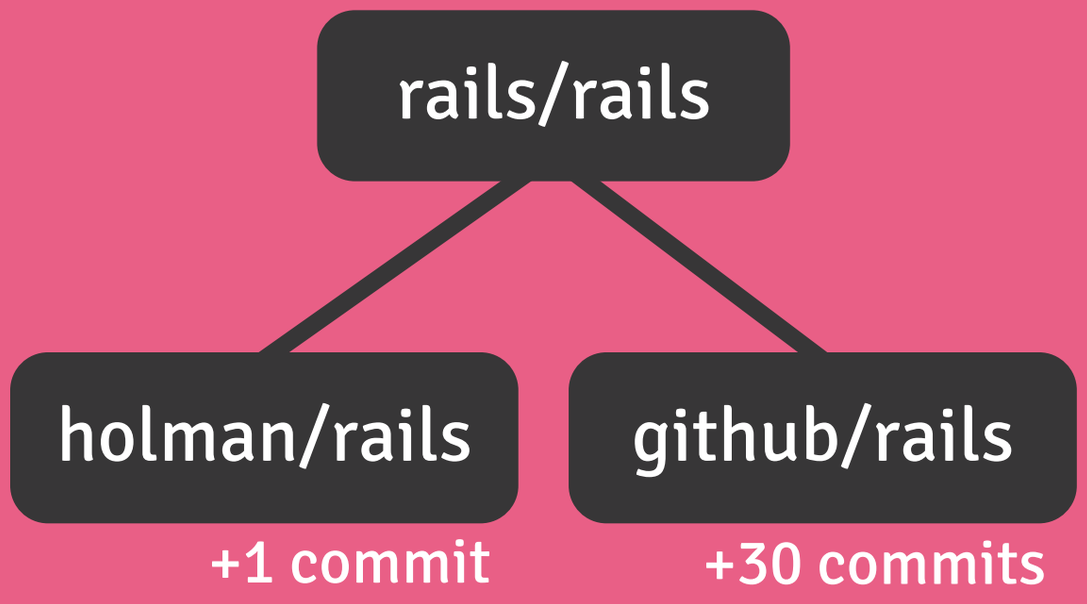
Hosted on Engine Yard
10 VMs
54GB RAM
shared GFS mount
one metric shit-ton of caching

Bare metal servers
2009

Hosted on Rackspace
16 bare metal servers
288GB of RAM
redundant disk storage

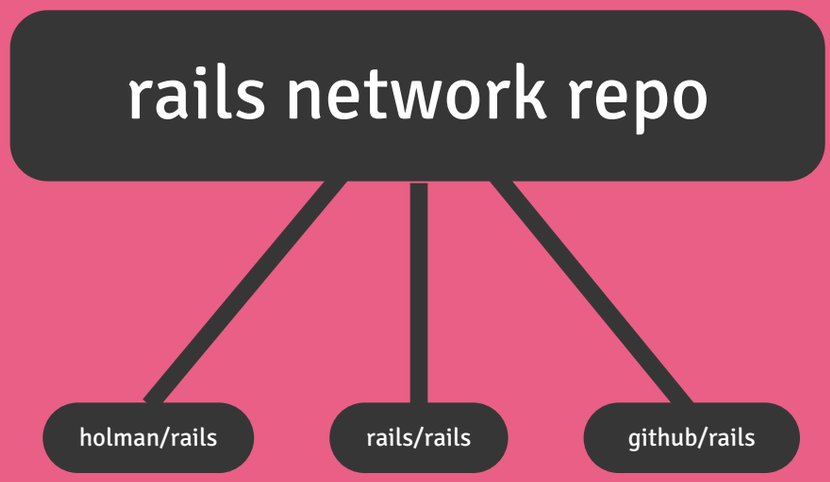
net-shard
2010

networks share a common repository



...multiplied 2,600 times

classic



fat network, skeleton forks

net-shard

net-shard
2010

networks share a common repository
they also share the same fs and partition
halves storage requirements
improves hit rate of kernel disk cache
speeds up backups
allows fast forks, merge button, network GC

For GitHub, scaling involved a lot of predictions of future trends, then acting appropriately.

Side Projects.

A THOUGHT EXPERIMENT:

Imagine I told you to build...

This grew organically, over dozens of projects, written by dozens of employees, when they felt like it.

Figure out how to let this happen. It’s hard.

Small hack days can result in real, imma-make-us-money impact.

Small hack days can also keep your developers insanely happy.

Small hack days can also lead to learning new techniques.

Projects and Posts.

CHAT ROOM ROBOT

github.com/github/hubot

JENKINS + CAMPFIRE

github.com/github/janky

OFFICE MUSIC DJ

github.com/holman/play

BLOG: GITHUB IS MOVING TO RACKSPACE

git.io/jByrlQ

BLOG: HOW WE MADE GITHUB FAST

git.io/p5v2Ag

BLOG: UNICORN

git.io/77Onfg

Technical

+ Organizational

Continually refine your process + workflow.

Worry about your computers, and worry about your humans.

Thanks.

ZACH HOLMAN

zachholman.com/talks

twitter+github: @holman