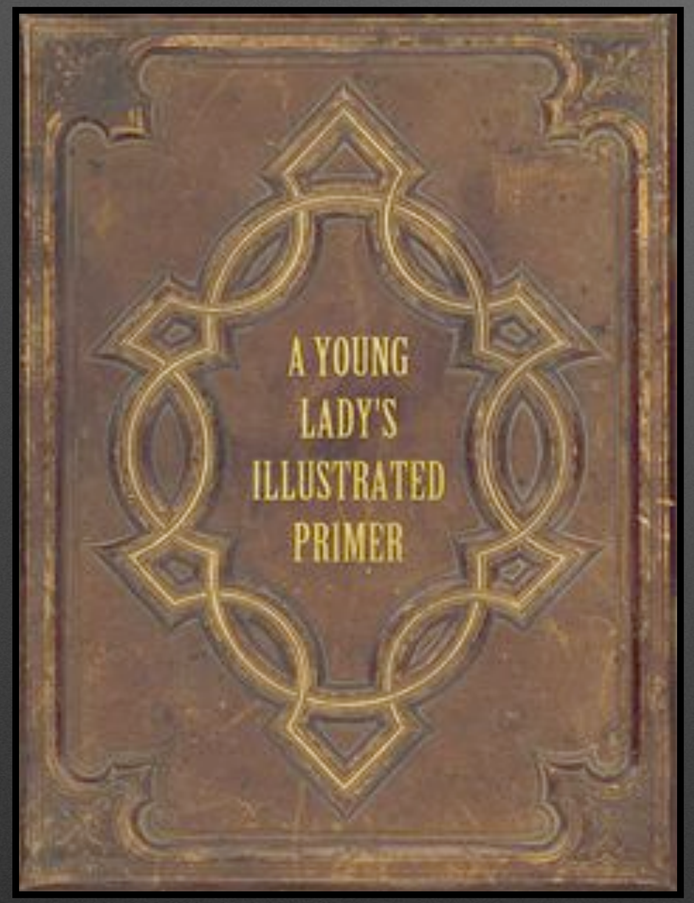
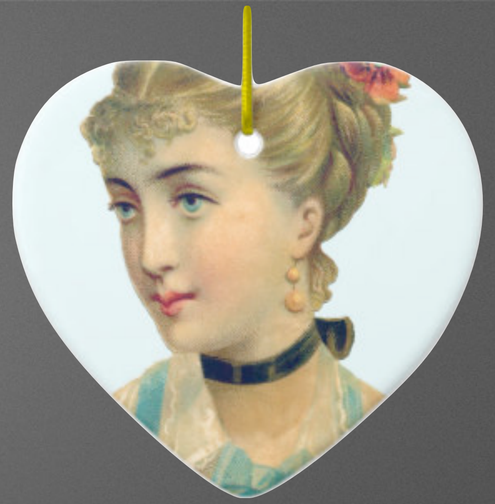
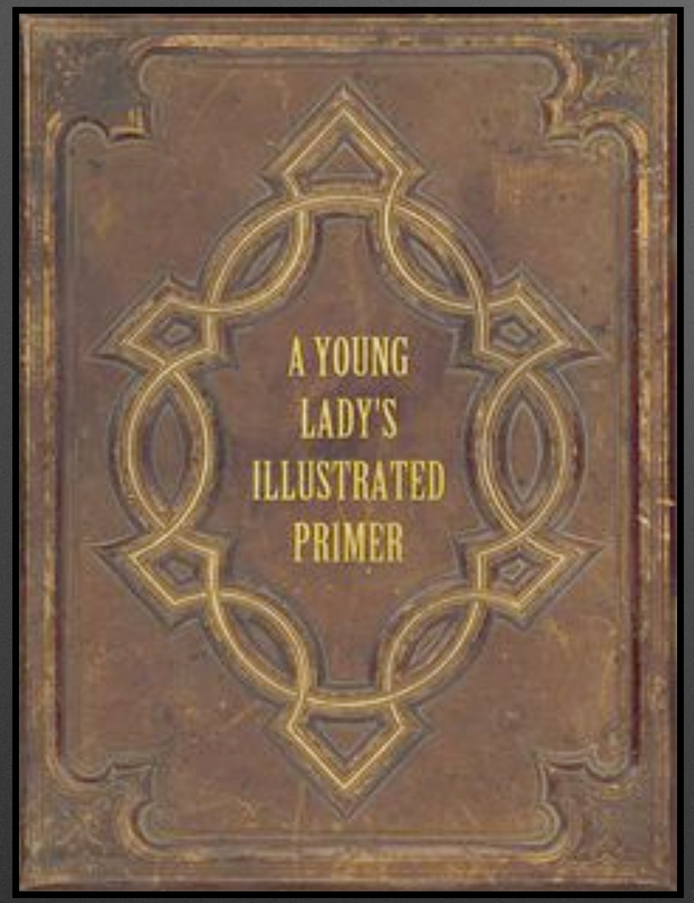
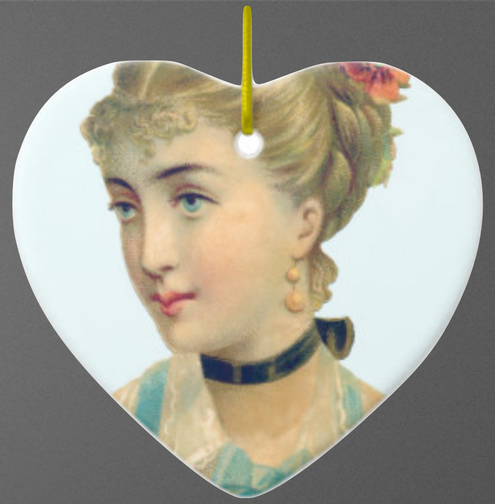




A Young Lady's Illustrated Primer to Architecture and Technical Decision-Making

Charity Majors, CTO, hound.sh
@mipsytipsy





A Young Lady's Illustrated Primer to Architecture and Technical Decision-Making

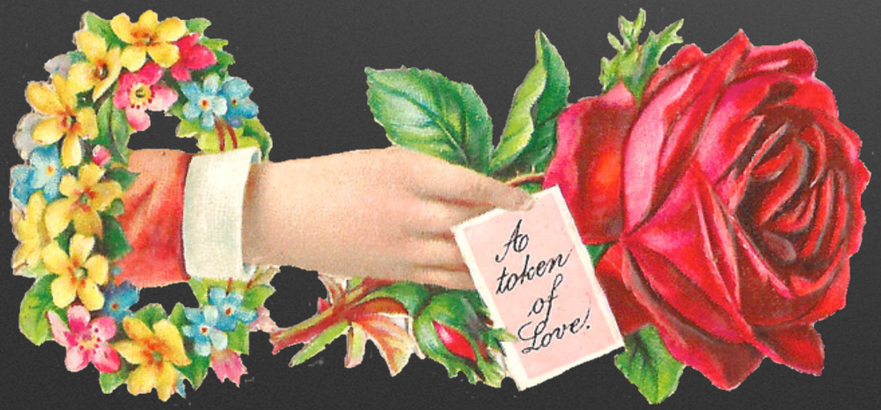
Charity Majors, CTO, hound.sh
@mipsytipsy


SREcon16
04.07.16-
04.08.16
SANTA CLARA, CA
usenix

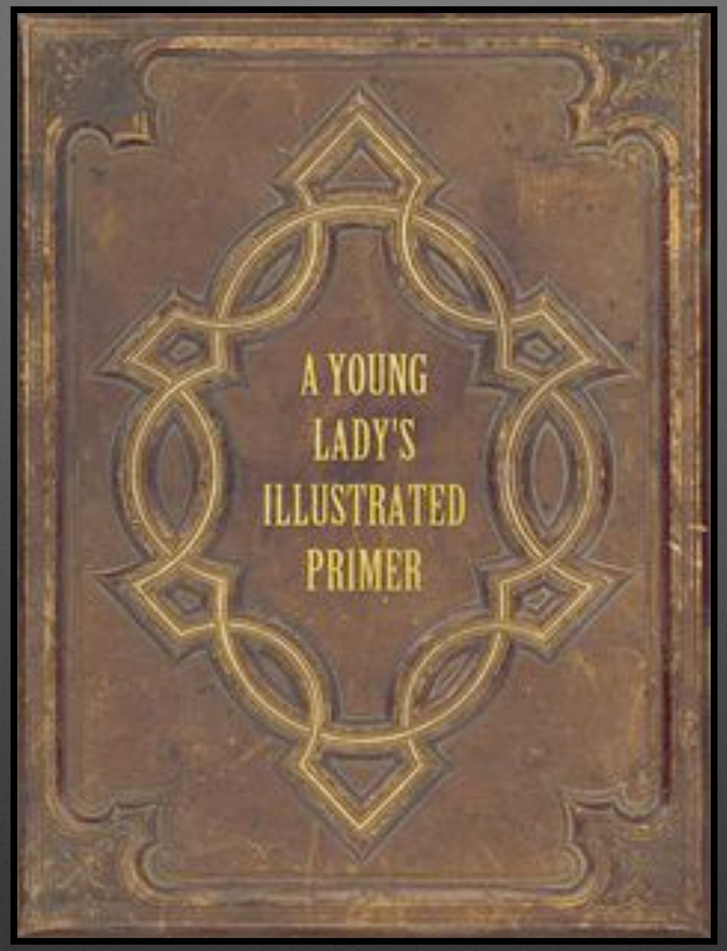
SREcon16

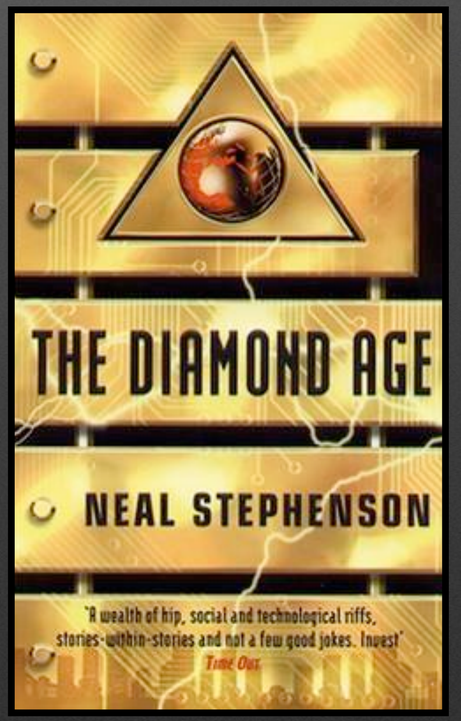
04.07.16-
04.08.16
SANTA CLARA, CA

usenix






A YOUNG
LADY'S
ILLUSTRATED
PRIMER


THE DIAMOND AGE
NEAL STEPHENSON
'A wealth of hip, social and technological riffs, stories-within-stories and not a few good jokes. Invest'
Time Out



Software is eating the world

and your brain, probably.

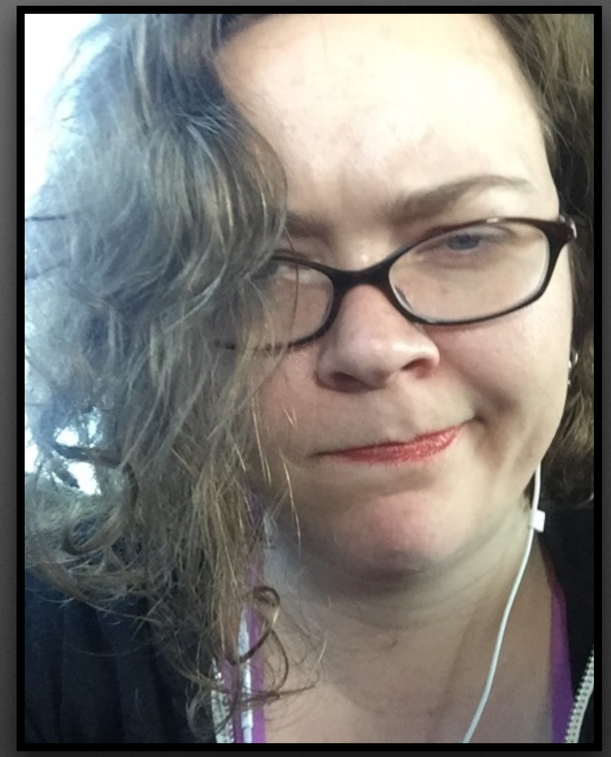
Making better choices with software.



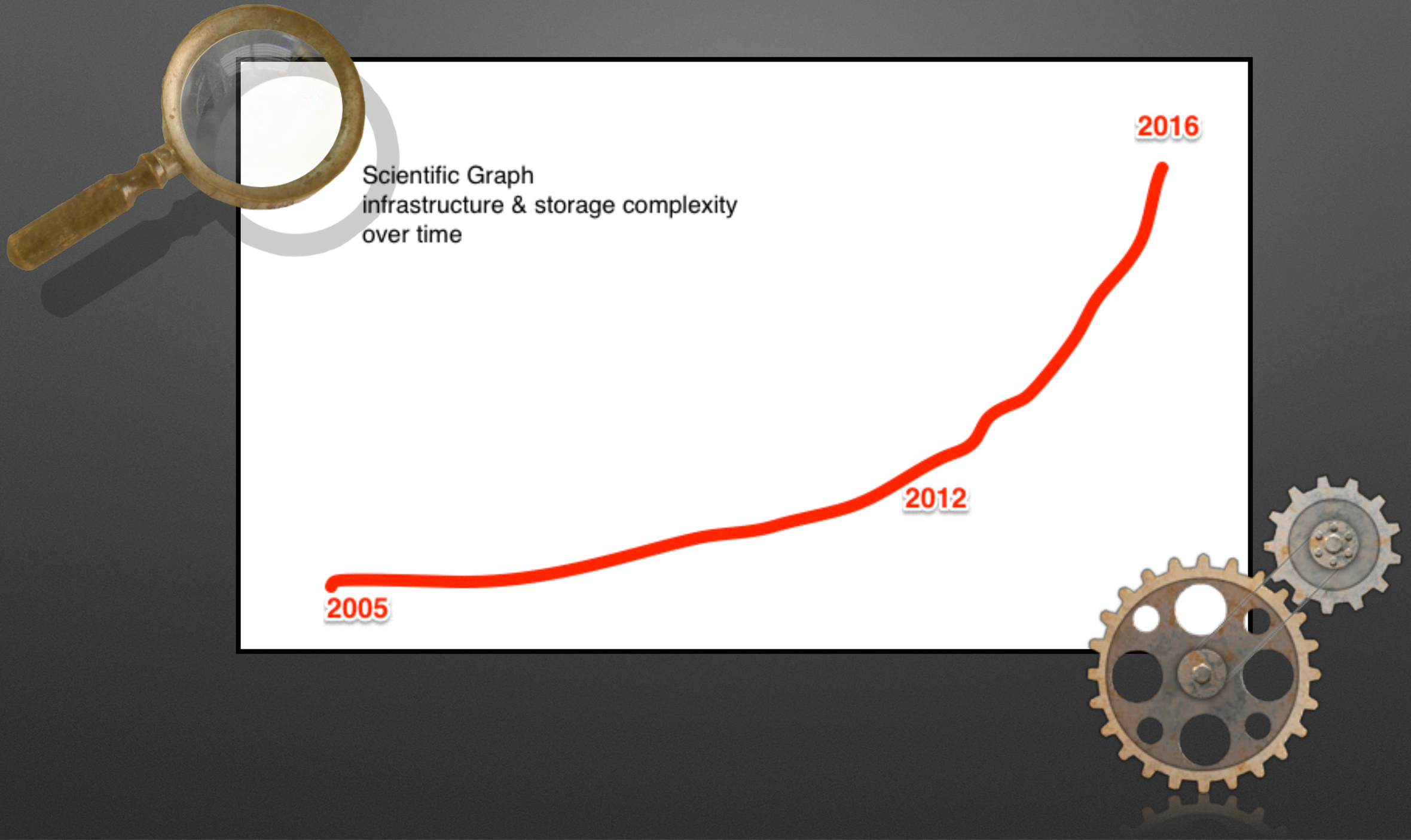


shit just got personal

i would like not to screw up my startup!



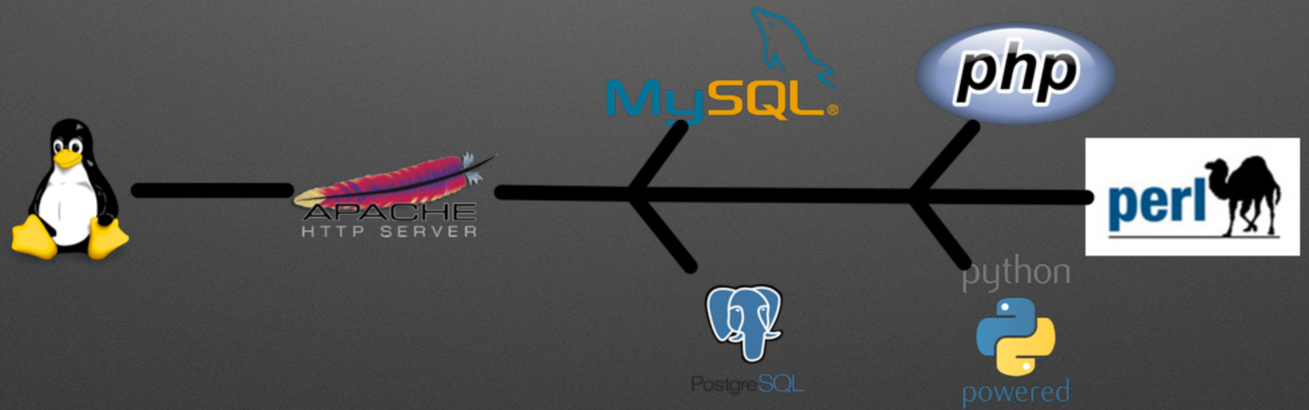



Scientific Graph
infrastructure & storage complexity
over time
2016
2012
2005



“u can haz LAMP stack”





2000



“Would you care to come over and discuss service oriented architectures and REST APIs? Over tea, perhaps.”

“Splendid! And have you heard of this ‘NoSQL’ oddity?”



“I am sure tis but a passing fad.”

2005

Recipe

Infrastructure

Ingredients

Instructions

- Virtualization
- Config management
- RDBMS
- Services with REST APIs
- Continuous integration/delivery
- Agile
- “DevOps”





2011-2013

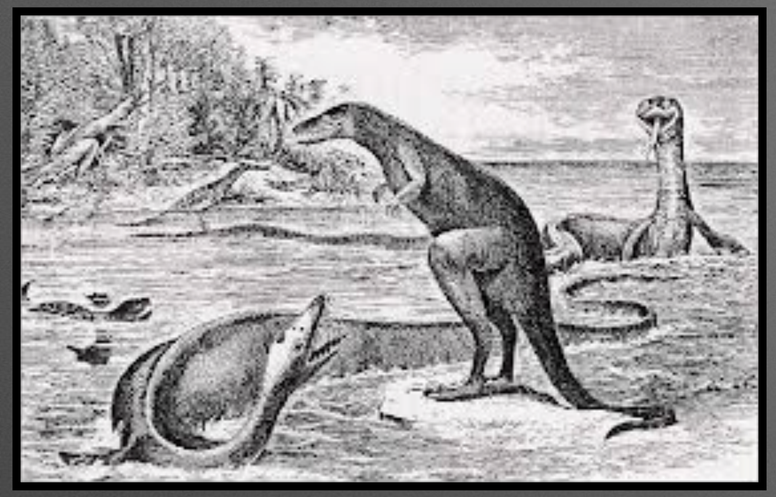

CoreOS
Google Compute Engine
mongoDB
etcd
Spark
⟨8K⟩ data
ANSIBLE
KNIME
docker
druid
amazon web services™
cassandra
EC2
Parse
CONSUL
RethinkDB
RabbitMQ™
Microsoft Azure
heroku
neo4j
VAGRANT
kubernetes
OrientDB
redis
openstack
CLOUD SOFTWARE
samza
kafka
MESOS
Chef
hadoop
2015: welcome to the jungle

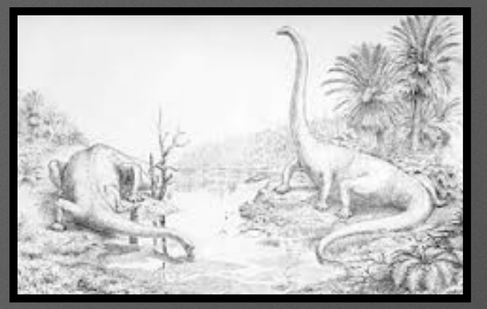


Accelerating trends in 2016

Polyglot storage
Composable infrastructure
Containerization, microservices
Coupling platforms (*aaS)
Storing more and more data forever







Cambrian Explosion of technical complexity





about 30 min from slide 15 to the end was 25 min from start to here, shoot for 15



the paradox of choice

this is actually really hard on humans.


LIFE IS PAIN, HIGHNESS.
ANYONE WHO TELLS YOU DIFFERENTLY IS SELLING SOMETHING.



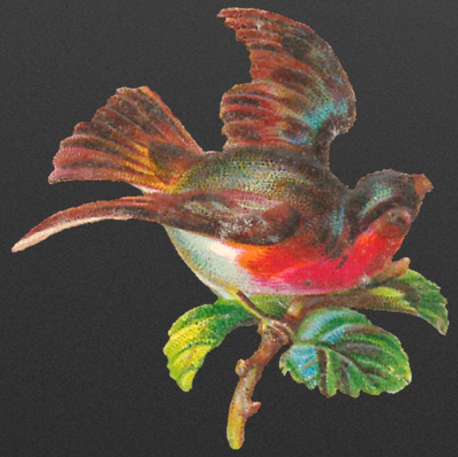
A few tips for making reasonable technical decisions

your best engineers already bring these intuitions to the table

Technology serves the mission



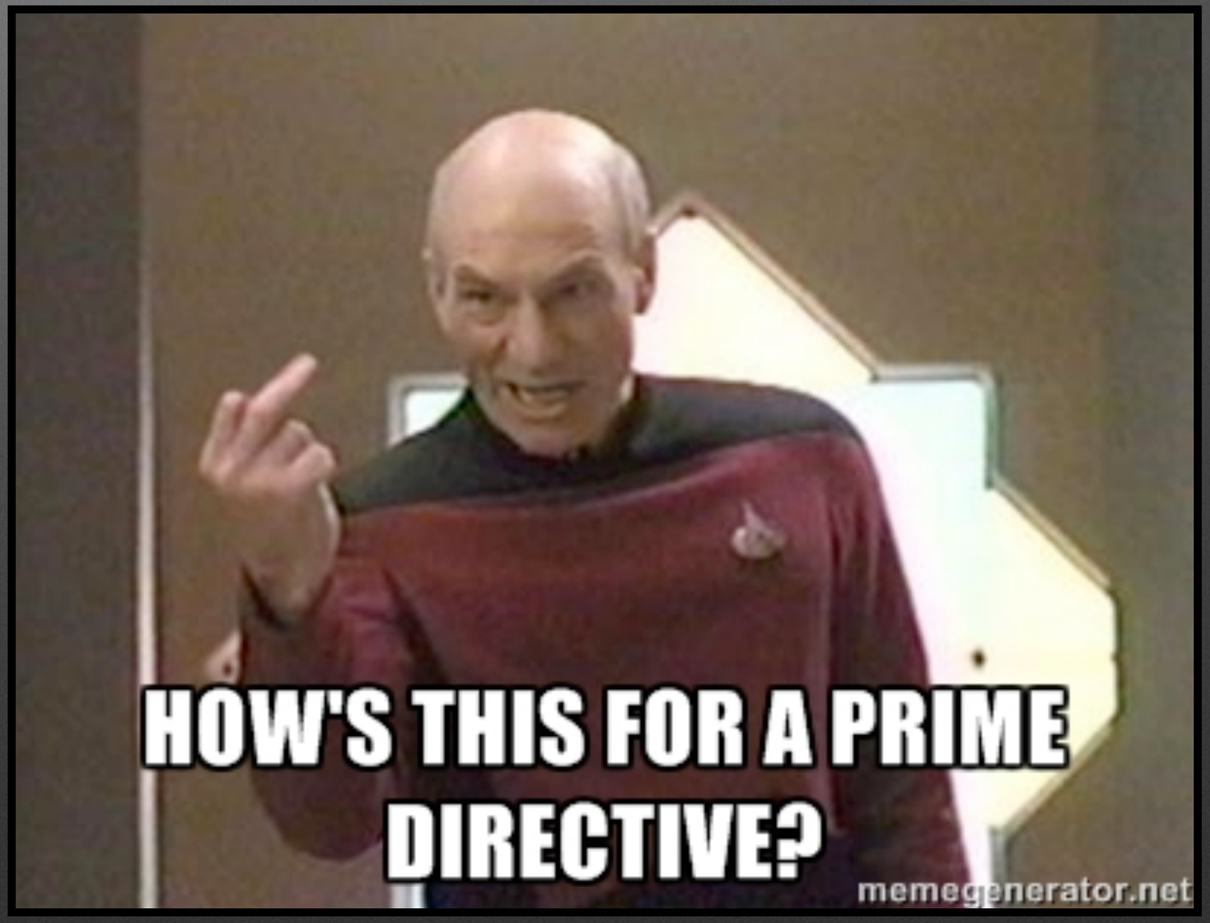




Software is the enemy



- Every piece of software adds fragility and points of failure
- Everything you write, someone will have to debug and maintain
- It is easy to add software to your stack, and hard to remove





Reduce, reuse, recycle.



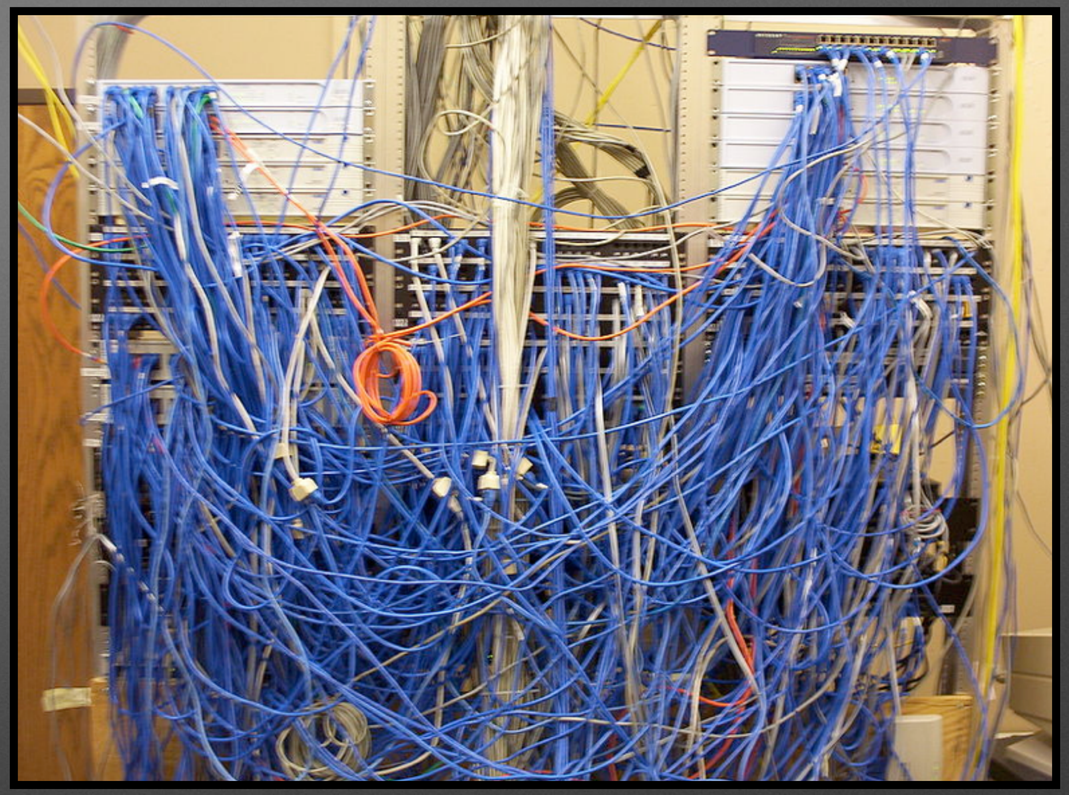




Can you solve the problem with your existing tools and solutions?



Optimize globally, not locally



If you pick the perfect language/storage solution for every local problem, you will have an unmanageable mess.



Have SOME process for gating major new components into production.



- Tech leaders: this is YOUR JOB.
- What is the relative gain?
- Everyone should know the process.
- The process should add some degree of friction
- Don’t micromanage outside the critical path



Good reasons for choosing new components:



- It is critical to a new feature which is critical to the company mission
- You need to replace an old crappy component — but define a timeline and get rid of the old one
- You and your team already have expertise in it



BAD reasons for choosing new components:



- You think it's cool technology and looks like fun
- The vendor benchmarks look fantastic!
- Hacker News says something is good and/or bad (and/or anything at all)

Choose boring technology





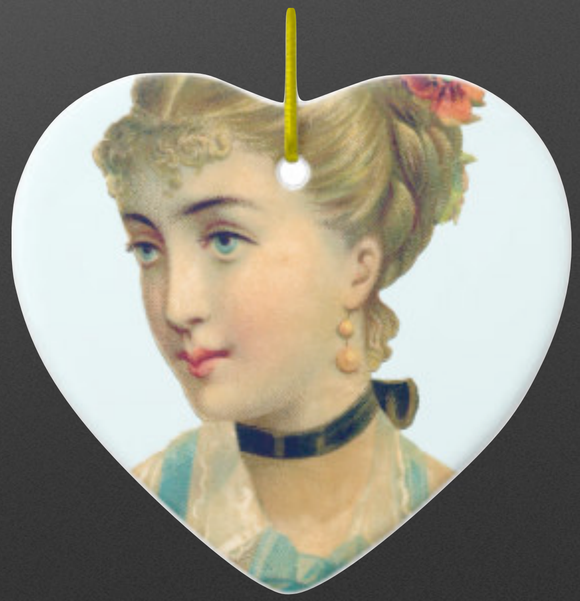


h/t @mcfunley, http://mcfunley.com/choose-boring-technology



Boring does not mean bad!

- Failure modes are well understood
- Rich library support for languages
- For databases, extensive production hardening
- Tooling and support for debugging problems
- Robust user base



Understand your appetite for risk

- Early startups have massively greater tolerance for risk.
- Use that risk! But spend it on your core differentiators.



If you decide to go bleeding-edge:





- Is it key to your mission?
- How confident are you the software will be around in 1-2 years?
- Are you willing to be a maintainer or committer?

It is totally legitimate to make software decisions that are influenced by the quality of the community.





Are they friendly and welcoming? Do they have a code of conduct, do they deal with assholes effectively? Do they value new contributors or are they tribal and clubby?

Operational Impact



The more mature your company becomes, the more your technical choices must be driven by operational impact.



Corollary: make as many ops problems as possible not your problem.







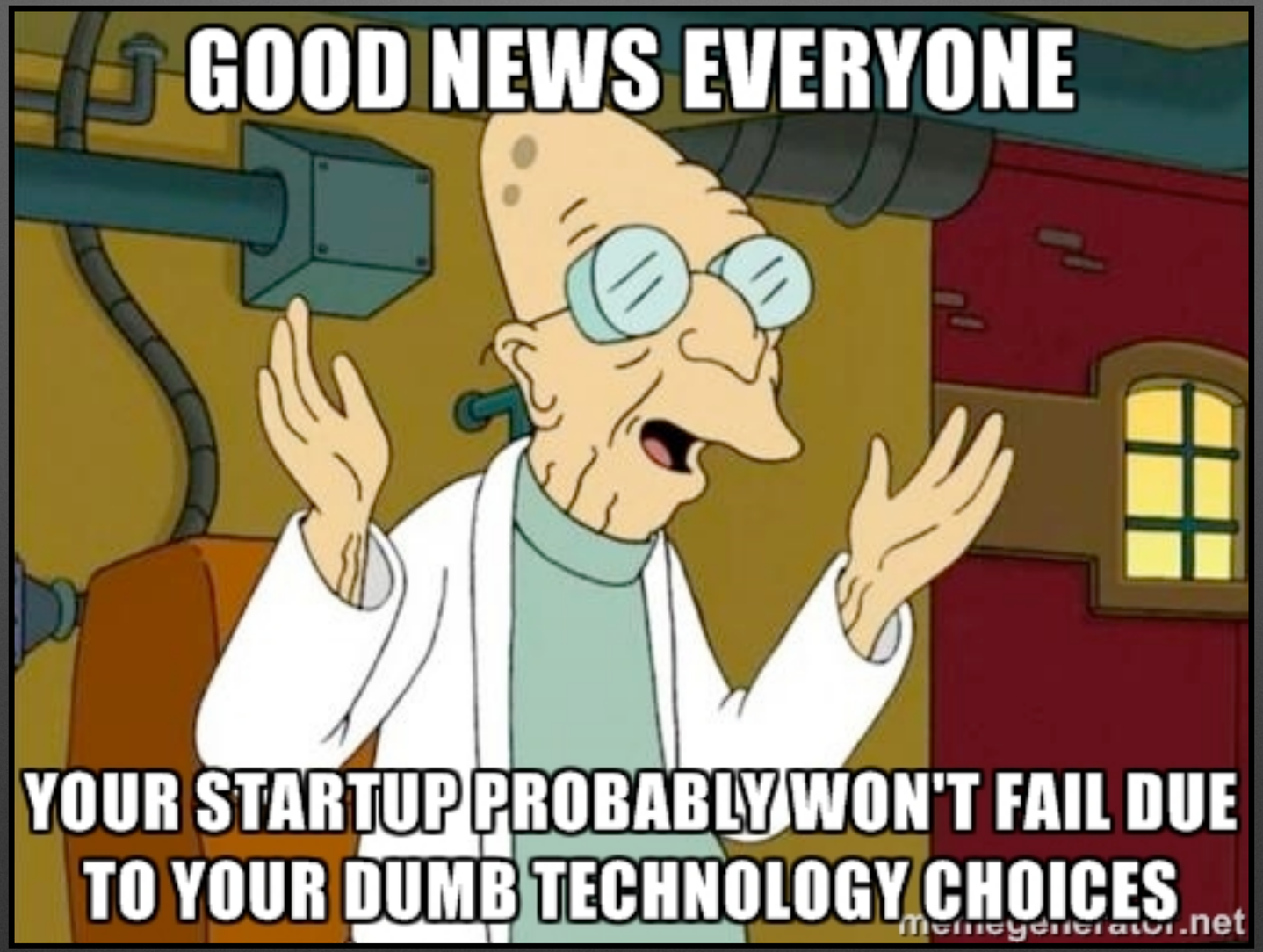
celebrate the engineers who remove code, deprecate, and refactor, as much as those who add features.










GOOD NEWS EVERYONE
YOUR STARTUP PROBABLY WON'T FAIL DUE TO YOUR DUMB TECHNOLOGY CHOICES
memegenerator.net

Engineers: know your power.



Ask before you join:

- How many languages do you use?
- What is your process for adding a new component?
- How do you make technical decisions?



Manifesto



- Technology serves the mission, not vice versa
- Reuse solutions. Resist software sprawl.
- Have an established process for adding major components.
- Choose boring technology, whenever you can.
- Understand your appetite for risk and exploit that when you can.
- The longer you survive, the more operational impact trumps all.

“I can’t shake the feeling that 15 years ago, someone was making this same presentation, except they were saying “We used to only have to worry about C++, but now there’s Perl, and Python, and PHP, and Java, wtf do we do in this crazy world?!? YOU MEAN I HAVE TO CHOOSE MY RDBMS THERE IS MORE THAN ORACLE v8 WTF DO I DO?”"

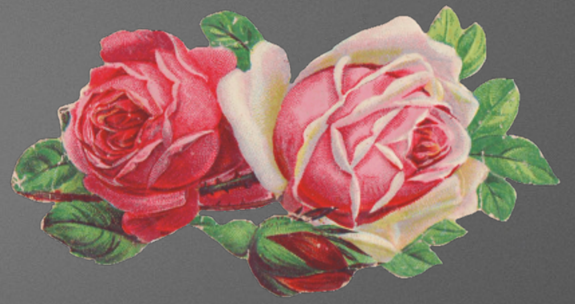


@ferlatte



remember, it’s only gonna get worse <3





thanks kids!
enjoy #SRECon16!

@mipsytipsy