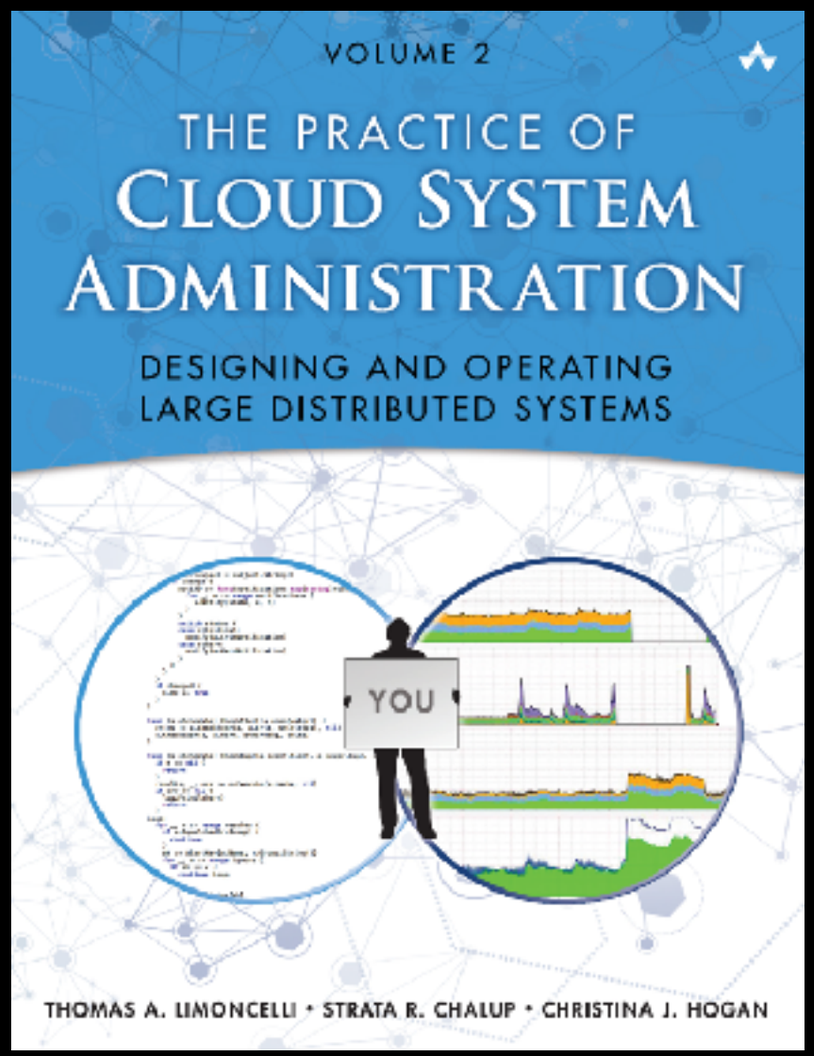
Radical Ideas Enterprises Can Learn From The Cloud



Tom Limoncelli, SRE

StackOverflow.com

the-cloud-book.com

@YesThatTom

www.informit.com/TPOSA
Discount code TPOSA35

Who is Tom Limoncelli?

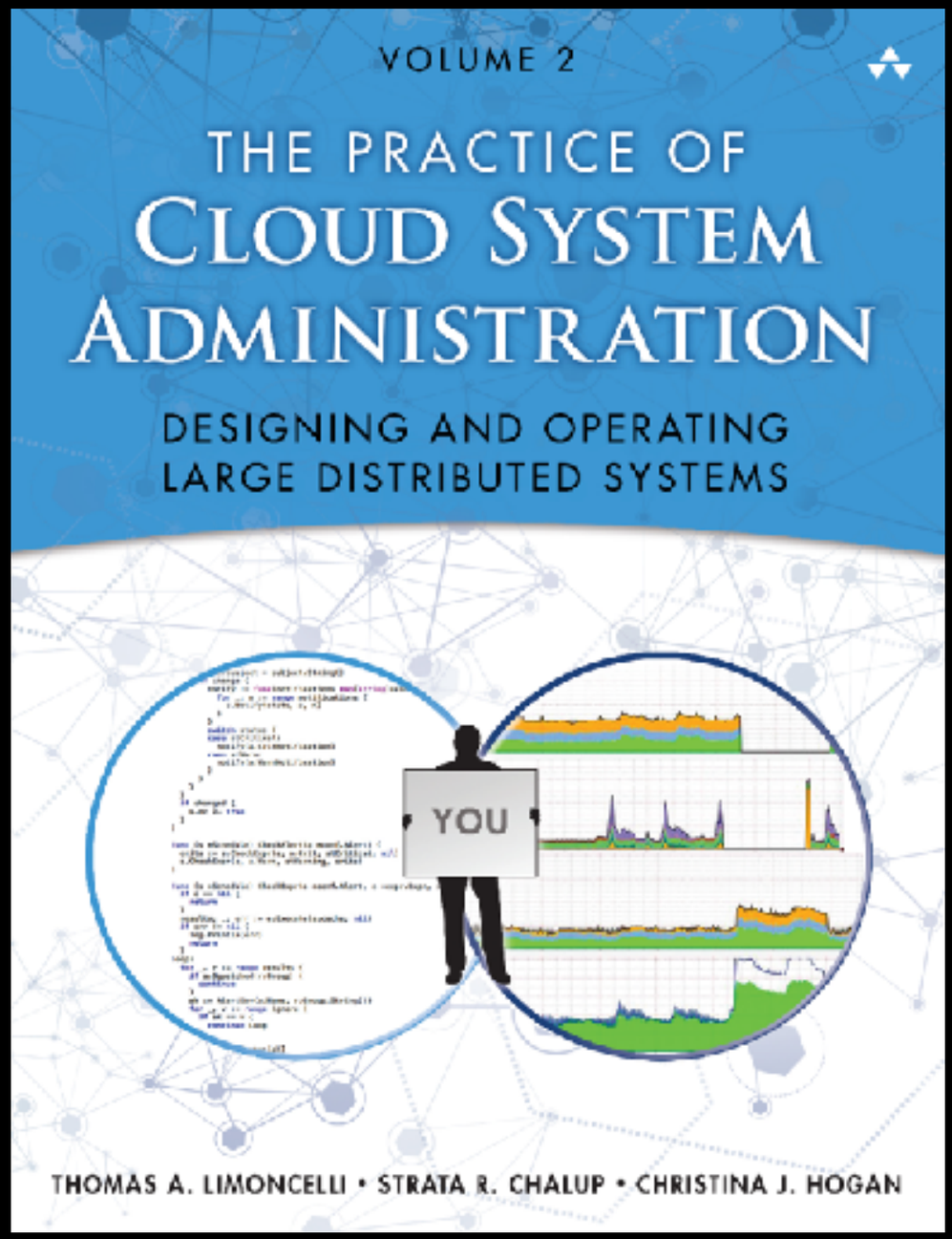
Sysadmin since 1988

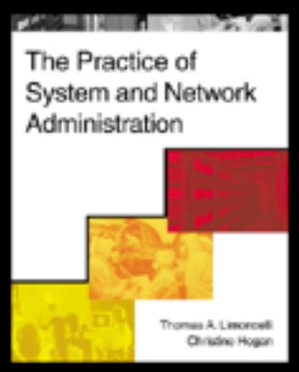
Worked at Google, AT&T/Bell Labs and many more.

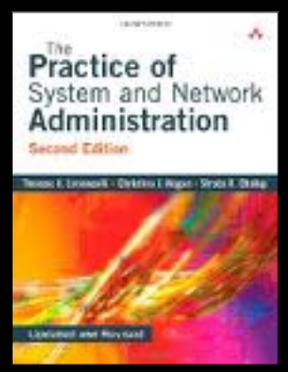
SRE at Stack Overflow, Inc http://careers.stackoverflow.com

Blog: **EverythingSysadmin.com**

Twitter: **@YesThatTom**







VOLUME 2

THE PRACTICE OF CLOUD SYSTEM ADMINISTRATION

DESIGNING AND OPERATING LARGE DISTRIBUTED SYSTEMS

THOMAS A. LIMONCELLI • STRATA R. CHALUP • CHRISTINA J. HOGAN

The Cloud

The Cloud

The

Clooooooouud

The Cloud!!!!!!

The Cloud!!1!

We

<heart>

The Cloud

The cloud solves all problems.

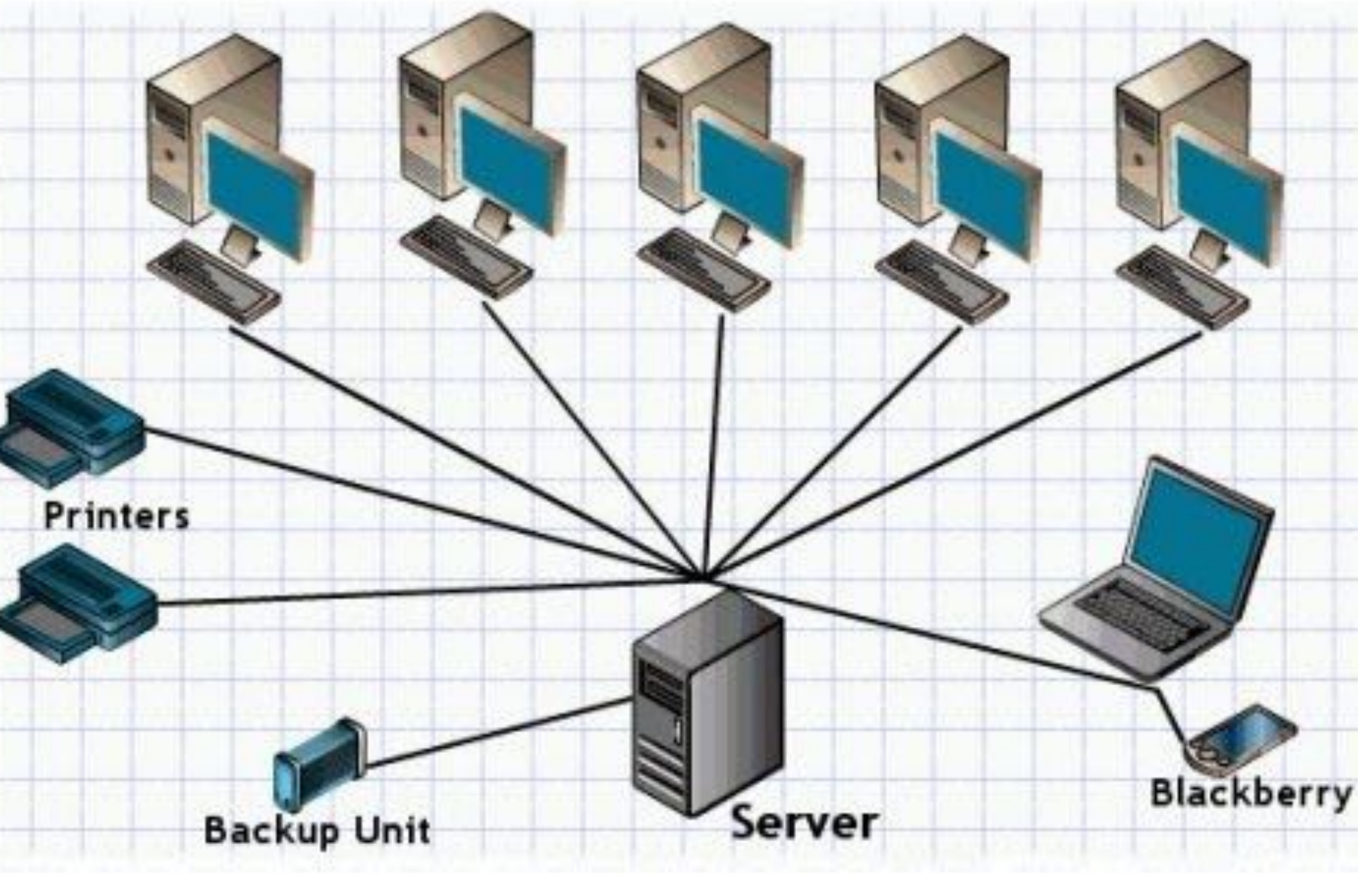
Cloud cloud cloud cloud cloud cloud cloud cloud
cloud cloud cloud cloud cloud cloud cloud cloud
cloud cloud cloud cloud cloud cloud cloud cloud
cloud cloud cloud cloud cloud cloud cloud cloud
cloud cloud cloud cloud cloud cloud cloud cloud
cloud cloud cloud cloud cloud cloud cloud cloud
cloud cloud cloud cloud cloud cloud cloud cloud
cloud cloud cloud cloud cloud cloud cloud cloud
cloud cloud cloud cloud cloud cloud cloud cloud
cloud cloud cloud cloud cloud cloud cloud cloud
cloud cloud cloud cloud cloud cloud cloud cloud
cloud cloud cloud cloud cloud cloud cloud cloud.

Distributed Computing

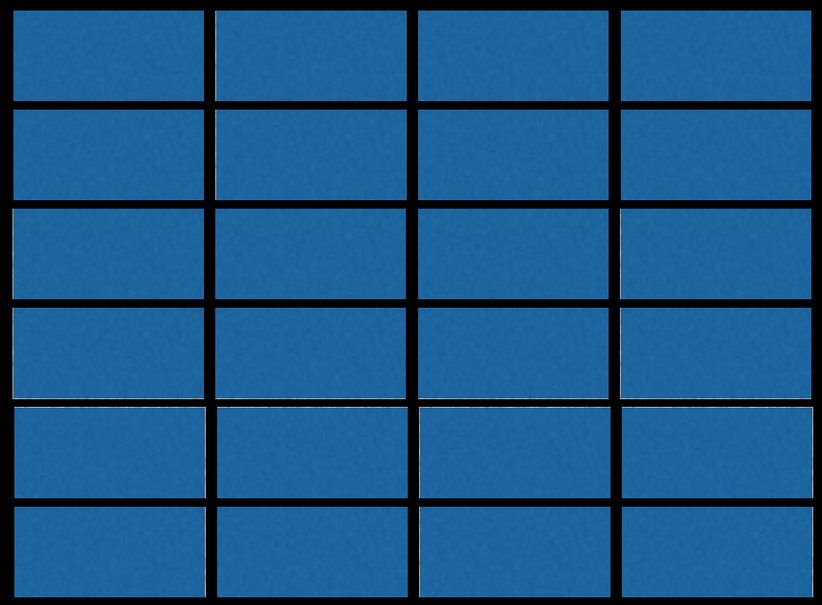
COURTESY NATIONAL MUSEUM OF COMPUTING/EXPRESS&STAR


Master Menu
Date: 12-11-1990
Time: 16:21:10
Page 1 of 2
Version 1.00
1. Filing Assistant
2. Reporting Assistant
3. Professional Finance Program
4. VP Planner
5. DOS Commands
6. Managing Your Money
7. Dr Halo
8. Word Perfect
9. Printmaster
↑ or ↓ point to option
ENTER select option
F1 Help
F7 Quit
F9 Menu Maintenance
PgDn Next page
IBM

Client Server Network



Distributed Computing



- Divide work among many machines
- Coordinated central or decentralized
- Examples:
  - Genomics: 100s machines working on a dataset
  - Web Service: 10 machines each taking 1/10th of the web traffic for StackOverflow.com
  - Storage: xx,000 machines holding all of Gmail's messages

Distributed computing can
do more “work” than the
largest single computer.

More storage.
More computing power.
More memory.
More throughput.

Mo’ computers, Mo’ problems

Thousands of Users

- Bigger risks
- Failures more visible
- Automation mandatory
- Cost containment becomes critical

In response: Radical ideas on

- Reducing risk / Improve safety
- Reliability becomes a competitive differentiator
- New automation paradigms
- Cost and economics

Make peace with failure

Parts are imperfect

Networks are imperfect

Systems are imperfect

Code is imperfect

People are imperfect

Learn how to

FAIL

BETTER

3 ways to fail better

1. Use cheaper, less reliable, hardware.
2. If a process/procedure is risky, do it a lot.
3. Don’t punish people for outages.

Fail Better Part 1 of 3:

Use cheaper, less reliable, hardware.

- Loss-damage waiver
- Liability
- Personal accident insurance
- Personal effects coverage

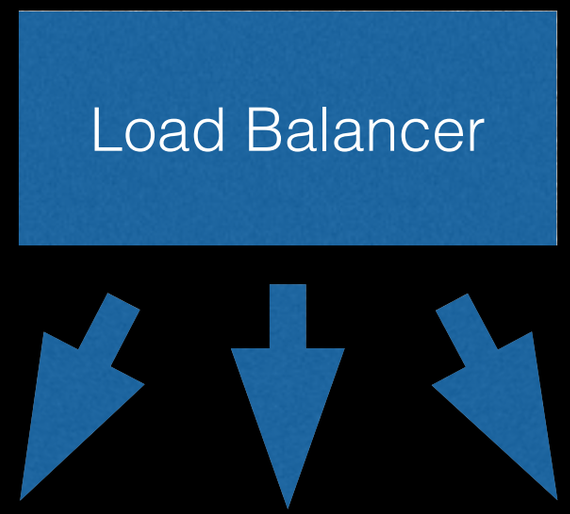
High-End Server

RAID

Dual PS

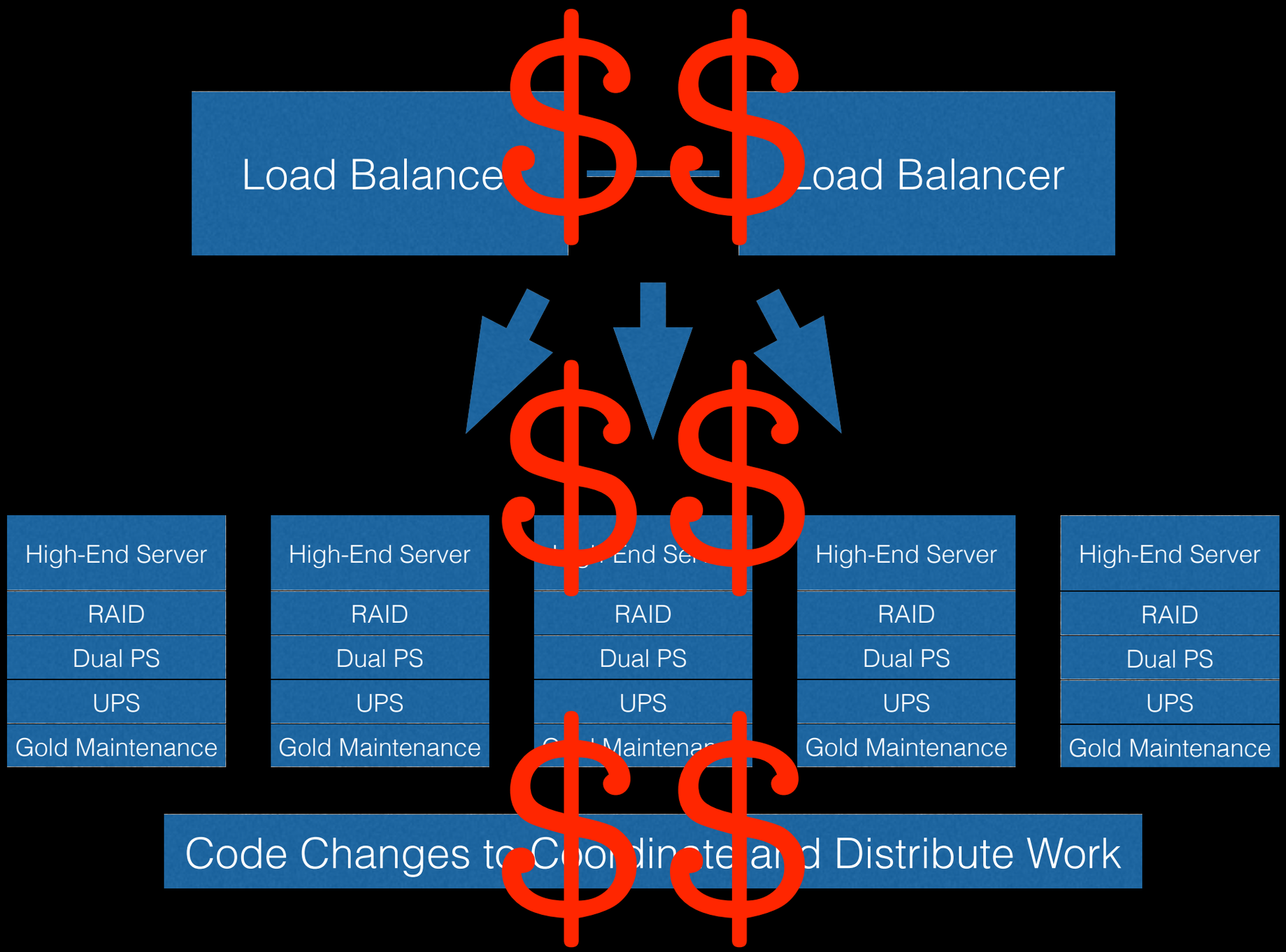
UPS

Gold Maintenance



High-End Server	High-End Server	High-End Server	High-End Server	High-End Server
RAID	RAID	RAID	RAID	RAID
Dual PS	Dual PS	Dual PS	Dual PS	Dual PS
UPS	UPS	UPS	UPS	UPS
Gold Maintenance	Gold Maintenance	Gold Maintenance	Gold Maintenance	Gold Maintenance

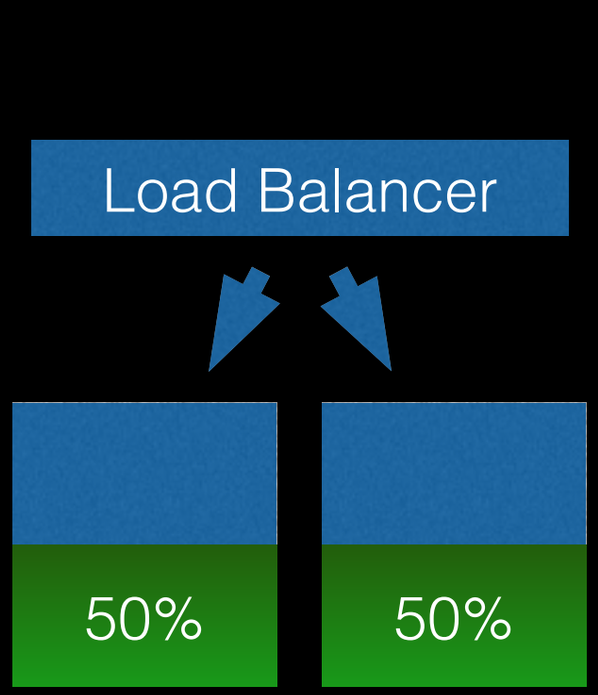
Code Changes to Coordinate and Distribute Work


Load Balance
Load Balancer
High-End Server
RAID
Dual PS
UPS
Gold Maintenance
High-End Server
RAID
Dual PS
UPS
Gold Maintenance
High-End Server
RAID
Dual PS
UPS
Gold Maintenance
High-End Server
RAID
Dual PS
UPS
Gold Maintenance
High-End Server
RAID
Dual PS
UPS
Gold Maintenance
Code Changes to Coordinate and Distribute Work

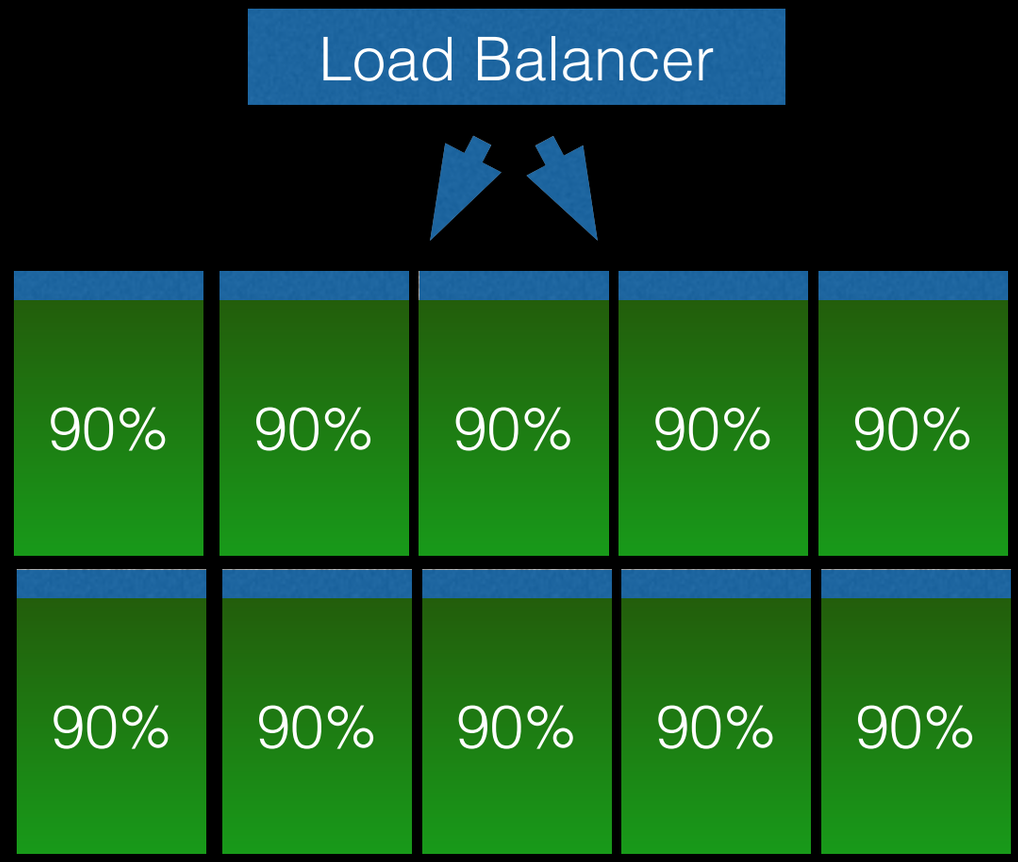
Reliability through software is better.

- Resiliency through software:
  - Costs to develop. Free to deploy.
- Resiliency through hardware:
  - Costs every time you buy a new machine.

Big resiliency is cheaper



50%
overhead



10%
overhead

The right amount of resiliency is good. Too much is a waste.

Aim for an SLA target so you know when to stop.

Load balancing &
redundancy is just one
way to achieve resiliency.

The cheapest
way to buy
terabytes of RAM.

Fail Better Part 1 of 3:

Use cheaper, less reliable, hardware.

Fail Better Part 2 of 3:

If a process/procedure
is risky, do it a lot.

Risky behavior

vs.

Risky procedures

Risky Behaviors are inherently risky

Smoking

Shooting yourself in the foot

Blindfolded chainsaw juggling

Risky behavior is risky.

Risky Processes can be improved through practice

- Software Upgrades
- Database Failovers
- Network Trunk Failovers
- Hardware Hot Swaps

StackOverflow.com
Failover from NY or Colorado

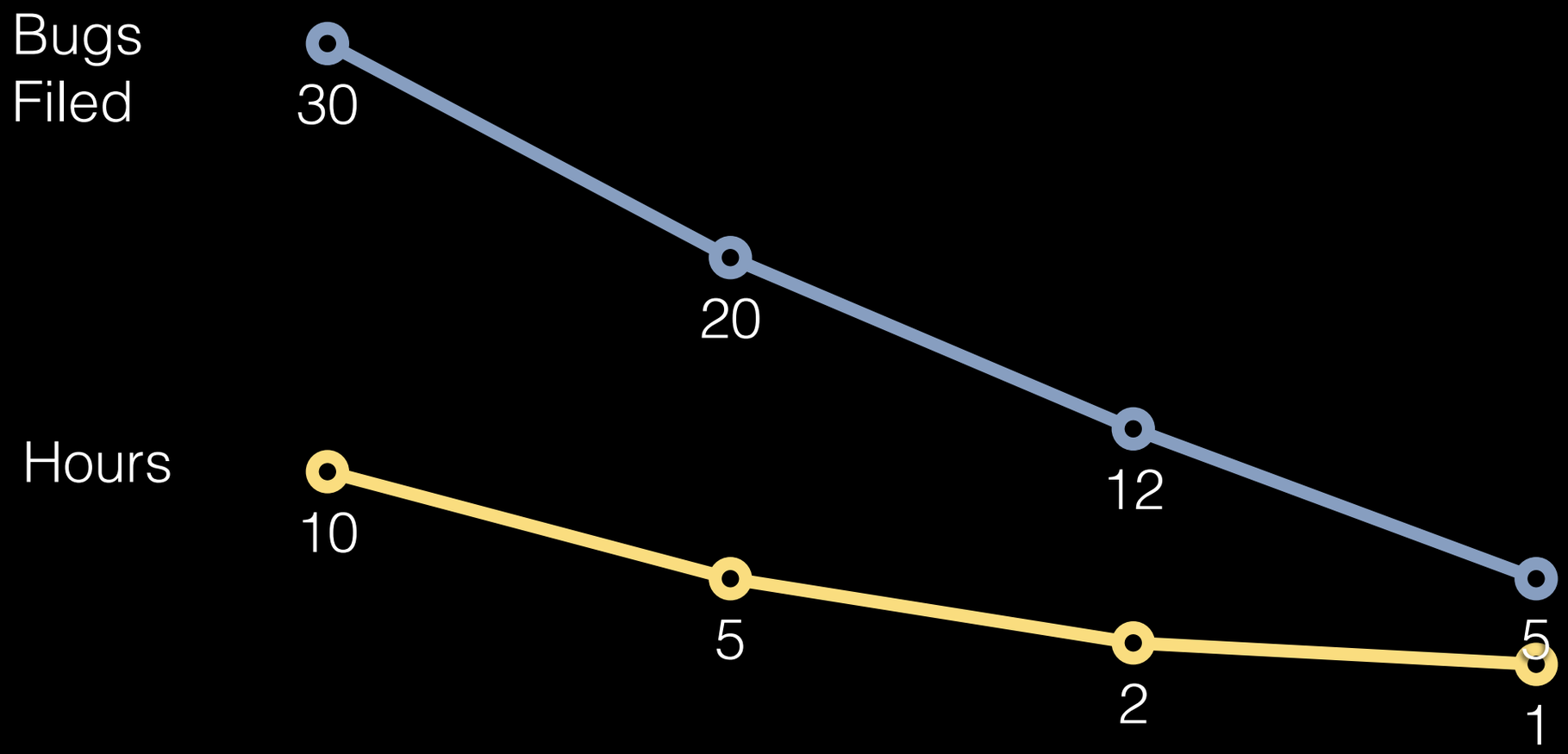
- StackOverflow.com has a “DR” site in Colorado.
- StackOverflow.com runs on SQL Server with “AlwaysOn” Availability Groups plus…

  Redis, HAproxy, ISC BIND, CloudFlare, IIS, and many home-grown applications

Process was risky

- Took 10+ hours
- Required “hands on” by 3 teams.
- Found 30+ “improvements needed”
- Certain people were S.P.O.F.

Drill Results



Why?

- Each drill “surfaces” areas of improvement.
- Each member of the team gains experience and builds confidence.
- “Smaller Batches” are better

Software Upgrades

- Traditional
  - Months of planning
  - Incompatibility issues
  - Very expensive
  - Very visible mistakes
  - By the time we're done, time to start over again.
- Distributed Computing
  - High frequency (many times a day or week)
  - Fully automated
  - Easy to fix failures
  - Cheap… encourages experiments

“Big Bang” releases are inherently risky.

Small batches are better

Fewer changes each batch:

- If there are bugs, easier to identify source

Reduced lead time:

- It is easier to debug code written recently.

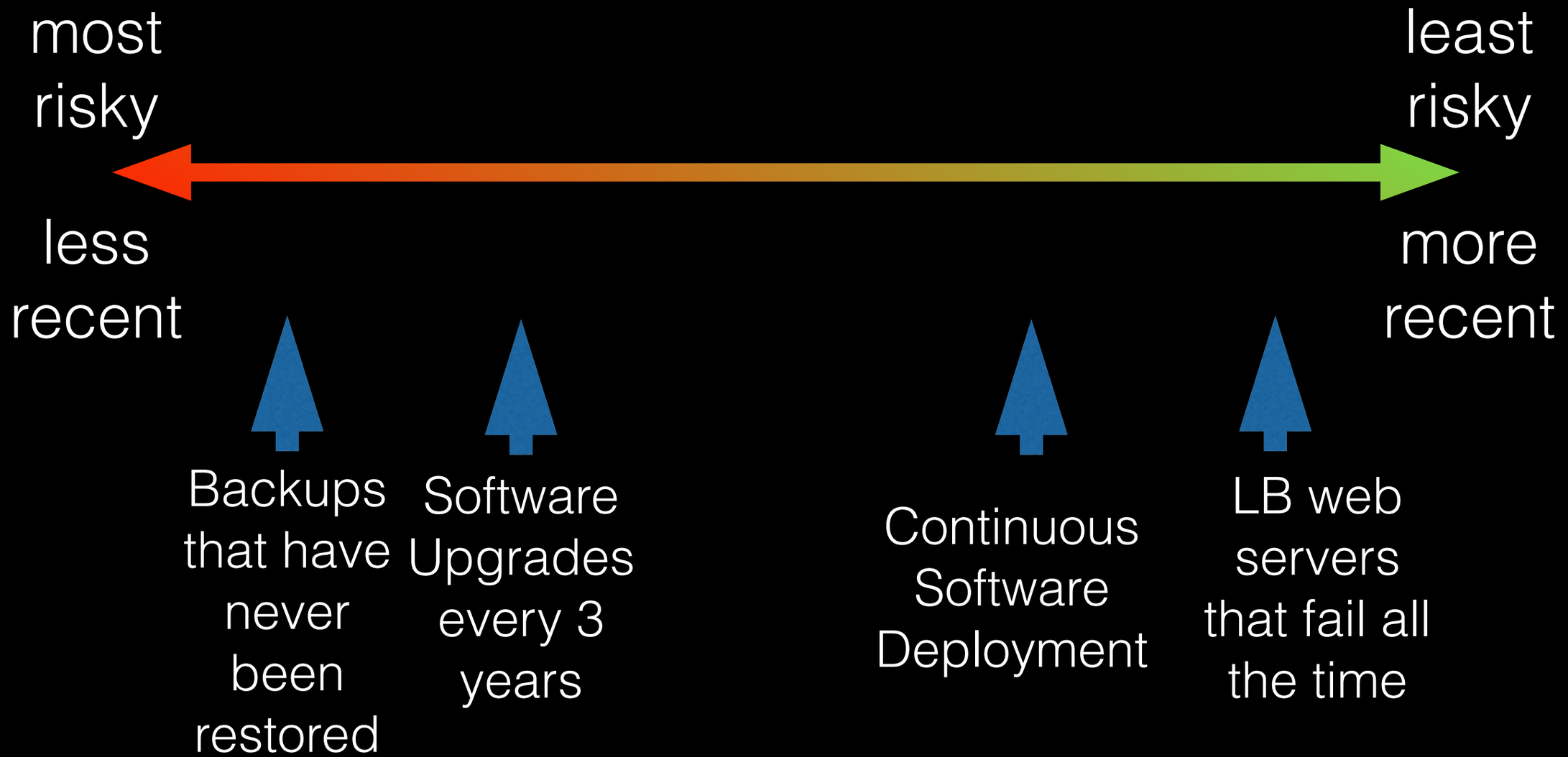
Environment has changed less:

- Fewer “external changes” to break on

Happier, more motivated, employees:

- Instant gratification for all involved

Risk is *inversely proportional* to how recently a process has been used



Netflix “Chaos Monkey”

Fail Better Part 2 of 3:

If a process/procedure
is risky, do it a lot.

Fail Better Part 3 of 3:

Don’t punish
people for outages.

There will always be outages.

Make peace with failure

Parts are imperfect

Networks are imperfect

Systems are imperfect

People are imperfect

Getting angry about outages is equivalent to expecting them to never happen… which is *irrational*.

Out-dated attitudes about outages

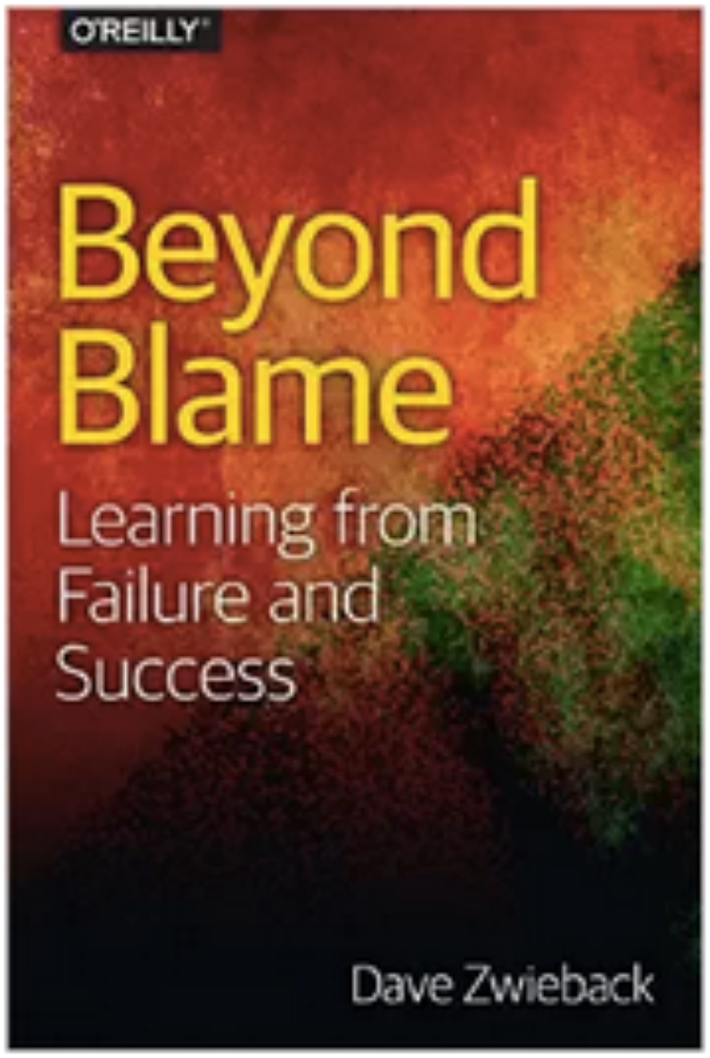
- Expect perfection: 100% uptime
- Punish exceptions:
  - fire someone to “prove we’re serious”
- Results:
  - People hide problems
  - People stop communicating
  - Discourages transparency
  - Small problems get ignored, turn into big problems

New thinking on outages

- Set uptime goals: 99.9% +/- 0.05
- Anticipate outages:
  - Strategic resiliency techniques, oncall system
  - Drills to keep in practice, improve process
- Results:
  - Encourages transparency, communication
  - Small problems addressed, fewer big problems
  - Over-all uptime improved

After the outage, publish a postmortem document

- People involved write a “blameless postmortem”
  - Identifies what happened, how, what can be done to prevent similar problems in the future.
  - Published widely internally and externally.
- Instead of **blame**, people take **responsibility**:
  - **Responsibility** for implementing long-term fixes.
  - **Responsibility** for educating other teams how to learn from this.



Beyond Blame: Learning From Failure and Success Paperback – November 1, 2015

by Dave Zwieback (Author)

★★★★★ 1 customer review

See all 2 formats and editions

Kindle	Paperback
$21.99	$16.60
Read on Any Device with the Free Kindle App	1 New from $16.60

LOOK INSIDE! "Big Magic" by Elizabeth Gilbert

Outage Post-Mortem - 2014-08-25/Outage

Summary:

On Aug 25, 2014 there was an outage of all web properties (Core and Careers) from 3:27pm to 3:32pm NYC-time (approx 7 minutes). The cause was an incorrect change to security settings. The solution was to revert the change via Puppet. Measures being implemented to prevent this problem in the future are listed below.

Outage Type	Sites Down
Outage Timeframe	2014-08-25 19:24, about 7m of downtime
Worst-Case Outage window	7 minutes
Assets affected	All
Summary of causes	Bad change to firewall rules.
Recommendations	Need to refactor firewall rules to be more easy to understand and update; Need to develop better testing methods for firewall rulesets.

Background Information

The intended change: SRE was attempting to update the firewall rules to to permit internal openid calls to work directly rather than going out to the internet and back in.

Outage Schedule of Events

2014-08-25 19:01 da2d38d6a Change pushed to Git
2014-08-25 19:26 Puppet runs on ny-lb05 (pushed bad change / outage BEGINS)
2014-08-25 19:27 Pagerduty and Pingdom page oncall (Tom)
2014-08-25 19:27 @David asked "Who broke everything but chat?" on SRE-team
2014-08-25 19:27 da2d38d6a1 Revert pushed to Git
2014-08-25 19:30 Puppet runs on ny-lb06 (pushed revert)
2014-08-25 19:32 Puppet runs on ny-lb05 (pushes revert) (outage RESOLVED)

Things that went Right

- Use of version control with Puppet means we are able to revert bad changes quickly.
- Everyone worked together to find and fix the problem.

Processes Needing Improvement

- Firewall rules should be refactored to be easier to understand and update.
- Firewall rules need a better testing method.

Immediate to do

- Improve comments in iptables files (there are wrong and misleacing comments) (Done: b55e654d9f)

Long term to do

- Move LB firewalls to the new structure being developed.
- Establish better testing methodology for firewall changes.

I dunno about anybody else, but I really like getting these post-mortem reports. Not only is it **nice to know what happened**, but it's also great to see **how you guys handled it** in the moment and **how you plan to prevent these events** going forward. Really neato. Thanks for the great work :)

—-Anna

Fail Better Part 3 of 3:

Don’t punish
people for outages.

Take-homes

- **“cloud computing” = “distributed computing”**

1. **Use cheaper, less reliable, hardware**
   - Create reliability through software (when possible)
   - Pay only for the reliability you need
2. **If a process/procedure is risky, do it a lot**
   - Practice makes perfect
   - “Small Batches” improves quality and morale
3. **Don’t punish people for outages**
   - Focus on accountability and take responsibility

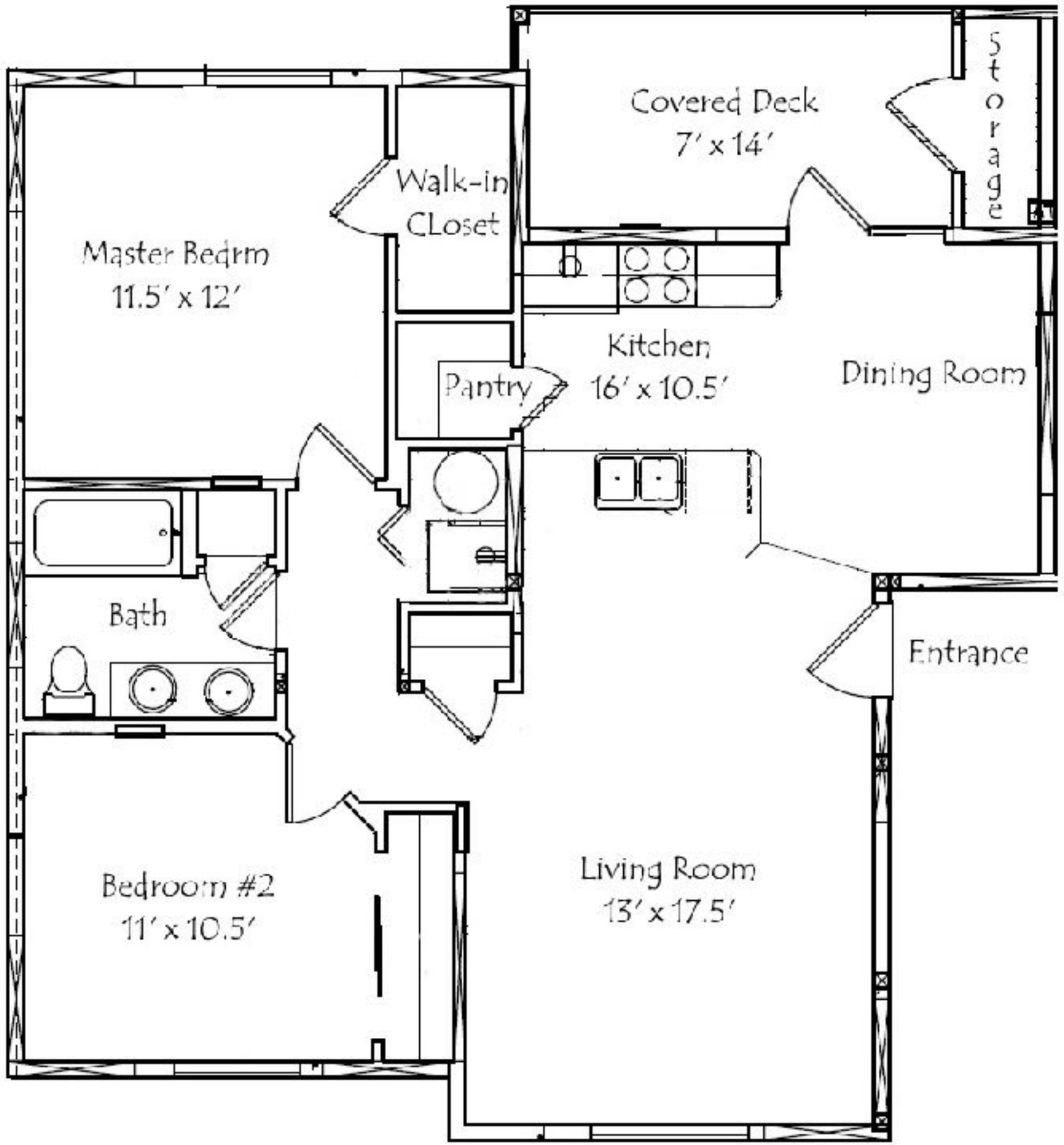
?

TOYOTA

HOW TO IMPLEMENT LEAN MANUFACTURING

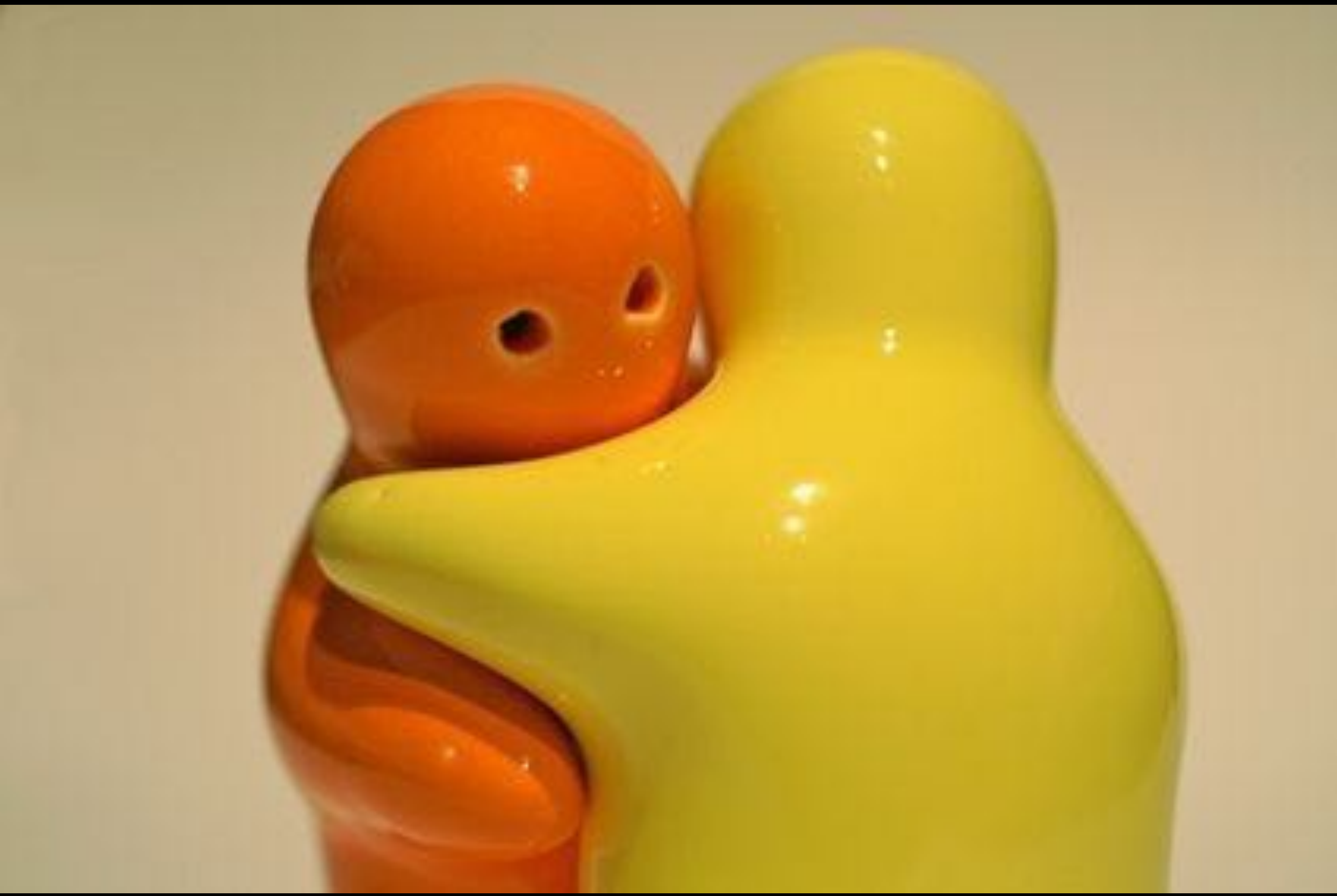
LONNIE WILSON

Home Life


Covered Deck
7' x 14'
Storage
Walk-in
CLoset
Master Bedrm
11.5' x 12'
Kitchen
16' x 10.5'
Pantry
Dining Room
Bath
Entrance
Bedroom #2
11' x 10.5'
Living Room
13' x 17.5'

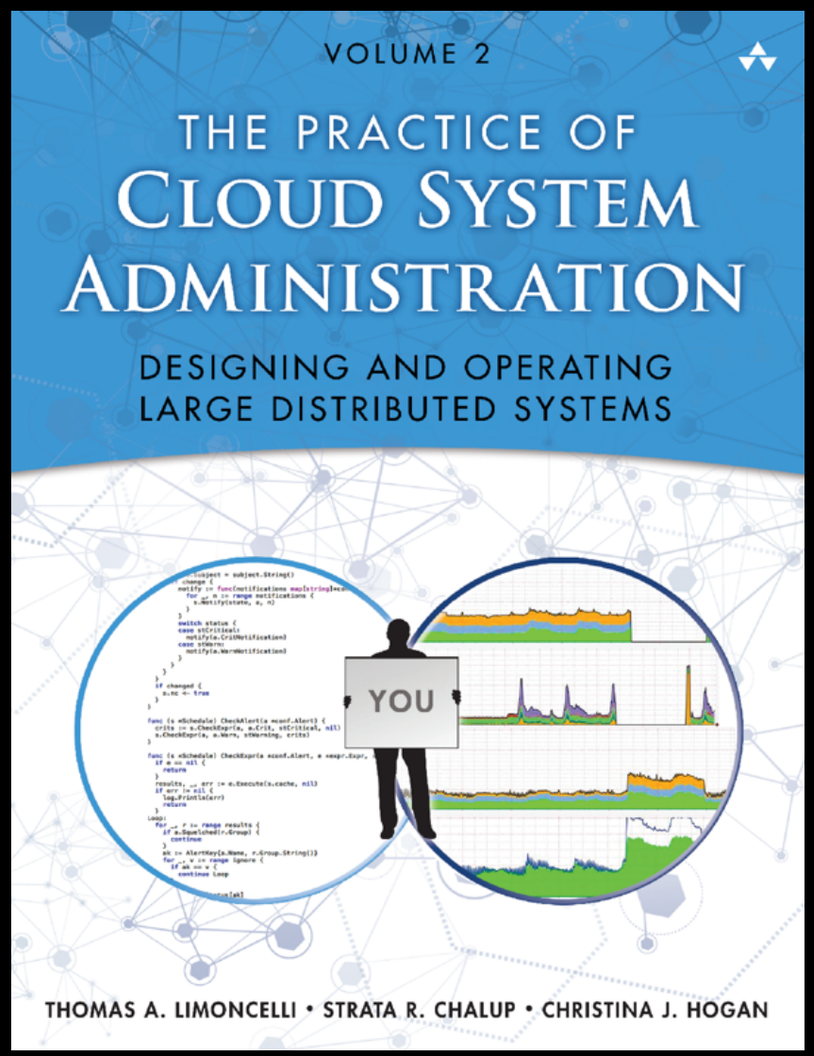

Occupancy
249


AWKWARD
MOMENT



Fail Better!

Very Reasonable ideas from The Practice of Cloud System Administration



Tom Limoncelli, SRE

StackOverflow.com

the-cloud-book.com

@YesThatTom

www.informit.com/TPOSA
Discount code TPOSA35

Take-homes

- **“cloud computing” = “distributed computing”**

1. **Use cheaper, less reliable, hardware**
   - Create reliability through software (when possible)
   - Pay only for the reliability you need
2. **If a process/procedure is risky, do it a lot**
   - Practice makes perfect
   - “Small Batches” improves quality and morale
3. **Don’t punish people for outages**
   - Focus on accountability and take responsibility

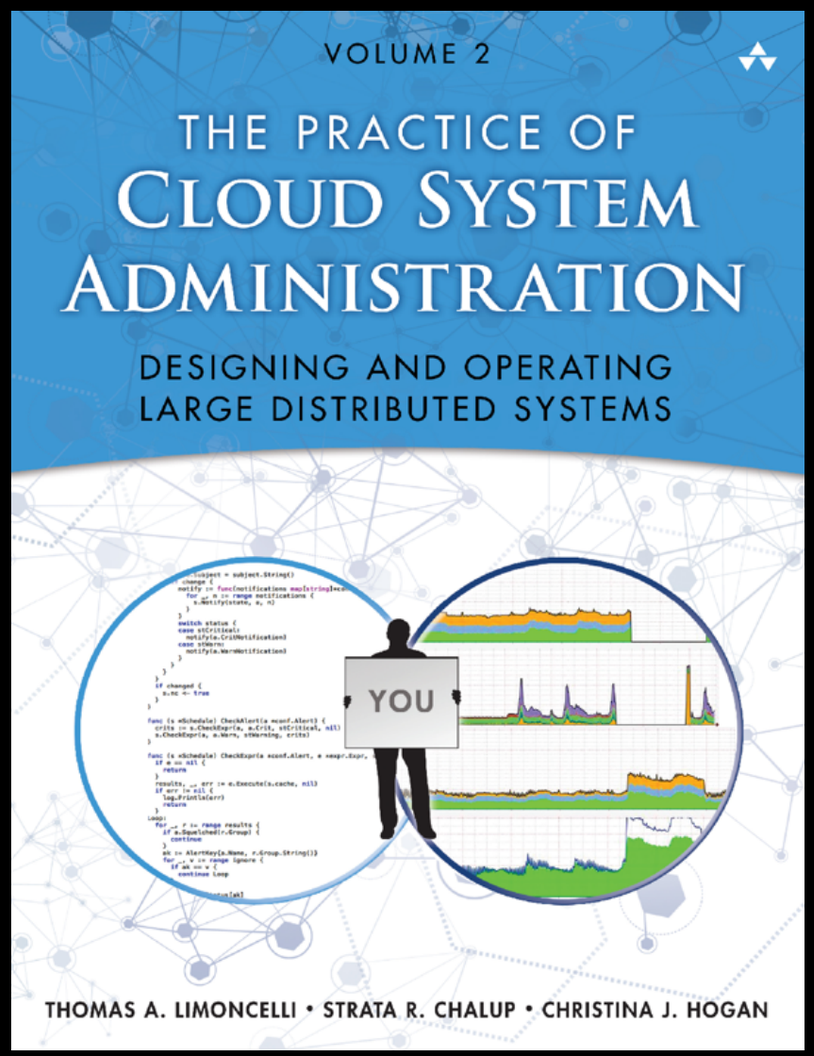
I need help with…

Finding case studies of DevOps in Ops-only environments: Desktop support, hardware operations, etc.

Q&A

Fail Better!

Radical ideas from The Practice of Cloud System Administration



Tom Limoncelli, SRE

StackOverflow.com

the-cloud-book.com

@YesThatTom

www.informit.com/TPOSA
Discount code TPOSA35

the-cloud-book.com

@YesThatTom