Post Ops Revisited:

Operations Recovery, or Are We There Yet?

Todd Underwood, Google

Post Ops Revisited:
A Quirky, Cranky Call to Action

Operations (and Systems Administration) is finished. DevOps is a helpful band aid. We need to move beyond the entire notion of “Administration” and “Operations” in the field of computing (software, systems and networking).

Let us stop doing the machines’ work for them.

Let us stop feeding the machines with human blood.

Post Ops Recovery: Are We There Yet?

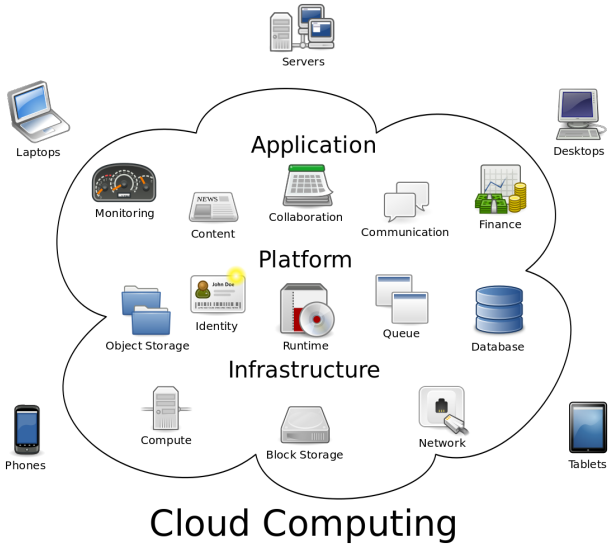
"Cloud" is our new platform.

It's not terrible

Some problems are solved.

Most solved problems undeployed or broken in deployment.

Some problems remain unsolved.



Solved & Deployed | **Solved but Broken (or undeployed)** | **Unsolved**

About me / context

Google SRE, working on Ads Quality, Commerce (Payments, Travel, Shopping), based in Pittsburgh, PA, USA.

- I got my first systems gig in the mid-1990s.
- Grew up running Internet Services at an ISP (AS2901 anyone?)
- I've swapped tapes and run cable, I've written crontab entries and automated account creation across 5 Unices and NT.
- I assembled datagrams by hand and calculated checksums on my fingers

Not a cloud expert (is anyone?).

Apologi[zs]e for use of 'cloud' up front. But we need a word for this, and we'll call it cloud.

Cloud Computing!

Image by Quinn Dombrowski, CC BY-SA 2.0

Cloud Bunny!

NB: Some content is gratuitous and purely for audience amusement.[1]

[1] http://www.buffymusical.com/Theory_Bunnies_Together.htm Take appropriate measure wrt bunnies.

Site Reliability Engineering

A quick reminder for an audience who already gets it


DURI ET PERITI
MISSION CONTROL
SRE
SITE RELIABILITY ENGINEERING

Site Reliability Engineering: A Mythical History

(This is not exactly what happened, but it's somewhat True anyway)

- Google was (still is) ~~cheap~~frugal: costs that scale linearly (or super-linearly) are bad when things get big
  - Applies to hardware (server and network)
  - Applies to people: no "NOC", no "sysops" in production
- Software developers run their own code in production
- They ~~hate~~dislike it, so they automate everything:
  - build, push, monitoring,
  - task restarting, task configuration and location,
  - distributed debugging, etc.
- Some are more production-oriented than others. These become the first "Site Reliability Engineers."

SRE is systems and software engineers who solve production problems with software.

We are constructive cynics. Productive pessimists. Creative cranks. Alliterative as……. Nevermind.

SRE Basics

Keep the service up—whatever it takes

Service unavailable? Our problem, whatever the reason

Work at a Large Scale

Many services, lots of data, many machines

Not so many people. People scale sub-linearly to services.

Balance competing demands

- Improve availability and reliability
- Improve efficiency

Solve production problems with software. It's all just software.

Shout Out: Adrian Cockcroft's "NoOps"

Adrian Cockcroft's "NoOps"[1]

Principle: Software developers work directly with production. No operations organization. Automation removes operational tasks entirely.

Platform as a Service: PaaS ; No operations required.

Implicit assumption (me, not Cockcroft):

- Services/jobs not VMs/machines
- scalable/programmable cluster OS that works
- [machine] configuration is the wrong level of abstraction
- configuration management **is** one right problem

[1] http://perfcap.blogspot.com/2012/03/ops-devops-and-noops-at-netflix.html

Operations Research, {Sys|Net|Dev}Ops

The historical, academic part of the talk you can ignore, perhaps with some useful tidbits.

A Brief Diversion in History:
Operations Research / Operations Management

- Basic principles:
  - statistical and process expertise to improve some process (often manufacturing).
  - maximum yield of some metric given a set of constraints
- Widespread application outside of manufacturing:
  - War
  - Critical path analysis
  - Network fault analysis
  - Scheduling
  - Project Management

The Project Phoenix[1] is a case in point:

Software-based production? What are the constraints?

[1] Kim, Behr, Spafford. The Phoenix Project. IT Revolution Press 2013

OR: Irrelevant for Software

- “Production” software systems lacks the constraints

- OR critically relevant for the physical infrastructure stack
  - Warehouse-scale computing
  - External adjacencies (power, fibre, other companies) not amenable to quick change
  - That’s a different talk

Software isn’t a factory.

Software Organizations Should Nix Operations

- Operations comes from the notion of extracting value out of a fixed set of assets.
- System/network operations:
  - Fixed asset (minicomputer, mainframe, RS6k, BFR, T640 name your poison)
  - Depreciating rapidly
  - Extract value by keeping it running

What is the fixed asset being depreciated in EC2? As a cluster application service engineer at Google, what am I "administering"?

Abstractions/software changes the cost of change.

Production vs. Operations

...of babies and bathwater

Operations culture and practice has many admirable characteristics we must not lose:

- Fast, careful troubleshooting
- Ethic of caring about production, availability, users
- Ethic of privacy, security
- Constructive pessimism brought to capacity planning, outage prediction, scaling, future failures in general
- <Insert your best “why I’m proud of being a sysadmin” here>

Production engineering practices and systems should recreate these values effortlessly.

Organizations that move post-Ops should embody these values.

Ops->DevOps->{NoOps|PostOps}

- Ops separate from everything else is terrible (wall of confusion).
- Embedded ops into other teams is a good start but not far enough.
- **Clueful Management** is mandatory
  - Some companies clearly have very technical management
  - If your management cannot understand your job, then either your job or your management don't matter.
- Need just enough operations to identify **new** problems and prioritize development. Cap ops work (50% or less?).
- Every other aspect of operations should be eliminated. With prejudice.

Burden of proof: Operations. Why do we want to preserve this specialty and expertise?

Clusters, Jobs, Tasks

Oh, my! What does a production systems or software engineer engineer do in the PostOps age?

A Likely Future for Production

Platforms

- hardware selection/custom building
- datacenter selection/building/operations
- power work, physical infrastructure, hardware operations

Infrastructure Software Engineering

- server/cluster OS development/testing
- network/storage/naming/shared services development

<Insert SRE/Production Software Engineer here>

Application Development / Application Administration

Systems/Production/Operations

Babies

- Production ethic
- Troubleshooting / problem solving
- Building Automation Systems
- Job/system/intent-based configuration management
- Monitoring systems and implementations
- Release engineering / canarying / testing / roll-out
- Capacity planning
- Incident management
- SLA definition/monitoring
- Constructive cynicism

Bathwater

- Rote/repeatable work
- Automatable work
- One-offs
- (OS) configuration management
- Logging in to machines
- Processing individual files
- Heroism

Progress?

We need to:

1. Agree upon, deploy stable APIs for platform infrastructure
2. Continue to innovate at the top of that platform
3. Build a culture and set of practices of application production engineering on top of that platform.

Iterate as necessary.

SRE has a first-class responsibility to execute on this.

Are we there yet?

Deployed Solutions

Everyone has these

Settle on the Stack

The more basics we agree upon, the more interesting conversations we can have about what to do with it.

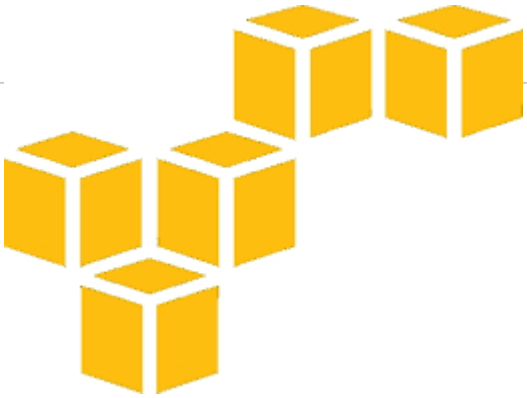
Look at:

- Widely deployed
- Tested
- Agreed-upon
- stable

APIs/services.

Available to the general public.

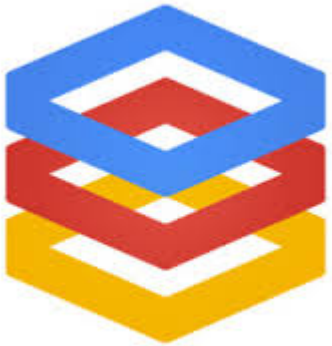
Public Cloud



"Cloud" is the new "Ethernet":

I don't know what the next paradigm of distributed, outsourced computation will be, but I know it will be called 'Cloud'.

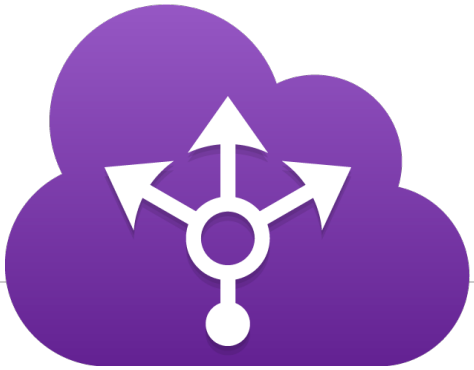
APIs for **basic** compute services exist:



- VMs
- Object storage
- Load Balancing/traffic management
- CDN for small objects & video

None of this is the right level of abstraction.

Better than Enterprise IT.



Not There Yet

Or not widely deployed, or available but people don't use it yet, or broken in deployment, or so unreliable as to be effectively undeployed.

Jobs/Tasks vs. Containers vs. VMs

VMs, the good:

- Clean API
- Security/Isolation
- Portability

VMs the bad:

- Extremely heavyweight
- Bad performance

Alternative: Jobs/Tasks: Google Borg and Apache Mesos. Fine grained, efficient. Possibly confusing to most users.

Likely compromise: Containers, with Kubernetes management (or EC2 Container Service or Docker Shipyard)

IO/Storage

Low latency, persistent (posix-compliant?) block storage

Local disk, EBS

Stable API to object storage

S3, GCS, many others

Dynamic geographic/instance relocation of storage.

Oh, so very many things we wish we had from our storage systems.

(Semi-)Structured Storage

Hbase
Cassandra
redis
CouchDB
MongoDB
Google Cloud Bigtable?

Different approaches, not converging into a stablized API yet.

distributed, replicated transactions:
spanner[1] (not available as a service yet, I think)

[1] http://research.google.com/archive/spanner.html

Unsolved Problems

No one has these

Automated Everything

- Build on commit
- automated testing
- push on green
- automated canarying with auto-rollback
- automated backups/restores for arbitrarily complex stacks on top of standard datastores

In other words:

- <an automatic pony>

Production<->Development

Tight, automated, seamless coupling of development with production.

- This code is new and causes crashes
- This new deployment fixes these bugs
- Regression test is 5% slower because this method is slow

Development environments need a tight coupling to production environments/monitoring.

Monitoring/Notification

Application-specific metrics exported and available

Service inspection for graphing, notification and alerting (monitor the right stuff, a dashboard that makes sense)

Service-to-service (or second-party or infrastructure) notifications: tell me when my use of your datastore is wrong.

TL;DR

- ~~Life~~The industry moves pretty fast. If you don't stop and look around once in a while, you could miss it.[1]
- We are collectively inventing a place where we can be SREs everywhere.
- It’s not done yet.

For SREs: Let us push for standardization as quickly as feasible and let us build production automation worthy of our attention.

[1] Ferris Bueller’s Day Off. But you knew that, already, obviously.

Questions? Kvetches? Rotten Tomatoes?

A cute kitten to distract the audience.