Operations at (small) Scale

Who are you?

Elliott Sims

Currently Senior Sysadmin at Backblaze

SRE, SRO, ClusterOps, and Production Engineer at Facebook from 2009-2015

Different job titles, same basic job:
"make the stuff work"

What I'm talking about

1. What's different?
2. Automation
3. How to evaluate options
4. Hosting
5. Monitoring/paging
6. Communication and Coordination

So, what's different?

You have to understand everything, and can't just stay in your corner. Also, it's humanly possible to understand everything.

Fault tolerance: A bit trickier than when you have thousands of hosts. Some special snowflakes are unavoidable

On-call: It can be a lot harder to ensure that there's always multiple people who can fix certain things

So, what's different?

Resources: you can't just grab 100 servers when you need them.

So, what's different?

Resources: you can't just grab 100 servers when you need them.

…but if you need twice as many servers as you thought, you don't need to build an entire DC for them first

So, what's different?

Resources: you can't just grab 100 servers when you need them.

…but if you need twice as many servers as you thought, you don't need to build an entire DC for them first

More important resource: people and knowledge. Can't just walk over one desk or one building and find one of the top experts in the world.

You have to become the expert you need.

So, what's different?

Ever looked at older and more established infrastructure and think "what idiot designed this?"

So, what's different?

Ever looked at older and more established infrastructure and think "what idiot designed this?"

Now it's your turn to be that idiot!

Don’t build for 1B users

Lots of advice out there for scaling up given large resources.

Elaborate scalable management systems are great, but lots of unnecessary overhead for small environments

Even 1B-user systems mostly were designed smaller and rewritten part-by-part as they were outgrown

Simple and stable is important. Production-tested stuff that “everyone uses” breaks less than shiny new “web scale” that’s been used in 0 or 1 environment.

But do build certain automation

Still certain things you should do, though, even with 2 hosts:

Backup. Live/on-site and also archival or off-site

Version control. Not only for code, but also for configs.

Configuration management. Ansible, Puppet, Chef, Salt, etc. Anything is better than nothing.

CMDB: Something should be the source of truth about what exists and where.

How do I decide what to use?

Financial cost (in people and dollars)

Time cost (both person-hours and time-to-market)

Control of your own destiny. Overlooked, but important.
Actually understanding all of your infrastructure with fewer black boxes makes diagnosing problems easier.
SLAs let you blame someone else for downtime, but they don't make your site any less offline.

Categories

Hosting: Affects pretty much all your other choices. Cloud vs Colo vs build

Monitoring and paging

Network equipment

Communication and coordination.

Hosting

AWS: Scales great, pretty complicated (especially Glacier billing)
  high opex
  To some extent, you can pick your tradeoffs
    (ELB/RDS vs DIY
  medium/low destiny control
  low person-hours (though complexity adds some)
  low setup latency
  AWS is reliable, but individual EC2 instances aren't

Also Azure and Google

Hosting

Less-cloudy VPS (Linode, DigitalOcean, etc)

Tends to scale in hours or days, not minutes

Usually cheaper for relatively static needs

Simpler setup and billing

Individual instances usually a bit more reliable

but it's all on you. No RDS/ELB/S3/etc

Hosting

Dedicated server: like a VPS, but worse in almost every way. Rarely a good option any more.

Colo: renting DC space
Good if you benefit from special hardware, or have predictable growth.
  High cost in person-hours.
  High start latency. Finding DC, building racks
  Lots of control over your own destiny
  High initial cost, but OpEx can be lower.

Build: Don't start here. Good if you need hundreds of thousands of hosts, and can predict demand well, and have lots of capital and employees.

Monitoring/paging

Don't build your own paging. It's harder than you think. When stuff is broken, it might be too broken to notify you. Just pay Pagerduty/VictorOps/OpsGenie

Monitoring: If you're coming from FB or Google, you're going to be sad.

Lots of subcategories, many solutions cover several.

- Instrumentation (especially performance)
- Log analysis
- port/health checks
- metric collection
- alerting
- metric display/querying/analysis

Monitoring/paging

Log analysis: Splunk if you have $$, loggly/loglogic/SumoLogic if you have $, Elasticsearch+Logstash+Kibana if you have time

Port/health checks: Good(?) Old Nagios, Icinga2, Zabbix, ZenOSS. Should have at least a little external monitoring from Pingdom/Alertsite/etc, not just angry Twitter users.

Metric display/querying/analysis: ZenOSS/Zabbix are all-in-one. Grafana/Graphite/Prometheus/wavefront.io for metric display/analysis. Some analyzers include storage, but if not there's OpenTSDB and InfluxDB.

Network

Unless you go colo or build, this is a black box. Hope it works, work around it when it doesn't.

If you build your own: Nobody ever got fired for buying Cisco.

Network

Unless you go colo or build, this is a black box. Hope it works, work around it when it doesn't.

If you build your own: Nobody ever got fired for buying Cisco… but if their startup runs out of cash they're still out of a job.

Network

Unless you go solo or build, this is a black box. Hope it works, work around it when it doesn't.

If you build your own: Nobody ever got fired for buying Cisco… but if their startup runs out of cash they're still out of a job.

Ciscoish traditional options: Juniper, Arista, Extreme, Force10

Whitebox with third party software: Broadcom Trident+/TridentII/Tomahawk from any vendor, plus Cumulus or Pica8, or SDN like BigSwitch or Nicira. Or if you're feeling ambitious, FBOSS or MS-SONiC.

Communication and coordination

Probably shouldn't build your own here unless you need total control (or regulatory requirements)

Old-school: IRC and email, dirt cheap and simple

Should probably outsource to Google/Quip/Hipchat/Slack/fb@work

Task/bug/incident tracking: Asana, Basecamp, Jira, Bugzilla, RequestTracker, Pivotal, Maniphest, and many more. Pick what matches your workflow, or what people have used before. Minimizing friction is important

Documentation: Pick a wiki. Google Docs has good search, but wiki is more structured and linkable.

Questions?