Scalable System Operations
About This Talk

- Set of principles
- Operations Engineering
- Tumblr project
- Server management
- Massively automated

- Software
- Techniques
- Example code
- Best practices
- Open source
About Me
About Me

- 1995: CompUSA
  Intro to The Internet
- 2000: Guru Labs
  Sun, Cisco, Red Hat
- 2002: Red Hat
  Sys admin courseware
About Me

• 2004: Fortress Systems
  Anti-spam/malware
• 2005: Red Hat
  Virtualization cert
  Remote learning
  Defined “cloud”
• 2011: Tumblr
  Lead Systems Eng
The Problem

Deploying new servers is very repetitive and slow. (and we hate that)
The Job We Don’t Want
The Solution

Automation
Automation

- Install OS
- Configure OS
- Install software
- Configure software
- Add to DNS
- Add to monitoring
- Add to trending

- Firmware
- Configure BIOS
- Set up BMC
- Inventory
- Stress testing
- Network config
The Goal Is Clear

Automation
Time To Strategize
Time To Strategize

Use open source?
Which?
Buy software?
Which?
Write software?
Mix and match?
The Choice Principle

The time to make a decision is a function of the possible choices.
Rapid Software Research
Rapid Software Research

1. Define
2. Gather
3. Disqualify
4. Rank
Rank

http://www.flickr.com/people/nirak/
Rank

- Modularity
- Compliance
- Novelty
- Disruption
My Requirements

- Asset inventory
- State management
- Robust API
- Event triggers
My Requirements

- Modular
- Flexible
- Extensible
- Fast
My Requirements

Manage physical hardware as easily as virtual machines.
The Usual Suspects

- Cobbler
- Foreman
- Satellite
- Orchestra
- Racktables
- Clusto
But Wait!

Creating a Server

Creating a new Entity is easy; just create an instance of the Driver class:

```python
from clusto.drivers import *
PenguinServer("server")
```

Great! You've created a server named "server", using the provided PenguinServer Driver. This server has been automatically added to Clusto's database. But the code above didn't keep track of the PenguinServer instance, so let's fetch it:

```python
server = clusn.get_by_name("server")
```

Clusto saw that the Entity's driver was PenguinServer, and it automatically instantiated an instance of PenguinServer to hold the Entity.
Data Entry

“Just import the data supplied by the hardware vendor...”
Missing Requirements

☐ Firmware
☐ Configure BIOS
☐ Set up BMC
☐ Inventory
☐ Stress testing
☐ Network config
☐ Add to monitoring
☐ Add to trending
We have to write software!
We have to write software!

- Delivery Schedule
- Scope Creep
- Maintenance
- Documentation
Tumblr Management Stack

philcollins is now following you. ^_^
The Glue Principle

Unix Rule of Parsimony:
Write a big program only when it is clear by demonstration that nothing else will do.

http://www.flickr.com/people/kodomut/
The Standards Principle

The nice thing about standards is that you have so many to choose from.
-Andrew Tanenbaum

http://www.flickr.com/people/usfwssoutheast/
The Simplicity Principle

Unix Rule of Simplicity:
Design for simplicity; add complexity only where you must.
The 3:00 AM Principle

It must be obvious to someone woken up from a sound sleep at 3:00 am.
The Don’t Break The OS Principle

The software should NOT prevent the OS from working as expected.

http://www.flickr.com/photos/philmanker/
The Amnesia Principle

Given enough time, you WILL forget why you did that.

http://www.flickr.com/people/zach_a/
Tumblr Management Stack

- iPXE
- Invisible Touch
- Collins
- Phil
- Kickstart
- Puppet
Why not pxelinux?
Why not pxelinux?

- TFTP
- Flat files
iPXE

- HTTP, FTP, iSCSI
- Scriptable
- Variables
- Dynamic
# subnet for the provisioning vlan
subnet <%= subnet %> netmask <%= netmask %> {
  option domain-name "<%= option_domain_name %>";
  option routers  <%= option_routers %>
  option domain-name-servers <%= option_dns_servers.map{|i| "#{i}"}.join("", "") -%>
  option subnet-mask  <%= option_subnet_mask %>
  default-lease-time 21600;
  max-lease-time 43200;
  range <%= range_start %> <%= range_end %>
  # If a pxe request comes in from ipxe send the config url
  if exists user-class and option user-class = "iPXE" {
    filename "<%= ipxe_config_url %>"; # http://foo.example.com/ipxe/${net0/mac}
  } else {
    next-server <%= next_server %>; # tftp server
    filename "<%= filename %>"; # path to ipxe binary on tftp server
  }
}
Fedora LiveCD Tools

```
lang en_US.UTF-8
keyboard us
timezone US/Eastern
auth --useshadow --enablemd5
selinux --enforcing
firewall --disabled
repo --name=centos --baseurl=http://127.0.0.1/pub/repo/centos/os/6.2
repo --name=infra --baseurl=http://127.0.0.1/pub/repo/infra/6.2
repo --name=epel --baseurl=http://127.0.0.1/repo/epel/6/x86_64/

%packages --excludedocs
@core
dracut
dracut-kernel
device-mapper
device-mapper-event
%end
```
Invisible Touch Kickstart

```plaintext
# Invisible Touch Live OS image
%include centos-6.2-livecd-minimal.ks
%packages --excludedocs

it
%end
%post

cat > /etc/issue <<EoF
Invisible Touch Live OS v0.0.4
Kernel \r
EoF

# set ipmi to start at boot up
/sbin/chkconfig ipmi on

# configure rsyslog

cat >> /etc/rsyslog.conf <<EoF

# invisible touch
local0.*                                                /var/log/it.log
local0.*                                                /dev/tty7
EoF

%end
```
Invisible Touch Utilities

- lshw
- llpdpd
- Breakin
- ipmitool
- Bash scripts
lshw

lshw generates hardware info XML

```
<node id="disk:1" claimed="true" class="disk" handle="SCSI:04:00:01:00">
  <description>ATA Disk</description>
  <product>ST91000640NS</product>
  <vendor>Seagate</vendor>
  <physid>0.1.0</physid>
  <businfo>scsi@4:0.1.0</businfo>
  <logicalname>/dev/sdf</logicalname>
  <dev>8:80</dev>
  <version>n/a</version>
  <serial>9XG0ETB8</serial>
  <size units="bytes">1000204886016</size>
  <configuration>
    <setting id="ansiversion" value="5" />
    <setting id="signature" value="000e1763" />
  </configuration>
  <capabilities>
    <capability id="partitioned">Partitioned disk</capability>
    <capability id="partitioned:dos">MS-DOS partition table</capability>
  </capabilities>
</node>
```
lldpd

lldpctl outputs network info in XML
Breakin

Stress testing framework
Breakin

- Standard tools
- LINPACK
- Extensible
- Bash scripts
Invisible Touch

- Firmware
- Configure BIOS
- Set up BMC
- Inventory
- Stress testing
- Network config
Collins

- Asset management system in Scala
- REST API
- Client libraries in Ruby, Python and Bash
- Shell tool for scripting and automation
- Callback system for hooking into events
- Granular permissions model
- Flexible web and API based provisioning
- Remote power management
- IP Address allocation and management
- Distributed mode for spanning data centers
Collins
Asset management for engineers

About
Collins started as a system to manage all of the physical servers, switches, racks, etc in Tumblr's production environments. As we started to inventory hardware, IP addresses, software, and so on, we found the API and data gave us an excellent way to drive automation processes. Today Collins can do push button cluster (HBase, Hadoop, web, etc) deployment, drive configuration generation when hardware cluster topologies change, drive infrastructure updates when software configuration changes, and help manage software deployments.

Because of the loosely coupled design of Collins, consistently applied conventions are a system requirement. This document serves as a guide to those conventions as well as the basic core concepts of the Collins system. If you’re just interested in the basic how-to or screenshots, click here.

Approach
Collins is extremely dumb. It knows about assets, their meta-data and asset tags. You can think of Collins as a key-value store where each asset has its own set of key/value pairs. There are no relationships between assets other than the ones you, through convention, derive. The API makes it trivial to create and manage the tags (meta-data, key/value pairs) associated with an asset, and to query based on those tags.

Collins is intentionally dumb. It worries about basic authentication, clean API interactions, and data persistence. If you start thinking, “Hey, I should build X into Collins”, you probably shouldn’t. Collins supports both a plugin architecture (for things that actually in some way change the behavior of collins) as well as a very usable API (including clients in Python, Ruby and Bash). Nearly everything you might want to do can be accomplished via the API and anything that can’t is doable as a plugin.

Pages
Introduction
Basic Concepts
Collins Functions
Provisioning, logging, cancelling, reboots, searching
Integration Points
Systems that integrate with Collins
The Collins API
RESTful interaction with your assets
The Asset API
Manipulating and querying assets
The Asset Management API
Managing assets
The Asset Log API
Create and query log data
The Asset Tag API
Query all tags
The IP Management API
Manage and query IP addresses
Tag Usage and Conventions
What tags are in use for what purposes
Callbacks
Callback Mechanism in Collins
Configuration
Configuration Options in Collins
Collins Asset Details

**Server Details**

### Hardware Details

<table>
<thead>
<tr>
<th>ID</th>
<th>Speed</th>
<th>MAC Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1 Gbps</td>
<td>00:11:22:33:44:55</td>
<td>802.11gn Gigabit-Ethernet Connection - Real Conexión</td>
</tr>
</tbody>
</table>

### CPU

<table>
<thead>
<tr>
<th>ID</th>
<th>Threads</th>
<th>Speed</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>3.5</td>
<td>AMD Opteron(TM) Processor (HT Technology)</td>
</tr>
</tbody>
</table>

### Memory

<table>
<thead>
<tr>
<th>Bank</th>
<th>Size</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4 GB</td>
<td>Entry Memory Bank</td>
</tr>
<tr>
<td>1</td>
<td>4 GB</td>
<td>Entry Memory Bank</td>
</tr>
</tbody>
</table>

### Disk

<table>
<thead>
<tr>
<th>ID</th>
<th>Size</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>200 GB</td>
<td>SGE25080NWS15S</td>
</tr>
</tbody>
</table>

### Network Interfaces

<table>
<thead>
<tr>
<th>ID</th>
<th>Speed</th>
<th>MAC Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1 Gbps</td>
<td>00:11:22:33:44:55</td>
<td>802.11gn Gigabit-Ethernet Connection - Real Conexión</td>
</tr>
</tbody>
</table>

---

Collins Resource Manager

Search, Logging, Create, Logout
Phil

• iPXE dispatcher
• Kickstart generator
• Light Ruby app
• Collins API client
Server Intake Workflow

1. Rack and stack
2. Power on
3. Enter physical data
Server Intake Process

1. Server boots iPXE via DHCP/PXE
2. iPXE gets config from Phil
3. Phil sends Invisible Touch
4. IT updates firmware (if needed)
5. IT configures BIOS
6. IT configures BMC
7. IT uploads inventory data to Collins
8. IT starts stress tests
9. IT powers down server
Provisioning Workflow

1. Search Collins
2. Choose Profile, Role, Pool
3. Click button
Provisioning Process

1. Server boots iPXE via DHCP/PXE
2. iPXE gets config from Phil
3. Phil sends install image
4. Install image gets Kickstart from Phil
5. Install runs Puppet in %post
6. End of %post calls back to Collins
7. Collins triggers vlan update
8. Collins triggers monitoring/trending
9. Added to production if “all green”
Result

Fast, scalable, no hassle provisioning!

http://www.flickr.com/people/mc4army/
Hurdles
Hurdles

- PXE kickstart w/ multiple NICs
- Network set up in %post
- Virident SSD set up in %post
PXE Kickstart / Multiple NICs

Phil iPXE config

```
initrd <%= os_install_url %>/images/initrd.img
kernel <%= os_install_url %>/images/vmlinuz ip=dhcp ksdevice=${mac}
```

Phil kickstart snippet

```
# network
network --bootproto=dhcp
```
%post Network Set Up

Phil kickstart snippet

# Bond Interface: <%= bond.name %>

cat > /etc/sysconfig/network-scripts/ifcfg-<%= bond.name %> <<EoF
DEVICE=<%= bond.name %>
BONDING_OPTS=""<%= bond.options %>">
BOOTPROTO=static
IPADDR=<%= bond.address %>
NETMASK=<%= bond.netmask %>
GATEWAY=<%= bond.gateway %>
EoF
%post Virident SSD Set Up

# Start the virident daemon
/etc/init.d/vgcd start
# create a device node
mknod /dev/vgca0 b 252 0
# create a mount point
mkdir -p /var/lib/mysql
# create partitions
parted -s /dev/vgca0 mklabel msdos
parted -s /dev/vgca0 unit s mkpart primary ext2 2048 100%
# make another device node
mknod /dev/vgca0p1 b 252 1
# make the filesystem
/sbin/mkfs.xfs -f -d su=64k,sw=3 -l size=32m,su=16k /dev/vgca0p1
# create fstab entry
echo " /dev/vgca0p1 /var/lib/mysql xfs noauto 0 0" >>/etc/fstab
# create virident config
cat > /etc/sysconfig/vgcd.conf << EoF
RESCAN_MD=1
RESCAN_LVM=1
MOUNT_POINTS="/var/lib/mysql"
RESCAN_MOUNT=1
EoF
# mount the virident
mount /var/lib/mysql

mount /var/lib/mysql
Lessons Learned

- Modularity is very important
- Hardware always has issues at scale
- Use modern Bash syntax
- 4 hour burn-in is not enough
Yes, we’re hiring!

Joshua Hoffman
joshua@tumblr.com
tumblr.com/jobs
tumblr.