Configuration Management is an anti-pattern
jonah@laptop$ cvs update website
jonah@laptop$ tar zcvf website.tar.gz website
jonah@laptop$ scp website.tar.gz root@server1:/var/something/
jonah@laptop$ ssh root@server1
server1# cd /var/something
server1# mv website website-`date`
server1# tar zxf website.tar.gz
server1# /etc/init.d/website restart
server1# ^D
… Rinse, Repeat …
#!/bin/bash

BOX=$1
NEWCODE=$2

scp $NEWCODE root@$BOX:/var/something/

ssh root@$BOX "(cd /var/something ; tar zxf $NEWCODE ; /etc/init.d/tomcat restart)"
#!/bin/bash

BOX=$1
NEWWCODE=$2

scp $NEWWCODE root@$BOX:/var/something/

ssh root@$BOX "(cd /var/something ; tar zxf $NEWWCODE ; /etc/init.d/tomcat restart)"

jonah@laptop$ cvs update website
jonah@laptop$ tar zcvf website.tar.gz website
jonah@laptop$ for box in `cat serverlist\boxen.txt` ; do \
    tools/update-code.sh $box website.tar.gz \
    done
Server Install Process (2001)

- Install server in rack
- Use Mandrake Linux CD to install OS
- Run through long manual configuration checklist - some of which was eventually scripted
- Push latest code (using the earlier script)
- Add to load balancer
Server Install Process (2012+)

- Launch new Amazon AMI
- Use the current version of Amazon Linux
- Run through long manual configuration checklist - some of which was eventually scripted
- Push latest code (using the earlier script)
- Add to ELB
So, who am I?

Jonah Horowitz

Site Reliability Engineer

Soon to be at Stripe

incoming@jonahhorowitz.com
Automating Linux and Unix
Nate Campi
Kirk Bauer

SECOND EDITION

Nate Campi and Kirk Bauer

@jonahhorowitz
CFEngine (2.x) was great... for its time

Before CFEngine

• Time to provision a new server: 1 Day
• Chance a mistake was made: 50/50
• Percentage of fleet we understood: 70
CFEngine (2.x) was great... for its time

Before CFEngine
- Time to provision a new server: 1 Day
- Chance a mistake was made: 50/50
- Percentage of fleet we understood: 70

After CFEngine 2
- Time to provision a new server: 1 hour
- Chance a mistake was made: 1%
- Percentage of fleet we understood: 99
Puppet

Chef

Ansible

Salt

Red Hat
What sucks about Config Management?
What sucks about Config Management?

Bad Option #1
Ops owns all configuration management

Bad Option #2
Ops doesn’t own all configuration management
Broken/Buggy/Out-of-Sync Deployments
Broken/Buggy/Out-of-Sync Deployments

That one server...
Release Engineering
Still Sucks
What’s the alternative?
What’s the alternative?
What’s the alternative?
Let’s walk through that again, slowly
- Base or Foundation AMI
  - Security patches
  - Infrastructure Packages (monitoring, logging, etc)
• Your application package and its dependencies
Tools Required

• Package Build System (Gradle)
• Image Build System (Aminator/Bakery/Docker/Packer)
• Deployment System (Spinnaker/Terraform/CloudFormation)
• Service Discovery (Eureka/Zookeeper/ELBs/DNS?/Swarm/Kubernetes)
• Dynamic Configuration (Feature Flags/Fast Properties)
Benefits
Benefits

- Simpler Operations
Benefits

- Continuous Deployments
Benefits

• Faster startup times
  • Horizontal/Auto-scaling
  • Instance Failure
  • Chaos Monkey
  • Cloud Reboots
Benefits

- Configuration in-sync / no “cruft” / always a known state
Benefits

- Same application code in Dev/Test/Prod
Benefits

• Easier to respond to security threats
Benefits

• Multi-region operations
Benefits

- That one server... sticks out like a sore thumb
Release Strategies

• Rolling Release
• Blue/Green Releases
Caveats
Oh, that database thing...
Jonah Horowitz
Site Reliability Engineer
@jonahhorowitz
incoming@jonahhorowitz.com
https://jonahhorowitz.com/