From Zero to Cloud in 90 Days with Chef

Created by Brad Knowles

Based on materials from Chef Fundamentals OPS150-04.01 by Opscode, Inc.

Speaker

- Name: Brad Knowles
- Company: ihiji, Inc.
- Experience:
  - System Administrator/Engineer/Consultant: 20+ years
  - Chef user: since 2011-09-14
- Contact
  - E-mail: bknowles@ihiji.com
  - Twitter: @bradknowles
  - github: bknowles

What does ihiji do?

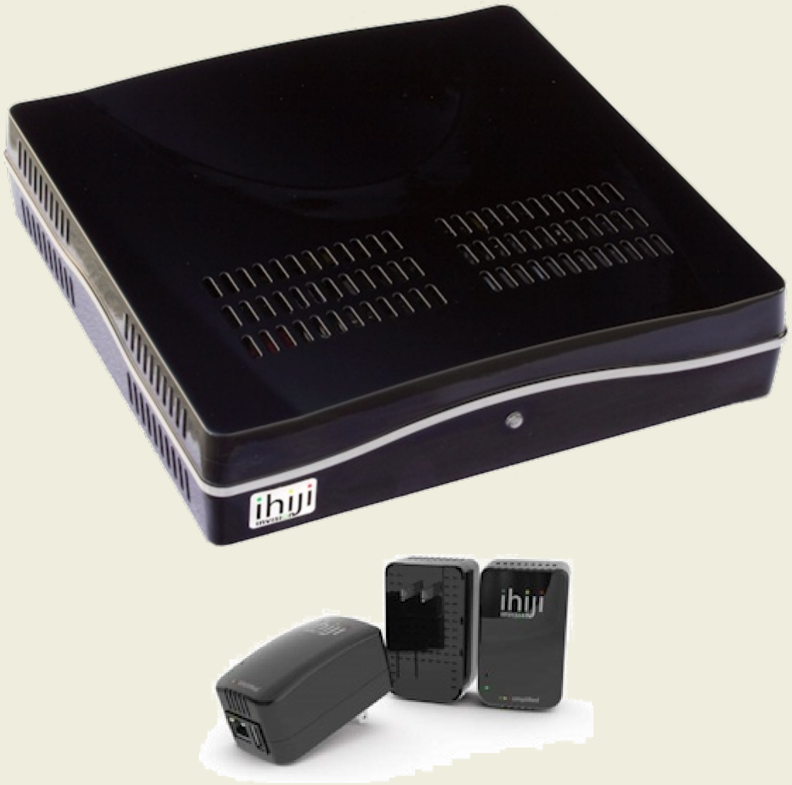
ihiji invision is a web based solution that provides a secure gateway for residential electronics systems contractors & integrators to remotely monitor, service and maintain client's in-home electronic systems, including but not limited to, audio & video equipment, network devices, etc....

This comprises an on-premise appliance and back-end supporting cloud infrastructure with web portal for dealers to remotely monitor and manage equipment that is installed at the client site

What Can Be Monitored?

- Any IP Device
  - Ping
  - Well known ports and services (e.g., HTTP, FTP, Telnet, SSH, etc...)
  - Any device supporting SNMP
  - Control4 devices
- Any RS-232, Zigbee, AMX AxLink or Crestron Cresnet Device
- Custom monitoring modules can be written to support any IP based AV device

What Do The Appliances Look Like?



Problem

Company is growing fast, but old ad-hoc hand-built infrastructure can't scale nor can it be (easily) made fault-resilient

Problem

Company is growing fast, but old ad-hoc hand-built infrastructure can't scale nor can it be (easily) made fault-resilient

By Ferdinand Reus from Arnhem, Holland (Safari Uploaded by mangostar) [CC-BY-SA-2.0], via Wikimedia Commons

Solution

- Rebuild from scratch
- In the cloud
- Must be scalable
- Must perform well
- Must be fault-resilient
- Must be very competitive re: cost/performance
- Using repeatable automated infrastructure management systems

Introducing Chef

Chef Can Help

Chef is designed to help manage this kind of complexity. You may have met already!

- Configuration management tool
- Systems integration framework
- API for infrastructure management

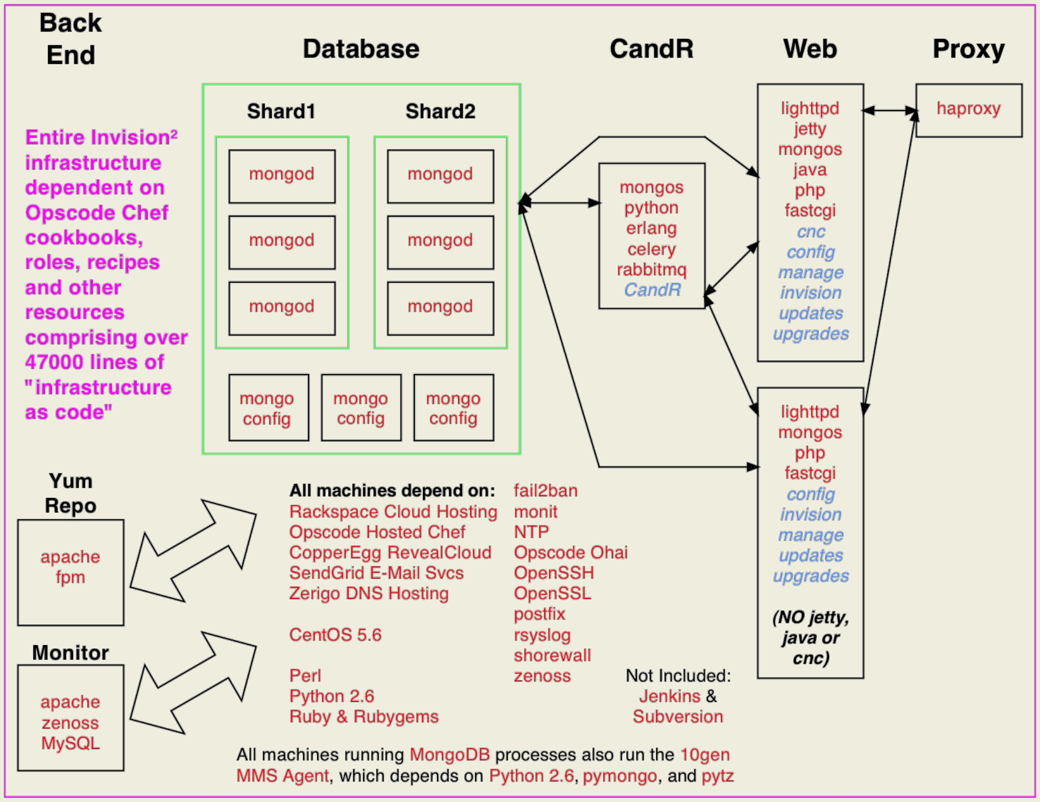
Problem

- I knew chef existed
  - But that was about it
- I had little prior exposure to CM tools
  - Only ever as a user of their services
  - Never as an engineer who implemented them
- I am **not** a developer

On the Plus Side

- I did have 20+ years of experience as a Unix System Engineer/Consultant
- I knew and trusted Matt Ray from his days at Zenoss
- Matt convinced us both that I could do this job with some training -- and some hands-on experience

Solution Diagram



Back End

Entire Invision² infrastructure dependent on Opscode Chef cookbooks, roles, recipes and other resources comprising over 47000 lines of "infrastructure as code"

Database

Shard1

mongod

mongod

mongod

Shard2

mongod

mongod

mongod

mongo config

mongo config

mongo config

CandR

mongos
python
erlang
celery
rabbitmq
CandR

Web

lighttpd
jetty
mongos
java
php
fastcgi
cnc
config
manage
invision
updates
upgrades

lighttpd
mongos
php
fastcgi
config
invision
manage
updates
upgrades

(NO jetty, java or cnc)

Proxy

haproxy

Yum Repo

apache
fpm

Monitor

apache
zenoss
MySQL

All machines depend on:
Rackspace Cloud Hosting
Opscode Hosted Chef
CopperEgg RevealCloud
SendGrid E-Mail Svcs
Zerigo DNS Hosting

CentOS 5.6

Perl
Python 2.6
Ruby & Rubygems

fail2ban
monit
NTP
Opscode Ohai
OpenSSH
OpenSSL
postfix
rsyslog
shorewall
zenoss

Not Included:
Jenkins &
Subversion

All machines running MongoDB processes also run the 10gen MMS Agent, which depends on Python 2.6, pymongo, and pytz

Solution Logos

the rackspace cloud
OPSCODE
copperegg
SendGrid
Email Delivery. Simplified.
ZERIGO
CentOS
Chef
shorewall
iptables made easy
OpenSSH
KEEPING YOUR COMMUNIQUÉS SECRET
RSyslog
POSTFIX
MONIT
OpenSSL
Network
Time
Protocol
python
pear
php
Java
ERLANG
PROGRAMMING REPUBLIC OF PERL
10gen
mongoDB
jetty://
LIGHTTPD
fly light.
HA
PROXY
RabbitMQ
Messaging that just works
The Apache Software Foundation
http://www.apache.org/
Zenoss
MySQL
NET-SNMP
Jenkins
SUBVERSION
ihiji

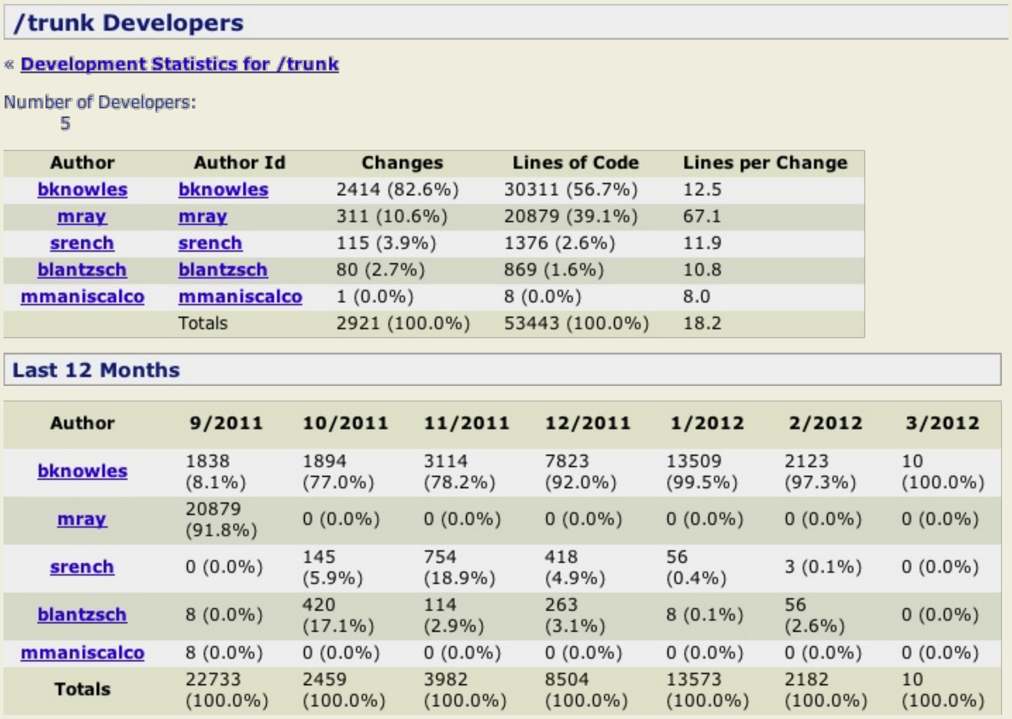
How Did We Get There?

- I Started work on August 16th, 2011
- In-depth interviews with management regarding current system design
- Reverse-engineer current systems
- Discover what needed to change and what could remain
- Working from first principles, deliver design documents:
    - High Level Architecture (40k foot)
    - Mid Level Services (10k foot)
    - Detail Level (one service @ 5k foot)
    - Project Goals
    - Rules of Thumb for Scalability
    - Next-generation Services and Service Names
- Use a couple of cloud instances to do some early testing

Training & Consulting

- Chef Fundamentals with Matt Ray
  - September 13th, 2011
- Rapid Prototype Infrastructure
  - Afternoon/Evening of September 13th, 2011
  - My first commit was on September 14th, 2011
  - Consulting complete by evening of September 15th, 2011

Developers



/trunk Developers

« Development Statistics for /trunk

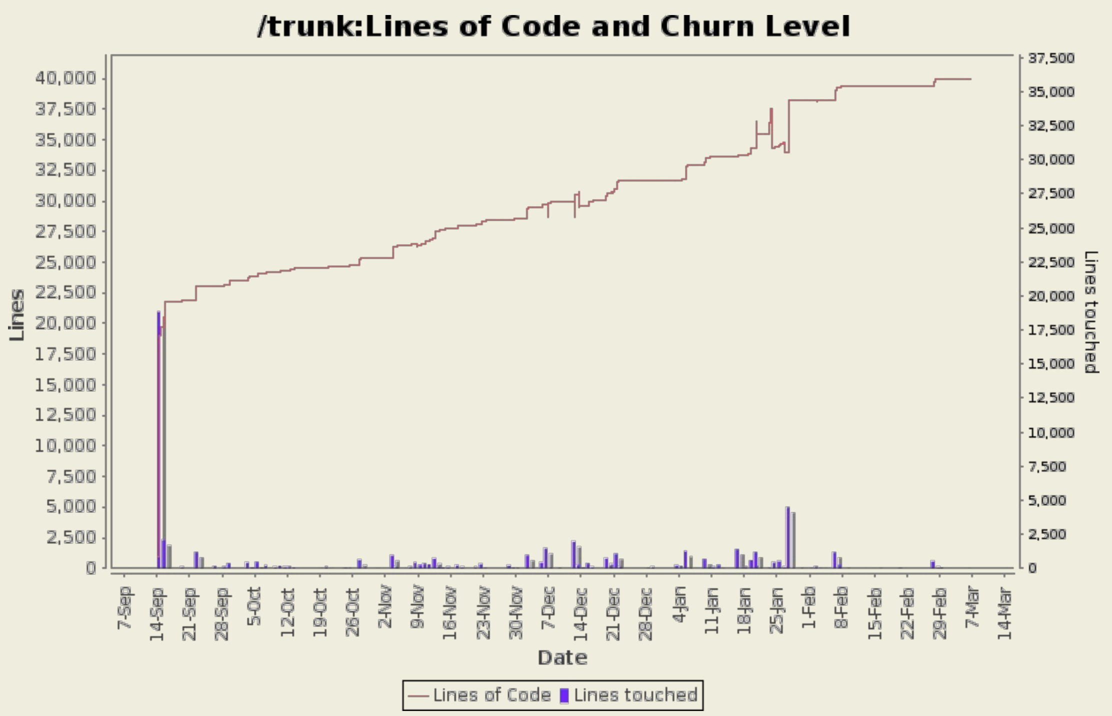
Number of Developers:
5

Author	Author Id	Changes	Lines of Code	Lines per Change
bknowles	bknowles	2414 (82.6%)	30311 (56.7%)	12.5
mray	mray	311 (10.6%)	20879 (39.1%)	67.1
srench	srench	115 (3.9%)	1376 (2.6%)	11.9
blantzsch	blantzsch	80 (2.7%)	869 (1.6%)	10.8
mmaniscalco	mmaniscalco	1 (0.0%)	8 (0.0%)	8.0
	Totals	2921 (100.0%)	53443 (100.0%)	18.2

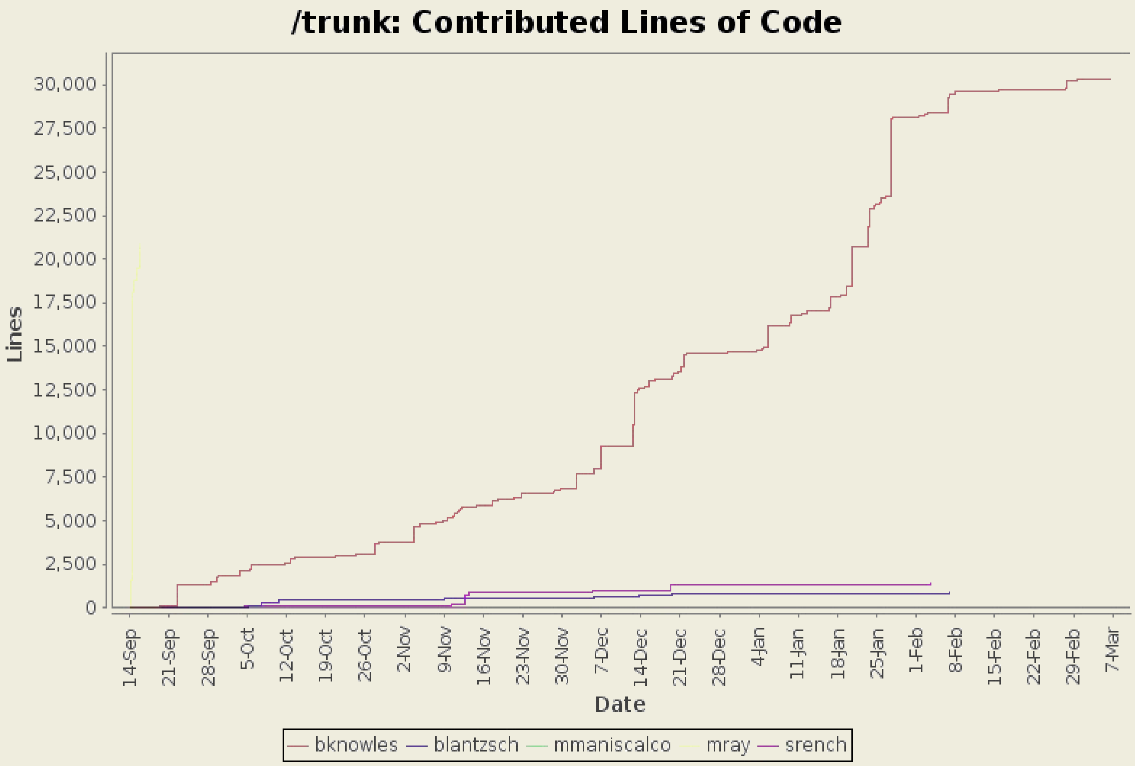
Last 12 Months

Author	9/2011	10/2011	11/2011	12/2011	1/2012	2/2012	3/2012
bknowles	1838 (8.1%)	1894 (77.0%)	3114 (78.2%)	7823 (92.0%)	13509 (99.5%)	2123 (97.3%)	10 (100.0%)
mray	20879 (91.8%)	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)
srench	0 (0.0%)	145 (5.9%)	754 (18.9%)	418 (4.9%)	56 (0.4%)	3 (0.1%)	0 (0.0%)
blantzsch	8 (0.0%)	420 (17.1%)	114 (2.9%)	263 (3.1%)	8 (0.1%)	56 (2.6%)	0 (0.0%)
mmaniscalco	8 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)
Totals	22733 (100.0%)	2459 (100.0%)	3982 (100.0%)	8504 (100.0%)	13573 (100.0%)	2182 (100.0%)	10 (100.0%)

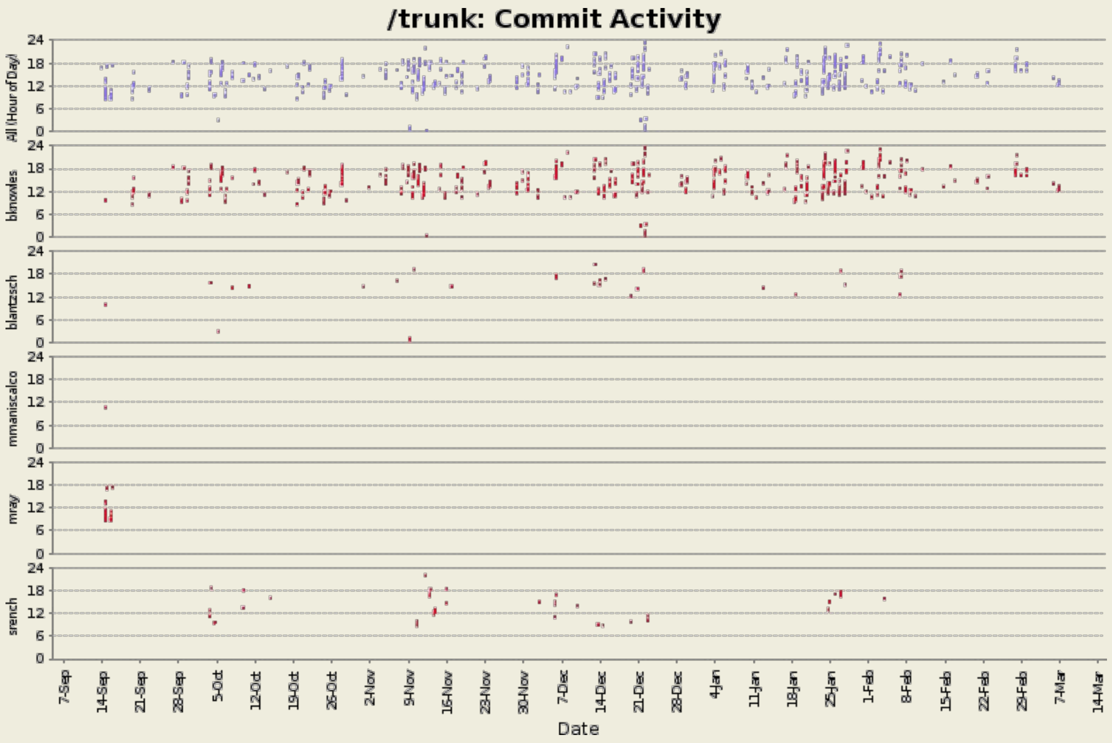
Commit Timeline



Contributed Lines of Code



Commit Heat Map



Patterns & Anti-Patterns

1. Make It Work, Then Make It Pretty
2. Publish Early, Publish Often
3. Use Git
4. Use Github
5. One Github Repo per Chef Cookbook
6. Use All Available Resources for Help
7. Use Code Reviews
8. Use Automated Testing
9. Use Continuous Integration/Deployment
10. Monitoring Is Just Continuous System/Service Testing
11. Logging is Monitoring Too
12. ChaosMonkey Is The (Current) Holy Grail

Make It Work, Then Make It Pretty

- Doing part one without part two is an **anti-pattern**
  - You will have jury-rigged a landmine in your infrastructure
  - Then foreverafter, you must always walk on eggshells around it
  - In twenty ${time-units} you won't remember what you did, why, or how and it **WILL** blow up in your face

Publish Early, Publish Often

- Helps encourage both parts of #1
  - Publishing something ugly is a negative feedback mechanism
- Encourages working in smaller increments that are more digestible

Use Git

- All tools for Chef that assist with SCM workflow will assume git
  - Probably won't work with anything else
- As a Distributed VCS, git encourages frequent small commits, where tags and branches are trivially easy
- WhyGitIsBetterThanX

Use Github

- Github is Facebook+LinkedIn for Developers & DevOps
  - It's how you show prospective employers what work you've done
  - It's how prospective employers show you what kind of work you might be doing
- Github is the largest Git community
  - Github works very hard to make it as easy as possible for people and groups to collaborate with each other
  - Git+Github is one of the most powerful tool combinations in your toolbox

One Github Repo per Chef Cookbook

- With Git and Github, collaboration occurs primarily at the repo level -- forking, sharing, remotes, committing, etc...
  - Each Chef cookbook should be largely standalone
  - Therefore, each Chef cookbook should be a separate repo
  - Can combine multiple repos together with Librarian, Braid, or git submodules

Use All Available Resources for Help

- No one can be an expert on everything
  - Whatever you're trying to do, someone else has probably already tried it
  - Don't stubbornly insist on re-inventing the wheel
- Use
  - Wiki: http://wiki.opscode.com/
  - Bug Tracker: http://tickets.opscode.com/
  - Community Cookbooks: http://community.opscode.com/
  - Community Mailing list: chef@lists.opscode.com
  - irc: #chef on irc.freenode.net
  - Hosted Chef:
    - Free for first five clients
    - Free access to Opscode Support: support@opscode.com

Use Code Reviews

- 80-90% of the time, I catch my code errors on proper inspection when I try to explain the code to someone else
- 80-90% of the remainder, my code errors are caught on inspection by the person who reviews my code
- If I skip code reviews, I lose the 95-99% error catch rate that having a programming partner would bring to the table

Use Automated Testing and Test Driven Development

- Chef minitest and Rspec are improving
  - Still beta quality code
  - Still not as well documented as they should be.
- Cucumber-chef is currently dependent on Amazon EC2
  - Needs more work (e.g., needs to support other cloud platforms)
  - Needs better documentation
  - Test-Driven Infrastructure with Chef is supposed to just be a "taste"
- Follow Stephen Nelson-Smith (@LordCope)
  - For all aspects of TDD/BDD, not just cucumber-chef

Use Continuous Integration/Deployment

- We use Jenkins but the tool you use is less important than doing CI in some fashion
  - chef-jenkins is the tool to use Jenkins to drive continuous deployment and synchronization of your Chef Environments from a git repository
- Chef is all about automating the management of the configuration of your infrastructure, and CI tools like Jenkins dovetail naturally with that

Monitoring Is Just Continuous System/Service Testing

- You need both internal and external monitoring
  - We use Zenoss and monit for internal monitoring
  - We use CopperEgg RevealCloud for external monitoring
- You need to monitor the systems, the applications, and the services
  - We use Zenoss and RevealCloud to monitor our systems
  - We use monit to monitor our applications
  - We use Zenoss to monitor our services

Logging is Monitoring Too

- Logging is just another way to monitor what is going on with your systems & applications
  - Forwarding & Gathering -- syslog-based (e.g., rsyslog & syslog-ng) and agent-based (incl. graylog agent, logstash, Splunk forwarders, etc...)
  - Analytics (e.g., graylog2, logstash, Splunk, etc...)
- Logging and log analysis is just as critical (if not more so) as monitoring
  - System, Service, and Application monitoring tell you **what** is happening
  - Log processing is more likely to be able to tell you **how** and **why**

ChaosMonkey Is The (Current) Holy Grail

- Failure **will** happen -- It's not a matter of **if**, but **when**
  - What if you could have some control over when failure occurs and how your systems respond?

Career Lessons

1. Know Yourself
2. Tools First
3. Value of Education, Dogfood, & #1 Revisited
4. DevOps vs. Politics
5. Family vs. Job, Value of Networking, & Rule of Three
6. More Value of Communities & Preparing to Speak/Teach
7. Location & Localization vs. Culturation & Distance
8. If You Don't Have the Proof, "It" Doesn't Exist
9. Choose Your Employers and Co-Workers Wisely

Know Yourself

- What is it that allows you to stay at a suboptimal job?
  - The People?
  - The Work?
  - The Money?
- For Me
  - People >> Work >> Money

Tools First

Quote:

```
"If I had eight hours to chop down a tree, I'd spend
six hours sharpening my axe.  Give me six hours to
chop down a tree and I will spend the first four
sharpening the axe." -- Abraham Lincoln
```

Value of Education, Dogfood, & #1 Revisited

- Humans tend to not value something until it is gone or they are no longer there
- Eating your own dogfood is a sign that you are committed to succeed and you will bet your own business on your product
  - **However** -- Don't lose sight of who the real customer(s) is (are)

DevOps vs. Politics

- For DevOps to be successful, you **MUST** have complete buy-in from stem-to-stern
- As soon as a single person isn't 100% committed to the teamwork required, the whole corporate train starts going off the rails

Family vs. Job, Value of Networking, & Rule of Three

- Not Company vs. Home Life, but Company **AS** Family, vs. Company **AS** Job
  - If the company pitches itself as a family, but treats everyone as employees, you are on a path for disaster
- The Collective is Much More Resourceful and Much Smarter than the Individual
- However Long You Honestly Think it will Take, Multiply by Three

More Value of Communities & Preparing to Speak/Teach

- Experienced Smart People doing Intelligent Design with Open Source can lead to Novel Work
  - Which can lead to invitations to speak at conferences
  - Which can lead to meeting lots of other very experienced and smart people
- Preparing to speak at a conference or teach a tutorial tends to force me to really dig deep to completely and totally understand the material

Location & Localization vs. Culturation & Distance

- Signficant physical distance makes work/life relationships hard
- Significant cultural and/or linguistic distance makes them harder
- Combine with economic downturn, and you may have the Perfect Storm

If You Don't Have the Proof, "It" Doesn't Exist

- You need to take a baseline when things are working well
  - If/when you need that baseline the time when you could have gotten it will have long since passed.

Choose Your Employers and Co-Workers Wisely

- Hiring = Talent Acquisition
  - Work for the kind of people you would want as friends regardless
  - Same for your co-workers
- If You Are Hiring
  - What kind of people you would want with you
  - If you were stuck in an elevator with them
  - ... for 24 hours
- Acquire the Right Talent
  - With the right personalities
  - With the right approaches to problem solving

Questions?

Contact

- E-mail: bknowles@ihiji.com
- Twitter: @bradknowles
- github: bknowles