Operations Level Up

Mandi Walls
Velocity Santa Clara
June 18, 2013
• Mandi Walls
• @lnxchk
• Consultant at Opscode
• My 6th Velocity
who are you?

• Ops?
• Devs?
• Management?
• Other functions:
  • sales, marketing, foosball, TPS reporting?
What is this madness

- Part call-to-arms
- Part techie hands-on things
- Part revival meeting
Why Do I Care?
Future of Operations

http://www.flickr.com/photos/x-ray_delta_one/5871906878/
Evolution of a Practice

- Craft Stage
- Commercial Stage
- Engineering Stage

http://www.flickr.com/photos/thaisfraga182/5285413020/sizes/z/in/photostream/
• Hand crafted artisanal organic free range bespoke systems

• Lots of personal heroics

• Land of the BOFHs
Commercial Stage

- Folklore written down
- Standard procedures emerge
- Training begins to occur
Engineering Stage

• Application of scientific principles
• Measurement
• Experimentation towards greater efficiency
New Workflows

- Visibility and planning
- Version control and code review
- Testing, testing, and more testing
- Metrics collection and interpretation

Basically, borrow some stuff from Dev

http://websites-development.com/sites/default/files/git_branch_strategy.png
New Goals

- Transparency - are we working on the right things
- Reliability - can we keep it running
- Resiliency - can we rebuild it? Do we have the technology?
- Correctness - are we sure it’s doing what we want it to do
New Goals

- Transparency - are we working on the right things?
- Reliability - can we keep it running?
- Resiliency - can we rebuild it?
- Building Trust
- Correctness - are we sure it’s doing what we want it to do
More than keeping the lights on.
I don’t write code, I’m a sysadmin
I don’t write code, I’m a sysadmin

I have to spend all my time fixing dumb things
I don’t write code, I’m a sysadmin

I have to spend all my time fixing dumb things

This takes too much time.
I don’t write code, I’m a sysadmin

I have to spend all my time fixing dumb things

This takes too much time.

I’m faster if I don’t have to talk to anyone about what’s going on.
I don’t write code, I’m a sysadmin

I have to spend all my time fixing dumb things

These tools are too hard to learn.

I will write my own thing.

I’m faster if I don’t have to talk to anyone about what’s going on.
So, some things to work on

• Some tools for mitigating risk
• Some processes and tips for making the right thing the easy thing
• Increase efficiency, learn some stuff, reevaluate your own work
• Don’t be afraid of borrowing from other disciplines
Eating Our Veggies

http://www.packriveryaks.com/
Opportunity Cost

The value of the things you could be doing while you were shaving that yak
Employability

Is This Good for the COMPANY?
Risk Vectors

- What Ops thinks of as risk
- New code, releases, tasks
- Other sources of risk
- Old products and workflows
- Unrepeatable processes
- Personal heroics

http://www.flickr.com/photos/baresone/4473290629/sizes/z/in/photostream/
Assessment of Risk

- Is your process:
  - well documented
  - repeatable
  - reliable
  - easy to do right?

http://www.flickr.com/photos/lemusgro/5494317161/sizes/z/in/photostream/
EASY TO DO RIGHT

Seriously. I’m not kidding.
Hands On

• Git and hooks for Ops
• Packaging your stuff
• Borrowing sanity checks from other places
• Basic testing without doing a lot of zomgcoding
Hands On

https://gist.github.com/lunixchk/5791121

- Packaging your stuff
- Borrowing sanity checks from other places
- Basic testing without doing a lot of zomgcoding
$ ssh -p 2222 velocity@127.0.0.1

• Login to your host using ssh
• The user is velocity
• The password is velocityconf
• Vagrant is port-forwarding ssh for you
• Login to your host using `ssh`
• The user is `velocity`
• The password is `velocityconf`
• Make sure your AWS Security Group allows inbound ssh
• You can allow ssh from your EC2 console, under “Security Groups”
If you’re using local virtual box, log in when the guest is initialized

The user is velocity

The password is velocityconf
Task 1 - Working with git

http://mattbanks.me/wp-content/uploads/2012/07/Git-Logo.png
• Distributed version control
• Everyone gets a copy
• Hub/spoke model for sharing
• Simple set up
• Easy to run a local git server
• Other offerings, like github, are pretty awesome too
You have a git server on your lab box

It’s root is in /srv/git

There is one project, bindfiles.git

velocity can use ssh:// to talk to the git server
• Directory ~velocity/bindfiles is already a git repo for you

• The remote origin is localhost

• It will use ssh to talk to the local git server

• It contains two trivial zonefiles
  • db.192
  • db.local
$ cd bindfiles
$ cat .git/config

[core]
repositoryformatversion = 0
filemode = true
bare = false
logallrefupdates = true

[remote "origin"]
fetch = +refs/heads/*:refs/remotes/origin/*
url = ssh://localhost/srv/git/bindfiles.git

[branch "master"]
remote = origin
merge = refs/heads/master
$ vi db.192

; BIND reverse data file for local loopback interface
;
$TTL 604800
@ IN SOA ns.local. root.local. ( 2Z ; Serial
604800 ; Refresh
86400 ; Retry
2419200 ; Expire
604800 ) ; Negative Cache TTL
;
@ IN NS ns.
10 IN PTR ns.local.

; also list other computers
21 IN PTR box.local.
Add a new host

Add “wat.local” with final octet 24

; also list other computers

<table>
<thead>
<tr>
<th>21</th>
<th>IN</th>
<th>PTR</th>
<th>box.local.</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>IN</td>
<td>PTR</td>
<td>wat.local.</td>
</tr>
</tbody>
</table>
$ git status

# On branch master

# Changed but not updated:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
# modified:   db.192
#
no changes added to commit (use "git add" and/or "git commit -a"
git tells you what it wants

# Changed but not updated:

# (use "git add <file>..." to update what will be committed)

# (use "git checkout -- <file>..." to discard changes in working directory)

#

# modified: db.192
$ git add db.192
$ git status

# On branch master

# Changes to be committed:

#   (use "git reset HEAD <file>..." to unstage)

#

# modified:   db.192

#
• `git add` stages your changes locally

• `git commit` will write them to your local git repository

• add your comment either inline with “-m” or `git commit` will open a buffer for you

```
$ git commit -m "this commit is awesome"
```
$ git commit

Added wat.local to reverse file

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
# modified:   db.192
[master 22371ab] Added wat.local to reverse file

1 files changed, 4 insertions(+), 0 deletions(-)

$ git status

# On branch master

# Your branch is ahead of 'origin/master' by 1 commit.

#

nothing to commit (working directory clean)
Making Good Comments

- At least explain what you did, Lucy
- If there is a ticket somewhere, add that in the comment
- If you made multiple changes, call them all out
$ git push

Counting objects: 5, done.

Compressing objects: 100% (3/3), done.

Writing objects: 100% (3/3), 335 bytes, done.

Total 3 (delta 1), reused 0 (delta 0)

To ssh://localhost/srv/git/bindfiles.git

06fa560..22371ab  master -> master
• **git push** sends your changes to the central git server

• **git pull** brings everyone else’s changes into your local repo

• Don’t hoard changes; push and pull often
• What did we forget to do?
Update the Serial!

- Lots of Operations tasks have tribal knowledge you need
- Zonefiles have a Serial that needs to be incremented when you make a change
- They are potentially outage-causing or hair-pulling problems that can be avoided
- Let’s let git remember to do that for us
• You can put hooks into your git repos

• Little tasks that happen at various steps in the process

• We can add a pre-commit hook to our bindfiles repo

• So you don’t have to remember! Saves time later! Helps junior staff!
```bash
#!/bin/bash
num=`git diff master db.192 | grep ^+ | wc | awk '{print $1}'`
if [ $num -gt 1 ] ; then
    serial=`git diff master db.192 | grep -i serial`
    if [ $? -ne 0 ] ; then
        echo "You made a change to the zone file but didn't update the Serial value"
        exit 1;
    fi
fi
```
• Rather messy, off-the-cuff example

• `git diff master db.192`

  • Looks for changes between what’s in the current master on your local repo

  • If the db.192 file has changed but the value for Serial is the same, it prints and error and exits with a non-zero return code

  • `git stops processing the commit`, saving you headaches later
Generalized pre-commit

#!/bin/bash

for i in `ls db.*` ; do

  num=`git diff master $i | grep ^+ | wc | awk '{print $1}'`
  if [ $num -gt 1 ] ; then
    serial=`git diff master $i | grep -i serial`
    if [ $? -ne 0 ] ; then
      echo "You made a change to the zone file $i but didn't update the Serial value"
      exit 1;
    fi
  fi
done

Want it? it’s in the gist!
Try it out

• Make a change to your reverse file, db.192: move wat.local to .22

• don’t update the serial, see what git does for you

• git add db.192

• git commit

You made a change to the zone file but didn't update the Serial value

Well. That was handy!

Fix the zonefile, try it again! Your commit will go through. git push it

Feeling hardcore? Update db.local to match, check it in too
What else to hook?

• Services with config checkers
• make a change to the config, run the checker in a hook
• nagios, named, apache, etc come with check tools
• Other syntax checking
• ruby, json, config management tools
What else to hook?

- Services with config checkers
- make a change to the config, run the checker in a hook
- nagios, named, apache, etc come with check tools
- Other syntax checking
- ruby, json, config management tools
Other git things

• git clone
  • gives new team members a working copy of the repo

• git branch
  • Lets you and other team members work on non-prod stuff

• Use sparingly; less complexity means more reliable results
Task 2: creating packages

• How do you get files, apps, stuff deployed on your hosts?
• `scp -r`?
• `tarballs`?
• build everything on every host, you gentoo fans?
• `crash cart? (omg)`
• Reap the benefits of what’s built in to your package manager
• Versioning
• Dependencies
• Metadata
• Build-once, install-many
• File transfer built right into stuff like yum and apt repos!
Package. All. The. Things.
Creating packages from scratch is tedious
There’s some esoteric stuff in the package managers
You really only need a few things
fpm, “f’ing package managers”!

Jordan Sissell

Creates multiple kinds of packages from various resources

https://github.com/jordansissel/fpm

It’s already installed on your lab box!
$ fpm -h

Intro:
This is fpm version 0.4.37
If you think something is wrong, it’s probably a bug! :) 
Please file these here: https://github.com/jordansissel/fpm/issues
You can find support on irc (#fpm on freenode irc) or via email with fpm-users@googlegroups.com

Usage:
fpm [OPTIONS] [ARGS] ...

Parameters:
[ARGS] ...                        Inputs to the source package type. For the 'dir' type, this is the files and
directories you want to include in the package. For others, like 'gem', it specifies the packages to download and
use as the gem input

Options:
-t OUTPUT_TYPE                the type of package you want to create (deb, rpm, solaris, etc)
-s INPUT_TYPE                 the package type to use as input (gem, rpm, python, etc)
-C CHDIR                      Change directory to here before searching for files

omg it just keeps going....
Create a package

- rpm-ify our zonefiles
- They’re in our git repo right now
- Live in /var/named for reals
- If we package them, we get versioning and other data
$ fpm -s dir -t rpm -v 1.0 --prefix=/var/named \
- n "zonefiles" --after-install \n/srv/velocity/restart_named.sh db*

- **-s dir**: we’re working with raw files rather than a gem, rpm, etc
- **-t rpm**: create an rpm package
- **-v 1.0**: first version!
- **--prefix=/var/named**: where the files will be installed
- **-n “zonefiles”**: name of the package
- **--after-install /srv/velocity/restart_named.sh**: run this after installing
- **db***: the files to be packaged
$ fpm -s dir -t rpm -v 1.0 --prefix=/var/named \\   -n "zonefiles" --after-install \\   /srv/velocity/restart_named.sh db* \\

Created rpm {:path=>"zonefiles-1.0-1.x86_64.rpm"}

$ rpm -qpl zonefiles-1.0-1.x86_64.rpm

/var/named/db.192
/var/named/db.local
• Nice!

• Now we can install it
$ sudo rpm -ihv zonefiles-1.0-1.x86_64.rpm

Preparing... #..................................................... [100%]

1:zonefiles #..................................................... [100%]

Stopping named: .[ OK ]

Starting named: [ OK ]

$ dig @localhost -x 192.168.1.22

;; QUESTION SECTION:

;22.1.168.192.in-addr.arpa. IN PTR

;; ANSWER SECTION:

22.1.168.192.in-addr.arpa. 604800 IN PTR wat.local.
Put the bits together

- Your zonefiles are in a git repo
- The repo has syntax and error checking pre-commit hooks
- The repo can also have packaging and deploy post-commit hooks
- Smooth the process, make the right way the easiest way
Task 3: Testing

- Lots of work in the dev space
- TDD, BDD, test, test
- Write tests first, prove they fail, write code to make them pass
- More risk reduction
- Looks scary

http://www.flickr.com/photos/nobleup/3995733415/sizes/z/in/photostream/
• So, there’s a DNS server on the lab box
• What else is here?
• Nagios server!
What can we borrow?

- Nagios plugins!
- Extensive set of checks for all sorts of services
- Usable from the command line
$ ls /usr/lib64/nagios/plugins

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>check_breeze</td>
<td>check_game</td>
<td>check_mrtgtraf</td>
<td>check_overcr</td>
<td>check_swap</td>
<td></td>
<td></td>
</tr>
<tr>
<td>check_by_ssh</td>
<td>check_hpjd</td>
<td>check_mysql</td>
<td>check_psql</td>
<td>check_tcp</td>
<td></td>
<td></td>
</tr>
<tr>
<td>check_clamd</td>
<td>check_http</td>
<td>check_mysql_query</td>
<td>check_ping</td>
<td>check_time</td>
<td></td>
<td></td>
</tr>
<tr>
<td>check_cluster</td>
<td>check_icmp</td>
<td>check_nagios</td>
<td>check_pop</td>
<td>check_ua</td>
<td></td>
<td></td>
</tr>
<tr>
<td>check_dhcp</td>
<td>check_ide_smart</td>
<td>check_ntp</td>
<td>check_procs</td>
<td>check_users</td>
<td></td>
<td></td>
</tr>
<tr>
<td>check_dig</td>
<td>check_imap</td>
<td>check_ntps</td>
<td>check_real</td>
<td>check_users</td>
<td></td>
<td></td>
</tr>
<tr>
<td>check_disk</td>
<td>check_ircd</td>
<td>check_nrpe</td>
<td>check_rpc</td>
<td>check_users</td>
<td></td>
<td></td>
</tr>
<tr>
<td>check_disk_smb</td>
<td>check_jabber</td>
<td>check_nt</td>
<td>check_sensors</td>
<td>check_users</td>
<td></td>
<td></td>
</tr>
<tr>
<td>check_dns</td>
<td>check ldap</td>
<td>check_ntp_pl</td>
<td>check_smtp</td>
<td>check_smtp</td>
<td></td>
<td></td>
</tr>
<tr>
<td>check_dummy</td>
<td>check ldaps</td>
<td>check_ntp</td>
<td>check_smtp</td>
<td>check_smtp</td>
<td></td>
<td></td>
</tr>
<tr>
<td>check_file_age</td>
<td>check load</td>
<td>check_ntp_pl</td>
<td>check_smtp</td>
<td>check_smtp</td>
<td></td>
<td></td>
</tr>
<tr>
<td>check_flexlm</td>
<td>check log</td>
<td>check_ntp_time</td>
<td>check_ssh</td>
<td>check_ssh</td>
<td></td>
<td></td>
</tr>
<tr>
<td>check_ftping</td>
<td>check mailq</td>
<td>check_nwstat</td>
<td>check_ssh</td>
<td>check_ssh</td>
<td></td>
<td></td>
</tr>
<tr>
<td>check_ftps</td>
<td>check mrtg</td>
<td>check_oracle</td>
<td>check_ssh</td>
<td>check_ssh</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
$ ls /usr/lib64/nagios/plugins

check_breeze check_game check_mrtgtraf check_overcr check_swap
check_by_ssh check_hpjd check_mysql check_pqsql check_tcp
check_clamd check_http check_mysql_query check_ping check_time
check_cluster check_icmp check_nagios check_pop check_udp
check_dhcp check_ide_smart check_nn' check_rpc check_wave
check_dig check imap check_nrpe check_sensors eventhandlers
check_disk check_irod check_nt check_simap negate
check_disk_smb check_jabber check_ntp check_smtp urlize
check_dns check_ldap check_ntp check_spop utils.pm
check_dummy check_ldaps check_ntp_peer check_snmp utils.sh
check_file_age check_load check_ntp.pl check_ssh
check_flexlm check_log check_ntp_time check_spop utils.sh
check_fping check_mailq check_nwstat check_ssmtp
check_ftp check_mrtg check_oracle check_ssmp
• We can use these plugins to test out what we’re doing
• Don’t require any additional frameworks or scary things
• Many of them work just fine over the network, too
$ /usr/lib64/nagios/plugins/check_dns -h

check_dns v1.4.16 (nagios-plugins 1.4.16)
Copyright (c) 1999 Ethan Galstad <nagios@nagios.org>
Copyright (c) 2000-2008 Nagios Plugin Development Team
   <nagiosplug-devel@lists.sourceforge.net>

This plugin uses the nslookup program to obtain the IP address for the given host/domain query. An optional DNS server to use may be specified. If no DNS server is specified, the default server(s) specified in /etc/resolv.conf will be used.

Usage:
check_dns -H host [-s server] [-a expected-address] [-A] [-t timeout] [-w warn] [-c crit]
When check_dns is ok

$ check_dns -H box.local -s 127.0.0.1 -a 192.168.1.21

DNS OK: 0.004 seconds response time. box.local returns 192.168.1.21 | time=0.004142s; ;; 0.000000

$ echo $? 

0
$ check_dns -H box.local -s 127.0.0.1 -a 192.168.1.22

DNS CRITICAL - expected '192.168.1.22' but got '192.168.1.21'

$ echo $?

2
• Now we have a way to test our changes
• Behaves in a predictable way
• Now let’s add one more component: a test harness
Task 4: testing with bats

- Bash Automated Testing System
- Like all good tools, bats is impossible to google
- https://github.com/sstephenson/bats

http://www.etsy.com/shop/theitsybitsyspider

http://www.flickr.com/photos/37539972@N06/3980094382/sizes/z/in/photostream/
What the bats

• Allows you to test that UNIX programs do what you expect
• Write stuff in bash to test other system commands
• Easy to get hold of return codes, output
• Let’s see an example: checking the nagios configs
Using bats

$ bats /srv/velocity/nagios.bats

1..1
ok 1 nagios is all good
Using bats

```
$ bats /srv/velocity/nagios.bats

1..1
ok 1 nagios is all good
```
Using bats

$ bats /srv/velocity/nagios.bats

1..1  Ran one test
ok 1 nagios is all good
$ bats /srv/velocity/nagios.bats

1..1  Ran one test
ok 1 nagios is all good
Using bats

$ bats /srv/velocity/nagios.bats

1..1  Ran one test
ok 1 nagios is all good

All good!
$ cat /srv/velocity/nagios.bats

#!/usr/bin/env bats

@test "nagios is all good" {
    result="$(sudo service nagios checkconfig)"
    [ "$?" -eq 0 ]
}

76 Tuesday, June 18, 13
#!/usr/bin/env bats

@test "nagios is all good" {
    result="$(sudo service nagios checkconfig)"
    [ " $? " -eq 0 ]
}

#!/usr/bin/env bats

@test "nagios is all good" {
    result="$(sudo service nagios checkconfig)"
    [ " $? " -eq 0 ]
}

Run a system command!
```bash
#!/usr/bin/env bats

test "nagios is all good" {
    result="$(sudo service nagios checkconfig)"
    [ "$?" -eq 0 ]
}
```
$ cat /srv/velocity/nagios.bats

#!/usr/bin/env bats
@test "nagios is all good" {
result="$(sudo service nagios checkconfig)"
[ "$?" -eq 0 ]
}

Run a system command!

Check the return code!
$ cat /srv/velocity/nagios.bats

#!/usr/bin/env bats
@test "nagios is all good" {
    result="$(sudo service nagios checkconfig)"
    [ "$?" -eq 0 ]
}
```bash
#!/usr/bin/env bats

@test "nagios is all good" {
    result="$(sudo service nagios checkconfig)"
    [-eq 0 ]
}
```

Run a system command!

Also grabs output, but we don’t need that here

Check the return code!
We can do this!

http://www.flickr.com/photos/usnationalarchives/3678696585/
Let’s say we have a task to add `velocity.local / 192.168.1.24` to our zone

1. write a new bats test so it fails
2. make the changes to our zonefiles
   1. check them in with git, build our package, install it
3. re-run our bats test to prove it passes
4. Win.
$ cd ~/bindfiles
$ mkdir tests
$ vi tests/velocity.bats

• What to test?
• Forward lookup : velocity.bats
• Backwards lookup : 192.168.1.24
• Use the check_dns nagios plugin
#!/usr/bin/env bats

@test "velocity.local forward lookup is good" {
    /usr/lib64/nagios/plugins/check_dns -H velocity.local -s 127.0.0.1 -a 192.168.1.24
    [ "$?" -eq 0 ]
}

test "velocity.local reverse lookup is good" {
    /usr/lib64/nagios/plugins/check_dns -H 192.168.1.24 -s 127.0.0.1 -a velocity.local.
    [ "$?" -eq 0 ]
}
$ bats testing/velocity.bats

1..2

not ok 1 velocity.local forward lookup is good

# /home/velocity/bindfiles/tests/velocity.bats:4

#    Domain velocity.local was not found by the server

not ok 2 velocity.local reverse lookup is good

# /home/velocity/bindfiles/tests/velocity.bats:9

#    Domain 192.168.1.24 was not found by the server
Check the tests in

$ git add tests
$ git commit -m "Adding bats testing"
$ git push

- Matter of preference, checking the new tests directory in first
2: Make your DNS changes

$ vi db.192

... ; also list other computers

21  IN  PTR  box.local.
22  IN  PTR  wat.local.
24  IN  PTR  velocity.local.
2: Make your DNS changes

$ vi db.192

... ; also list other computers

21 IN PTR box.local.
22 IN PTR wat.local.
24 IN PTR velocity.local.

Update your Serial too!
Or don’t; pre-commit will know!
$ vi db.local

....

;also list other computers

<table>
<thead>
<tr>
<th>Box</th>
<th>IN</th>
<th>A</th>
<th>192.168.1.21</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wat</td>
<td>IN</td>
<td>A</td>
<td>192.168.1.22</td>
</tr>
<tr>
<td>Velocity</td>
<td>IN</td>
<td>A</td>
<td>192.168.1.24</td>
</tr>
</tbody>
</table>
$ vi db.local

....

; also list other computers

<table>
<thead>
<tr>
<th>box</th>
<th>IN</th>
<th>A</th>
<th>192.168.1.21</th>
</tr>
</thead>
<tbody>
<tr>
<td>wat</td>
<td>IN</td>
<td>A</td>
<td>192.168.1.22</td>
</tr>
<tr>
<td>velocity</td>
<td>IN</td>
<td>A</td>
<td>192.168.1.24</td>
</tr>
</tbody>
</table>
$ git add db.*
$ git commit -m "Adding velocity.local"
$ git push
3: Build the zonefiles rpm

```bash
$ fpm -s dir -t rpm -v 1.2 --prefix=/var/named -n "zonefiles" --after-install /srv/velocity/restart_named.sh db*

Created rpm
{:path=>"zonefiles-1.2-1.x86_64.rpm"}

$ sudo rpm -Uhv zonefiles-1.2-1.x86_64.rpm

Preparing... #--------------------------------------------------- [100%]
1:zonefiles #--------------------------------------------------- [100%]
Stopping named: [ OK ]
Starting named: [ OK ]
```
4: Re-run your tests

$ bats tests/velocity.bats

1..2

ok 1 velocity.local forward lookup is good
ok 2 velocity.local reverse lookup is good
How about that

- We wrote a test first, before making changes
- The test fails before the changes are made!
- Make the changes
- Run the test again, it succeeds!

http://www.coolfbcovers.net/wp-content/uploads/2013/05/Success-Kid.jpg
Our Goals: Transparency

- Are we working on something that adds value?
Our Goals: Reliability

- Does our new process keep things running?
Our Goals: Resiliency

• Does our new process make it easy to rebuild, recover, scale?
Our Goals: Correctness

- Does our new process ensure that the work we’re doing is correct?
Building from here

http://www.flickr.com/photos/kalmyket/691478431/sizes/z/in/photostream/
Configuration Management

• USE IT

• Yes, there is a learning curve

• Yes, it will be worth it in the long run
• Big Data and Business Intelligence analytics are everywhere

• Logs have value to Ops, Dev, BizDev, Marketing, etc

• Tools like logstash help you put together better pipeline

• http://logstash.net
We’ve seen lots of talks about metrics at Velocity

- statsd
- graphite

Learn you some R

Metrics also have value to groups outside of Ops
• Monitor all, alert few

• Check out Monitorama, a conference and emerging community around monitoring

• http://monitorama.com
• Do more real-world testing
• vagrant, vbox, cloud providers
• Make Dev and QA really look like prod
Fostering innovation by making resources available
Takeaways

- Reliable, repeatable processes
- Make stuff easy to do right
- Reduce risk of mistakes, misunderstandings
- Reduce the need for personal heroics

http://www.flickr.com/photos/ginnerobot/2877212845/sizes/z/in/photostream/
• Thanks for your kind attention

• Please keep the conversation going with your teams