10 deploys per day
Dev & ops cooperation at Flickr

John Allspaw & Paul Hammond
Velocity 2009
3 billion photos

40,000 photos per second

http://flickr.com/photos/jimmyroq/415506736/
Dev versus Ops
“It’s not my machines, it’s your code!”
“It’s not my code, it’s your machines!”
Spock

Little bit weird
Sits closer to the boss
Thinks too hard

Scotty

Pulls levers & turns knobs
Easily excited
Yells a lot in emergencies
Says “No” all the time
Afraid that new fangled things will break the site
Fingerpointy
Ops stereotype

Because the site breaks unexpectedly

Because no one tells them anything

Because they say “NO” all the time
Traditional thinking

Dev’s job is to add new features
Ops’ job is to keep the site stable and fast
Ops’ job is **NOT** to keep the site stable and fast
Ops’ job is to **enable** the business

*(this is dev’s job too)*
The business requires change
But change is the root cause of most outages!
Discourage change in the interests of stability or Allow change to happen as often as it needs to
Lowering risk of change through tools and culture
Dev and Ops
Ops who think like devs
Devs who think like ops
“But that’s me!”
You can always think more like them
Tools
1. Automated infrastructure

If there is only one thing you do…
1. Automated infrastructure

If there is only one thing you do...

System Imager

Chef

Puppet

CFengine

FAI

BCfg2

Cobbler
Role & configuration management

OS imaging
2. Shared version control
Everyone knows where to look

http://www.flickr.com/photos/thunderchild5/1330744559/
3. One step build
Staging

Last SVN-triggered precompile of the templates

Recompile started at 20:43:45 on 2009-06-19
saving prev version..ok (3 ms)
exporting svn........ok (3202 ms)
changing paths.......ok (1 ms)
grabbing old files...ok (6 ms)
creating folders.....ok (317 ms)
compiling config.....
    host: stage-local
    colo: mud
    prod: 1
...............ok (349 ms)
precompiling........ok (29.816 sec)
sync strings........
    added: 14
    deleted: 1
...............ok (16.065 sec)
cleaning up.........ok

Last Stage:

checking precompile..ok
exporting svn........ok (6.034 sec)
fetching versions.....ok (4225 ms)
changing paths.......ok (2 ms)
creating folders.....ok (0 ms)
grabbing snapshot.....ok (16489 ms)
baking help form.....ok (2776 ms)
baking autosuggest...ok (298 ms)
fetching generated...ok (6 ms)
push to staging......ok (8525 ms)
asset foldup.........ok (280 ms)
rewriting CSS paths..ok (2745 ms)
baking output.js......ok (672 ms)
calcing js md5s......ok (210 ms)

Completed in 42.911 sec

Perform Staging

If the staging version looks wonky, you can ask the Magi-cal Pixie to recompile all the templates.
3. One step build and deploy
Deploy Log

Before staging or deploying anything that might block deploy, you must check and update the deploy log.

```
[2009-06-18 13:05:25] [mygrant] NO DEPLOY PLEASE
[2009-06-18 11:29:23] [aaron] site staged
[2009-06-18 11:19:26] [harmes] site deployed (changes...)
[2009-06-18 11:16:44] [harmes] starting deploy...
[2009-06-18 10:20:26] [harmes] site deployed (changes...)
[2009-06-18 10:15:00] [harmes] starting deploy...
[2009-06-18 10:11:16] [harmes] site staged
```

Deployment

When the staging version is ready, click the button below.

WARNING: This sync's the staging version to the live servers. In theory this is what will change, but you might want to test it maybe?

Last deploy:

- pulling site from staging host...ok (10.096 sec)
- syncing to ramdisk in mud........ok (5.992 sec)
- syncing to ramdisk in re2.........ok (7.443 sec)
- stage 1........................ok (25.322 sec)
- stage 2........................ok (42.108 sec)
- stage 3........................ok (45.952 sec)

Completed in 2 min, 16 sec
Deployment

When the staging version is ready, click the button below.

WARNING: This sync's the staging version to the live servers. In theory this is what will change, but you might want to test it maybe?

**Active Deploy:**

- pulling site from staging host...ok (4636 ms)
- syncing to ramdisk in mud.......ok (6.049 sec)
- syncing to ramdisk in re2.......ok (7.349 sec)
- stage 1.........................ok (50.25 sec)
- stage 2.........................ok (1 min, 37 sec)
- stage 3.........................

Phase 1

Phase 2

Phase 3

Waiting for 151 hosts
Running for 2 minutes & 58 seconds
Who? When? What?
Project tools_statsd

Workspace

Last Successful Artifacts

- flickr_statsd-20090603.1244066774.T51204.tgz

Recent Changes

Permalinks

- Last build (#31), 19 days ago
- Last stable build (#31), 19 days ago
- Last successful build (#31), 19 days ago
- Last failed build (#10), 1 mo 5 days ago
Small frequent changes

http://www.flickr.com/photos/mauren/2429240906/
4. Feature flags
(aka branching in code)
Desktop software
Web software
Always ship trunk
Everyone knows exactly where to look
Feature flags

```php
#if ($cfg['enable_feature_video']){
    ...
}#

{ /* smarty */
{if $cfg.enable_feature_beehive}
    ...
{/if}
```
Members Only

Private betas

http://www.flickr.com/photos/healthserviceglasses/3522809727/
Bucket testing

http://www.flickr.com/photos/davidw/2063575447/
Dark launches

http://www.flickr.com/photos/jking89/3031204314/

Dark launches
Free contingency switches

http://www.flickr.com/photos/flattop341/260207875/
5. Shared metrics
Application level metrics
Application level metrics
Adaptive feedback loops

App

RU ok?

System Metrics

maybe?
6. IRC and IM robots
Dev, Ops, and Robots
Having a conversation

IRC

build logs

deploy logs

alerts monitors

search engine
Culture
1. Respect
If there is only one thing you do...
Don’t stereotype
(not all developers are lazy)

http://www.flickr.com/photos/aaronjacobs/64368770/
Respect other people’s expertise, opinions and responsibilities.
Don’t just say “No”
Don’t hide things
Developers: Talk to ops about the impact of your code:

• what metrics will change, and how?
• what are the risks?
• what are the signs that something is going wrong?
• what are the contingencies?

This means you need to **work this out** before talking to ops
2. Trust
Ops needs to trust dev to involve them on feature discussions

Dev needs to trust ops to discuss infrastructure changes

Everyone needs to trust that everyone else is doing their best for the business
Provide knobs and levers
Ops: Be transparent, give devs access to systems

http://www.flickr.com/photos/williamhook/3468484351/
3. Healthy attitude about failure
Failure will happen
If you think you can prevent failure then you aren't developing your ability to respond.
Fire drills

http://www.flickr.com/photos/dnorman/2678090600
4. Avoiding Blame
No fingerpointing

http://www.flickr.com/photos/rocketjim54/2955889085/
Fingerpointyness

problem!!!
argggh!

freaking out,
not talking,
finding fault

blaming,
covering ass

whining,
hiding.
hurt egos

figuring it out

fixed.

fixing things

time
Being productive

problem!!!
argggh!

fixing things
fixed.

figuring it out

feeling guilty
move on with life

move on with life
Developers: Remember that someone else will probably get woken up when your code breaks

http://www.flickr.com/photos/alex-s/353218851/
Ops: provide constructive feedback on current aches and pains

http://www.flickr.com/photos/allspaw/2819774755/
1. Automated infrastructure
2. Shared version control
3. One step build and deploy
4. Feature flags
5. Shared metrics
6. IRC and IM robots

1. Respect
2. Trust
3. Healthy attitude about failure
4. Avoiding Blame
This is not easy
You could just carry on shouting at each other...
(Thank you)