Always ship trunk
Managing change in complex websites
Paul Hammond
ph@pacu:code$ ls -1
_index.html
_index.html.old
_index.html.old.keep
backups
index.html
index.html_old
index.html_older
index.html_paul_DO_NOT_DELETE
index_2009_01_10.html
index_2009_01_11.html
index_2009_01_12.html
index_2009_01_13.html
index_2009_01_13.html_broken
index_2009_01_14.html_broken
index_2009_01_29.html_broken
index_2009_01_30.html_broken
index_2009_02_10.html_broken
index_good.html
index_old.html
ph@pacu:code$
Revision Control
ph@pacu:code$ svn log

r2 | ph | 2010-06-22 20:23:27 -0700 (Tue, 22 Jun 2010) | 1 line

basic user model

r1 | ph | 2010-06-22 20:18:27 -0700 (Tue, 22 Jun 2010) | 2 lines

initial import
If you’re not using any revision control system leave now and start using one
But...
Coordinating many people working on many things at the same time is still hard.
“Just use branching”
...which means merging
“Branching causes problems in Subversion because Subversion doesn’t store enough information to make merging work. In Mercurial, merging is painless and easy, and so branching is commonplace and harmless.”

“Git will allow you to have multiple local branches ... the creation, merging and deletion of those lines of development take seconds ... Git makes this process incredibly easy and it changes the way most developers work when they learn it”

Distributed branching & merging is awesome
But...
“What the !#$* is going on?”
“What the !#$* just went wrong?”
“What the !#$* is running on www34?”
All existing revision control systems were built by people who build installed software
Three kinds of software:

1. Installed software
2. Open source installed software
3. Web applications & Software as a Service
Web apps are not like installed apps
“In the online world, a software product drives a service to which users have access. There is, most often, a single copy of the actual software product in use. There is one consumer of the software: you. The users are consumers of the service built atop the software.”

– Theo Schlossnagel
http://omniti.com/seeds/online-application-deployment-reducing-risk
Does your team admin every computer your software is installed on?
Each new release happens exactly once
Once an upgrade has happened, the old code will never be run again.
Well, kind of...
Upgrades are not deployed to all servers simultaneously.
Staging/QA environments
Beta environments
Upgrades are not deployed to all users simultaneously
A/B testing
Configuration management
Application code
Installed library dependencies
Web service dependencies
Coordinating many people working on many things and running them all at the same time is hard.
But nobody knows you just deployed
(unless you tell them)
Idea one: separate feature launches from infrastructure launches
You can deploy the code for a feature long before you launch it and nobody will know
You can completely rewrite your infrastructure and keep the UI the same and nobody will know
Idea two: run multiple versions of your code at once
You can repeatedly switch between two backend systems and keep the UI the same and nobody will know
You can deploy a non-user facing change to only a small percentage of servers and nobody will know
You can offer different user interfaces to different users and nobody will know
Need a revision control system that allows us to manage multiple parallel versions of the code and switch between them at runtime.
Branches?
Branches don’t help you switch between versions at runtime
Branches don’t help you manage dependency changes that need to happen on all versions at once
Need a revision control system that allows us to manage multiple parallel versions of the code and switch between them at runtime.
Need a revision control system that allows us to manage multiple parallel versions of the code and switch between them at runtime.
Need a revision control system that allows us to manage multiple parallel versions of the code and switch between them at runtime
Manage the different versions within your application
Branch in code
if ($cfg['use_snowflake_ticket_server']) {

    // new hotness
    $ticket = snowflake_ticket();

} else {

    // old and boring
    $ticket = db_generate_ticket();

}
# hardcoded to not run
$cfg["use_snowflake_ticket_server"] = false;
# only in certain environments
if ($cfg['environment'] == 'dev') {
    $cfg['use_snowflake_ticket_server'] = true;
} else {
    $cfg['use_snowflake_ticket_server'] = false;
}
# only for a percentage of users
if (($user->id % 100) < 3) {
    $cfg['use_snowflake_ticket_server'] = true;
} else {
    $cfg['use_snowflake_ticket_server'] = false;
}
# only for a percentage of requests
$rand = rand(0,99);
if ($rand < 3) {
    $cfg['use_snowflake_ticket_server'] = true;
} else {
    $cfg['use_snowflake_ticket_server'] = false;
}
# only for a percentage of requests
if ($cfg['environment'] == 'dev') {
    $cfg['use_snowflake_percentage'] = 100;
} else {
    $cfg['use_snowflake_percentage'] = 2;
}

$rand = rand(0,99);
if ($rand < $cfg['use_snowflake_percentage']) {
    $cfg['use_snowflake_ticket_server'] = true;
} else {
    $cfg['use_snowflake_ticket_server'] = false;
}
# done testing, let's launch
$cfg["use_snowflake_ticket_server"] = true;
# uh oh...
$cfg[\'use_snowflake_ticket_server\'] = false;
# ok, it works again
$cfg['use_snowflake_ticket_server'] = true;
Feature testing on production servers
# team testing
$team = array(41, 287, 5806, 5930);
if (in_array($user->id, $team)) {
  $cfg['use_new_context_widget'] = true;
} else {
  $cfg['use_new_context_widget'] = false;
}
# private staff alpha testing

```php
if ($user->is_admin()) {
    $cfg['use_new_context_widget'] = true;
} else {
    $cfg['use_new_context_widget'] = false;
}
```
# opt-in private staff alpha testing

$has_cookie = isset($_COOKIE['new_context']);
if ($user->is_admin() && $has_cookie) {
    $cfg['use_new_context_widget'] = true;
} else {
    $cfg['use_new_context_widget'] = false;
}
Feature Flip

You can temporarily turn all these features off: **IGNORE ALL**

<table>
<thead>
<tr>
<th>url param</th>
<th>config name</th>
<th>kind</th>
<th>status</th>
<th>change?</th>
</tr>
</thead>
<tbody>
<tr>
<td>explore_search</td>
<td>$cfg[disable_feature_explore_search]</td>
<td>feature_flag</td>
<td>ENABLED FOR THIS HOST</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Turns on the new full width search page.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unicorn Polo</td>
<td>$cfg[enable_unicorn_polo]</td>
<td>feature_flag</td>
<td>ENABLED</td>
<td><strong>turn off</strong></td>
</tr>
<tr>
<td></td>
<td>Very secret; ask Dunstan.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hot Tub Time Machine</td>
<td>$cfg[enable_hot_tub]</td>
<td>feature_flag</td>
<td>DISABLED</td>
<td><strong>turn off</strong></td>
</tr>
<tr>
<td></td>
<td>Not as fun as it sounds.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

# opt-in public betas
if ($user->has_pref('new_context')) {
    $cfg['use_new_context_widget'] = true;
} else {
    $cfg['use_new_context_widget'] = false;
}
# opt-in public betas via groups

```php
if ($user->in_group('new_context')) {
    $cfg['use_new_context_widget'] = true;
} else {
    $cfg['use_new_context_widget'] = false;
}
```
# user tagging (ala doppllr)
if ($user->has_tag('context_widget')) {
    $cfg['use_new_context_widget'] = true;
} else {
    $cfg['use_new_context_widget'] = false;
}
Flexibility
dark launches can be ramped up

```php
if ($cfg['front_page_dark_launch']) {
    # notice we're not keeping the data
    get_some_really_complex_data()
}
```
Three types of feature flags:

1. Development on user facing features
2. Development on infrastructure
3. Kill-switches
# killswitch
if ($cfg['disable_login']) {
    error('Sorry, login is unavailable');
    exit;
}
Usually need multiple flags at once during development and testing
# for development
$cfg['disable_search_tests_all'] = false;
$cfg['disable_search_ui_beta_test'] = false;
$cfg['disable_search_darklaunch'] = false;

# for post launch
$cfg['disable_search'] = false;
Complexity
Separate operational controls from development flags
Be disciplined about removing unused feature flags
Always deploy trunk to every server on every deploy and manage versions through config
# integrate with configuration management

if (in_array('beta', $node['roles'])) {
    # …
} else {
    # …
}

# or generate the application config file
# from configuration management system…
“I can’t tell you how many Subversion users have told me the following story: “We tried to branch our code, and that worked fine. But when it came time to merge back, it was a complete nightmare and we had to practically reapply every change by hand, and we swore never again and we developed a new way of developing software using if statements instead of branches.

“Sometimes they’re even kind of proud of this new, single-trunk invention of theirs. As if it’s a virtue to work around the fact that your version control tool is not doing what it’s meant to do.”

—Joel Spolsky,
http://www.joelonsoftware.com/items/2010/03/17.html
Distributed branching & merging is awesome
Use branches for early development

Branches merged into trunk

Use flags for rollout of almost-finished code
“I can’t tell you how many Subversion users have told me the following story: “We tried to branch our code, and that worked fine. But when it came time to merge back, it was a complete nightmare and we had to practically reapply every change by hand, and we swore never again and we developed a new way of developing software using if statements instead of branches.

“Sometimes they’re even kind of proud of this new, single-trunk invention of theirs. As if it’s a virtue to work around the fact that your version control tool is not doing what it’s meant to do.”

“I can’t tell you how many Subversion users have told me the following story: “We tried to branch our code, and that worked fine. But when it came time to merge back, it was a complete nightmare and we had to practically reapply every change by hand, and we swore never again and we developed a new way of developing software using if statements instead of branches.

“Sometimes they’re even kind of proud of this new, single-trunk invention of theirs. As if it’s a virtue to work around the fact that your version control tool is not doing what it’s meant to do.”

—Joel Spolsky,
http://www.joelonsoftware.com/items/2010/03/17.html
All existing revision control systems were built by people who build installed software
Web apps are not like installed apps
What would a revision control system built for supporting deployed web applications be like?
Thank you!

paul@paulhammond.org
http://www.paulhammond.org/2010/trunk