Continuous Integration using Jenkins

Creating More Robust Code,
Developing More Rapidly,
Improving Testing,
Deploying More Reliably,
Gaining Confidence,
and
Saving Money

2012-05-14

Chris Shenton
@shentonfreude
chris@koansys.com

http://koansys.com/tech/jenkins/

We're a small consulting company with Federal, non-profit and startup clients.
We're UNIX guys who do python, pyramid, mongodb, plone, etc.

What's Continuous Integration?

Continuous Integration is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible. Many teams find that this approach leads to significantly reduced integration problems and allows a team to develop cohesive software more rapidly.

-- Martin Fowler

Nowadays, developers tend to commit code checkins much more frequently, so se can checkout and build many times throughout the day.

Continuous Integration: Some Free Systems

- Cruise Control
- BuildBot
- Hudson/Jenkins

Cruise Control and Jenkins are Java, BuildBot is Python: they all can build projects written in any language. I find Jenkins to be the easiest to setup, and have the most active developer community

Hudson replaced by Jenkins

Dismissed by
Oracle

Forked by
Community

Written by Kosuke Kawaguchi while at Sun
Since the split in November:
* 4x increase in commits and fixed tickets
* Over 13,000 downloads a week
* 1500 users on mailing list, 4000 followers on twitter

How Developers Code

```
while ( not done ) {
    write some code;
    test by poking at it tediously;
    fix the bugs you see;
}
deploy;
cross your fingers nobody finds new bugs;
return;
```

Isn't There Enough World Suffering??

How to Improve It?

- What if we could hire a new developer?
- Make them test all day long
- Every time we make a code change
- Track and report progress
- Give metrics, graphs on improvements
- Report failures quickly

That New Guy is Jenkins

- Doesn't get bored
- Doesn't draw a salary
- Tells you about problems it finds
- Doesn't make fun of you or your bugs
- http://jenkins-ci.org/
- https://github.com/jenkinsci/jenkins

Being non-judgmental is surprisingly important for developers

Basic Continuous Integration

- Check the code repo for changes every few minutes
- Build or compile the code
  - the build must be automate-able
- Run your tests: unit, regression, etc
  - of course you have to have tests to run!
- Alert if problems
- Gather metrics if appropriate

Jenkins Benefits

- Never gets bored doing builds and tests
- Catches problems fast: rapid feedback
- Alerts developers while code is fresh in their minds
- Prevents bugs from propagating downstream
- Cheaper to fix bugs earlier, before QA or Deployment

Importance of Rapid Feedback, as Rebecca Parsons said

Expensive,
Hard to Install

Expensive, Hard to Install

NOT!

Expensive, Hard to Install NOT!

- Free download (9 seconds)
  - $ curl -L -O http://mirrors.jenkins-ci.org/war/latest/jenkins.war
- Run it (18 seconds until ready)
  - $ java -jar jenkins.war [--httpPort=4321]
- Configure though the web
  - http://localhost:4321/
- Profit!

For production use, you'd put this behind Apache or Tomcat or something,
and you'd want to have access controls on it,
or use Jenkins' built-in user manager

Distributed Builds

- Master + slave nodes
- Start Master-only, add slaves later
- Slaves launched by ssh, WMI+DCOM, custom script, Java Web Start (JNLP)
- Slaves can be different platforms
- Allows build/testing on different OSes

Can use JNLP on Slaves inside a firewall to connect to a publicly-visible Master.
The JNLP mechanism can be done via Slave-side web browser or even headless.
We're setting up an OpenStack cloud so Jenkins can spin up different VMs and build/test on them.

Alerts by Mail, RSS
Alerts and Commands by IM

Chris Shenton
!status
Jenkins ContinuousIntegration
status of all projects:
pyrajenkins: last build: 6 (2 days 12 hr ago): SUCCESS: http://localhost:8080/job/pyrajenkins/6/
webcompare: last build: 17 (2 days 20 hr ago): FAILURE: http://localhost:8080/job/webcompare/17/

Chris Shenton
!health
Jenkins ContinuousIntegration
health of all projects:
pyrajenkins: Health [Build stability: 1 out of the last 5 builds failed.(80%), Cobertura Coverage: 72% (78/108) Lines(90%), Test Result: 0 tests failing out of a total of 7 tests.(100%): http://localhost:8080/job/pyrajenkins/6/
webcompare: Health [Build stability: 2 out of the last 5 builds failed.(60%): http://localhost:8080/job/webcompare/17/

Jenkins ContinuousIntegration
Project pyrajenkins build #7: FAILURE in 24 sec: http://localhost:8080/job/pyrajenkins/7/
Chris Shenton: Add support for pylint so Jenkins can yell at me

Jenkins ContinuousIntegration
Yippie, build fixed!
Project pyrajenkins build #9: FIXED in 5.5 sec: http://localhost:8080/job/pyrajenkins/9/

I find the IM integration to be flakey sometimes, the first command works, subsequent ones get no response; might be a bug (see the tracker), might be network/firewall, or could be me.

Helpful Plugins

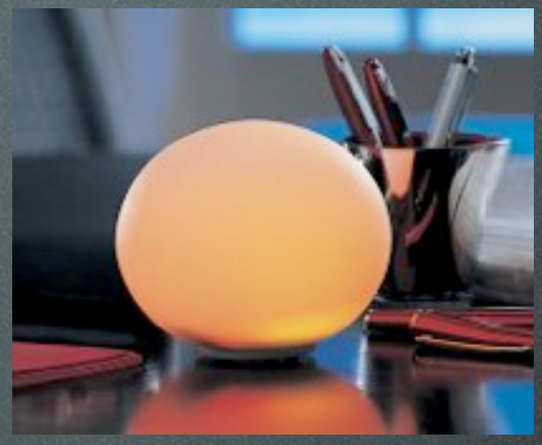
- Subversion: code repo checkout
- Git: code repo checkout, build all branches
- Cobertura: test coverage reporter
- Maven: many plugins available
- Various style violations checkers
- Continuous Integration Game: high scores
- Chuck Norris: motivational help :-)

Over 300 Plugins

- Authentication: LDAP, AD, OpenID, SQL, ...
- Trac, GitHub, Jira, Redmine: trackers
- Selenium, Sauce, Windmill: functional, click testing
- Libvert, VirtualBox, VMware: test in VM
- Xvnc: UI testing
- Android emulation

This large ecosystem is a sign of a healthy developer community

Fun With Extreme Feedback



- Ambient Orbs
- Lava Lamps
- Code Smells

Or Growl or Twitter or Sound files or ...

Immediate Benefits

- Easy to put projects under Jenkins
- Find build problems, quicker to fix
- Find test breakage, quicker to fix
- Reports on increasing test coverage
- Motivation to increase test coverage

At a recent sprint, we had a bunch of coders all working on one project. If any of them committed code that broke the build, Jenkins alerted us immediately, the responsible developer was notified, and they could fix the problem before other developers were affected. This was a big help, and avoided a lot of frustration.

Critical Prerequisites: Automated Build

- No human intervention allowed
  - ant, maven
  - setup.py, pip, buildout
  - (every language has one)

Critical Prerequisites: Code that Tests Your Code

- Faster than clicking around your app
- Can run without firing up the whole stack (web framework, security, ...)
  - unit tests: junit, nose, ...
  - functional tests: selenium, abott, ...
- Run locally before checking in your code: “don’t break the build”

Philip von Weitershausen: Untested code is broken code.
Chris McDonough: 100% test coverage is the LEAST you can do.

Continuous Deployment

- Relies on extensive test suite
- Deploy to production when all pass
- Phased deployment to subset of servers
- Monitor for problems
- Mechanism to roll-back
- You've got to ask yourself one question: Do I feel lucky? Well, do ya, punk?

Rebecca Parsons: "Delivery is where you get the value"

Simple Unit Tests Can Save Hours Debugging

```
class TestUrlManglers(unittest.TestCase):

  def test_normalize_url(self):

    from webcompare import Walker
    w = Walker("fakeurl", "fakeurl", ignoreres=['\\?.*', '#.*', '/RSS.*', '<bound.*'])

    self.assertEquals(w._normalize_url("http://example.com/something"),
                      "http://example.com/something")

    self.assertEquals(w._normalize_url("http://example.com/something/RSS"),
                      "http://example.com/something")

    self.assertEquals(w._normalize_url("http://example.com?querystring"),
                      "http://example.com")

    self.assertEquals(w._normalize_url("http://example.com#fragment"),
                      "http://example.com")
```

Recap

- Jenkins watches the code repo for commits
  - Checks out code
  - Builds it with your build ‘scripts’
  - Runs your tests: unit, performance, UI, ...
  - Reports problems by email, IM, etc
  - Tracks and graphs metrics and trends
  - Can build artifacts, deploy, etc

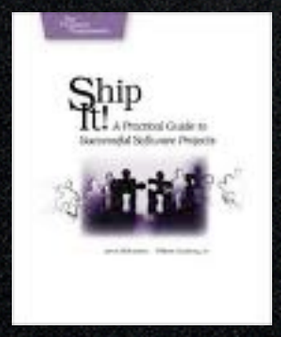
Take Away

- (Visual) Feedback encourages better code
- Non-judgmental, friendly competition
- So easy to setup, no excuse not to! :-)

These Books are Helpful

- They're not specifically about Jenkins, SVN, Java, Python, Ruby, .net, ...
- They cover real programmer hands-on practice with code repos, tests, continuous integration

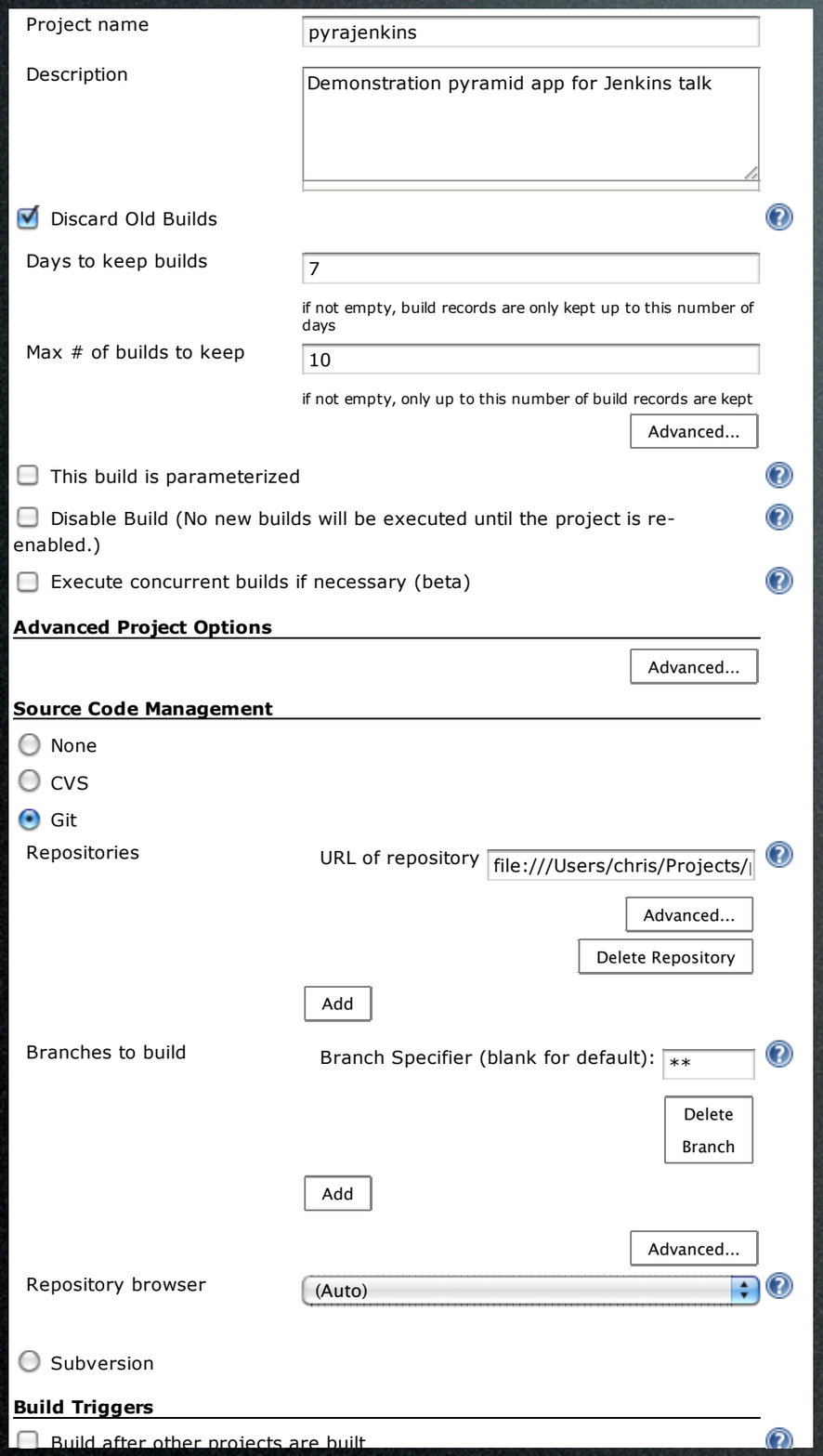
- Pragmatic Project Automation: How to Build, Deploy, and Monitor Java Applications (Mike Clark)

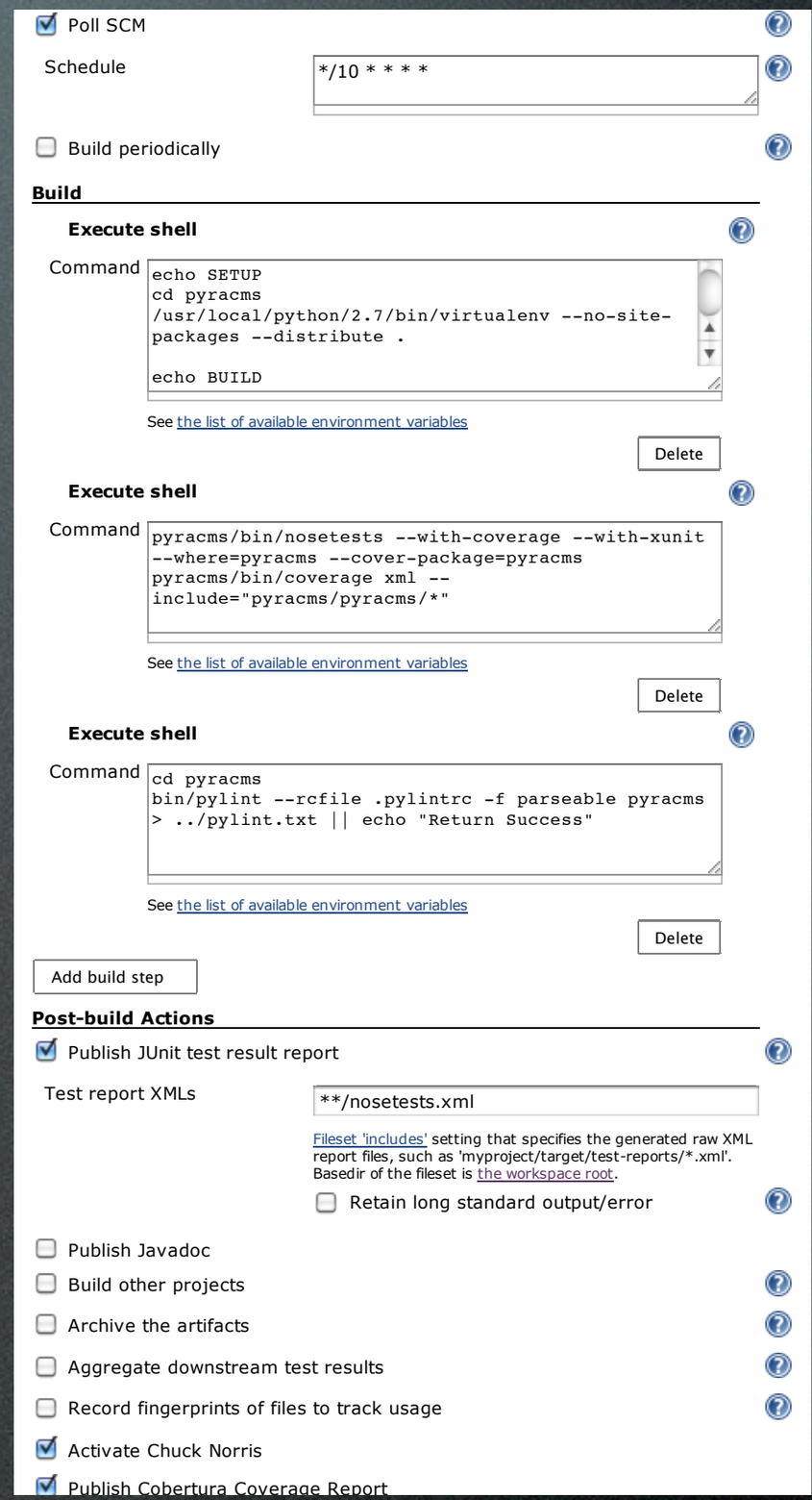


- Ship It! A Practical Guide to Successful Software Projects (Richardson, Gwaltney) [excellent for SW project managers]

Danger!
Live Demo!

Demo: Create Project (1)





1. Give it a name, a code repo to watch; for Git it can build all branches that change.
2. Poll every 10 minutes, then 3 text boxes with (here) Unix commands to run the build and tests.

It can be tricky to get the tests to run where you want them, and output where Jenkins expects it.
You likely will want to turn these commands into scripts once you get them debugged through the web.

Demo: Create Project (2)

Cobertura xml report pattern: **/coverage.xml

This is a file name pattern that can be used to locate the cobertura xml report files (for example with Maven2 use ****/target/site/cobertura/coverage.xml**). The path is relative to the module root unless you have configured your SCM with multiple modules, in which case it is relative to the workspace root. Note that the module root is SCM-specific, and may not be the same as the workspace root.
Cobertura must be configured to generate XML reports for this plugin to function.

Consider only stable builds

Include only stable builds, i.e. exclude unstable and failed ones.

Source Encoding: ASCII

source.encoding.description

Coverage Metric Targets

Condi	70	0	0
Line	80	0	0
Met	80	0	0

Delete Delete Add

Configure health reporting thresholds.
For the row, leave blank to use the default value (i.e. 80).
For the and rows, leave blank to use the default values (i.e. 0).

Report Violations

				XML filename pattern
checkstyle	10	99	99	
codenarc	10	99	99	
cpd	10	99	99	
findbugs	10	99	99	
fxcop	10	99	99	
gendarme	10	99	99	
jcreport	10	99	99	
jslint	10	99	99	
pmd	10	99	99	
pylint	10	99	99	pylint.txt

Source encoding: default

Git Publisher

E-mail Notification

Recipients: chris@koansys.com

Whitespace-separated list of recipient addresses. May reference build parameters like $PARAM. E-mail will be sent when a build fails, becomes unstable or returns to stable.

Send e-mail for every unstable build

Send separate e-mails to individuals who broke the build

Jabber Notification

Targets: chris@koansys.com, chris.shenton@gmail.com

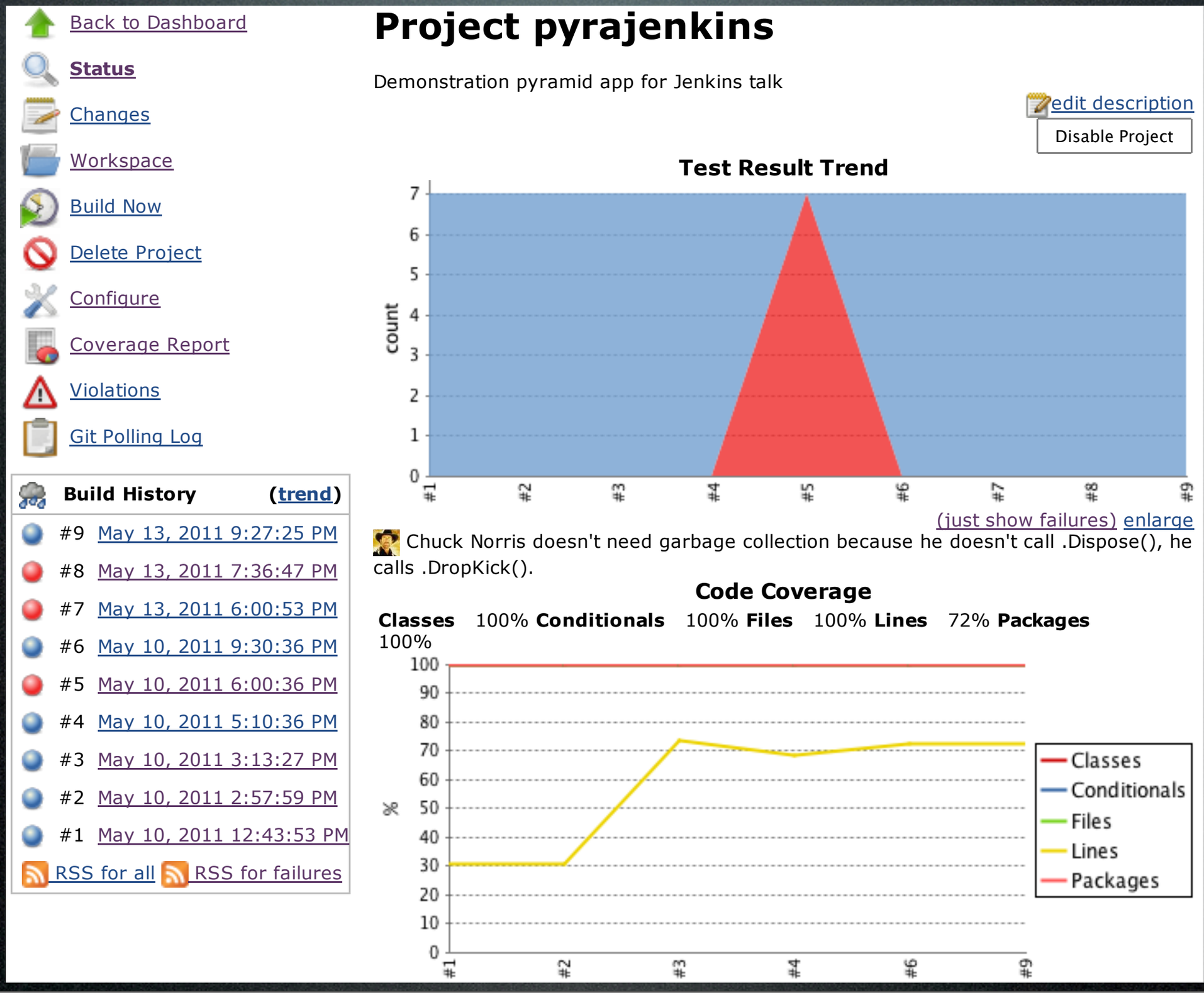
Whitespace separated list of accounts to send notifications to (like 'peter@myjabberserver.org') -- for group chats, prepend a '*' (e.g. '*commitroom@conference.myjabberserver.org') -- note that JIDs that contain '@conference.' are automatically recognized as group chats (no need to prepend '*')

Advanced...

Save

3. The Cobertura code coverage plugin needs to know the data file to display, same with the Violations plugin for Python styles
4. Notify by email and IM

Demo: Status Overview



Demo: Status: Failed Test

Back to Project
Status
Changes
Console Output
Edit Build Information
History
Polling Log
Git Build Data
Test Result
Previous Build
Next Build

Regression

pyracms.tests.PageEditTests.test_save_url (from nosetests)

Failing for the past 1 build (Since #5)
Took 0 ms.
add description

Error Message

```
DummyResource instance has no attribute 'title'
```

Stacktrace

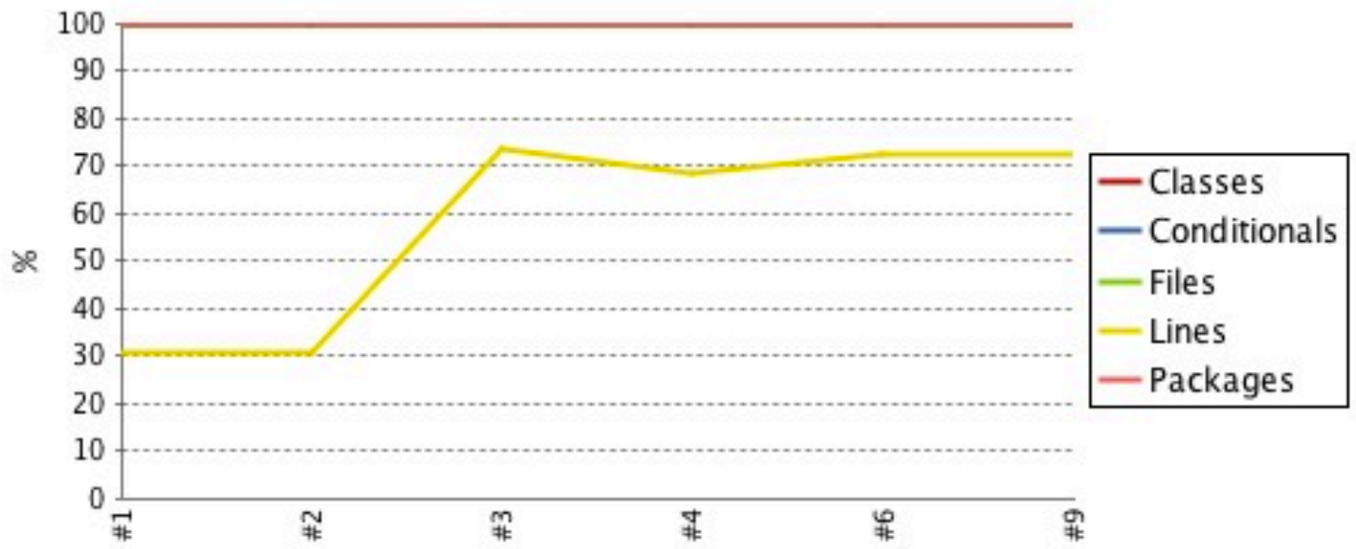
```
Traceback (most recent call last):
  File "/usr/local/python/2.7/lib/python2.7/unittest/case.py", line 318, in
run
    testMethod()
  File
"/Users/chris/.hudson/jobs/pyrajenkins/workspace/pyracms/pyracms/tests.py",
line 48, in test_save_url
    info = self._make_one()
  File
"/Users/chris/.hudson/jobs/pyrajenkins/workspace/pyracms/pyracms/tests.py",
line 41, in _make_one
    return page_edit(context, request)
  File
"/Users/chris/.hudson/jobs/pyrajenkins/workspace/pyracms/pyracms/views.py",
line 46, in page_edit
    'crumbs': _crumbs(context, request),
  File
"/Users/chris/.hudson/jobs/pyrajenkins/workspace/pyracms/pyracms/views.py",
line 20, in _crumbs
    crumbs.append({'title': resource.title, 'url': url})
AttributeError: DummyResource instance has no attribute 'title'
```

Demo: Status: Coverage (1)

Code Coverage

Cobertura Coverage Report

Trend



Project Coverage summary

Name	Classes	Conditionals	Files	Lines	Packages
Cobertura Coverage Report	100% 4/4	100% 0/0	100% 4/4	72% 78/108	100% 1/1

Coverage Breakdown by Package

Name	Classes	Conditionals	Files	Lines
pyracms.pyracms	100% 4/4	N/A	100% 4/4	72% 78/108

Cobertura's designed for Java, so only the Lines applies to my Python code.
The dip at checkin #4 was due to added code without corresponding tests;
I then wrote a test committed in #6 to bring it back up.
Note the Red bars indicating missing coverage.

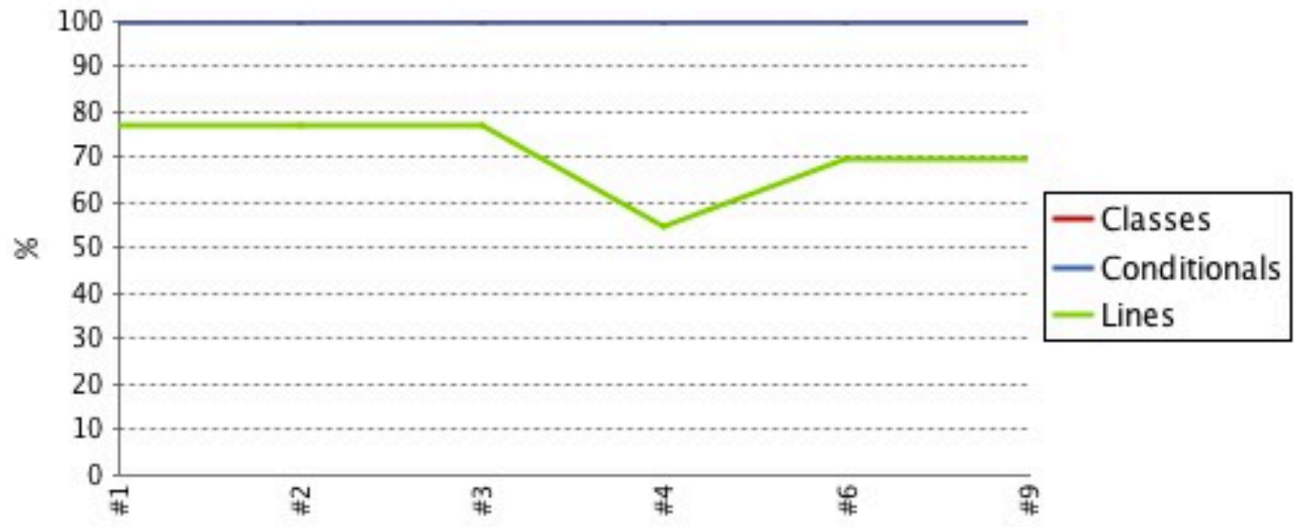
Demo: Status: Coverage (2)

Code Coverage

Cobertura Coverage Report > pyracms.pyracms >

views.py

Trend



File Coverage summary

Name	Classes	Conditionals	Lines
views.py	100% 1/1	100% 0/0	69% 25/36

Coverage Breakdown by Class

Name	Conditionals	Lines
views	N/A	69% 25/36

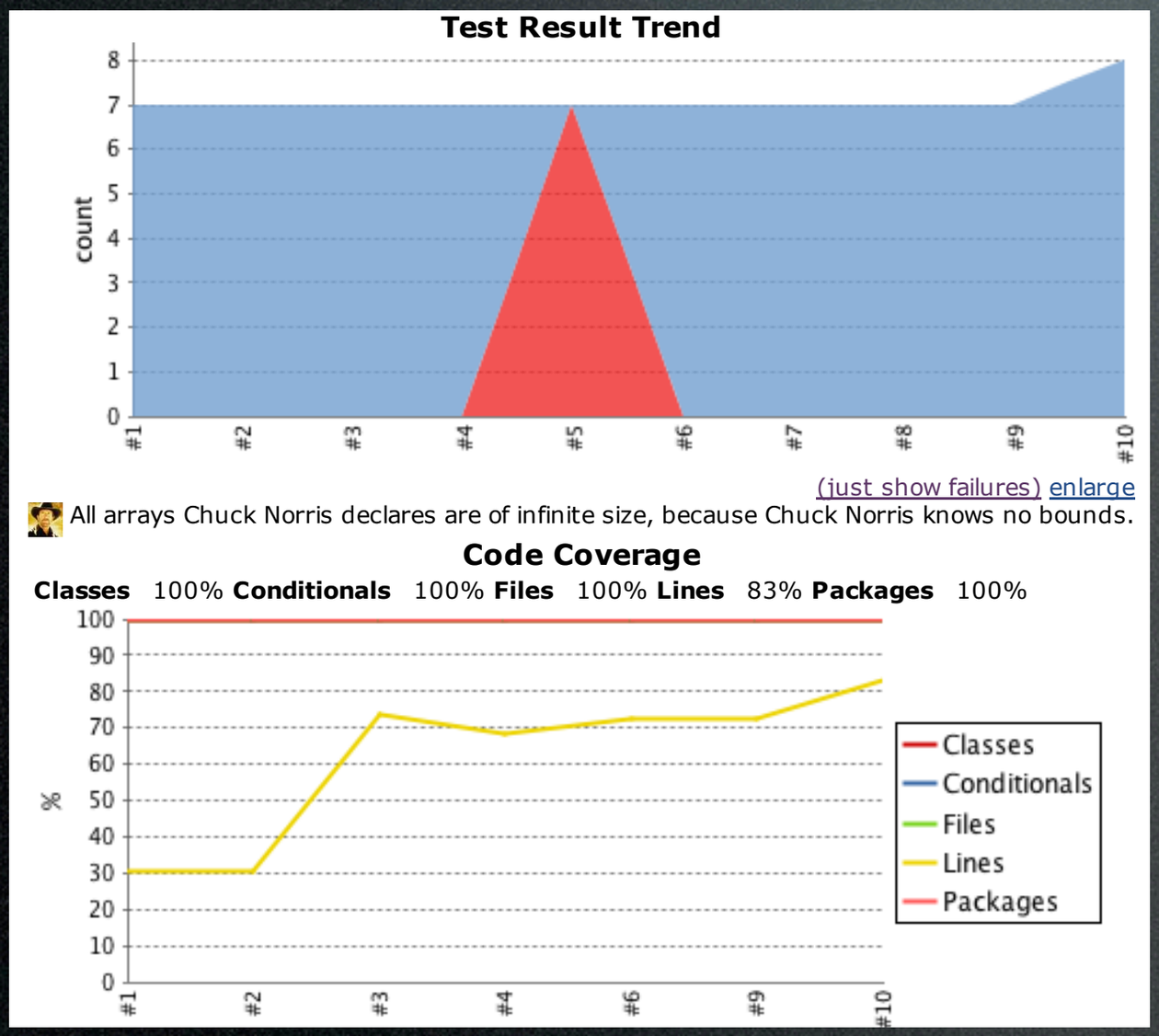
pyracms/pyracms/views.py

```
1 1 from pyramid.view import view_config
2 1 from pyracms.models import Page
3 1 from pyramid.traversal import find_resource
4 1 from pyramid.traversal import find_root
5 1 from pyramid.traversal import resource_path_tuple
6 1 from pyramid.url import resource_url
7 1 from pyramid.httpexceptions import HTTPFound
8
9 1 def _crumbs(context, request):
10       """Return list of dict of title, url for path to context.
11       """
12 1     path_tuple = resource_path_tuple(context)
13 1     root = find_root(context)
14 1     crumbs = []
15 1     pathpartial = []
16 1     for pathseg in path_tuple:
17 1         pathpartial.append(pathseg)
18 1         resource = find_resource(root, pathpartial)
19 1         url = resource_url(resource, request)
20 1         crumbs.append({'title': resource.title, 'url': url})
21 1     return crumbs
22
23 1 @view_config(context=Page, renderer='pyracms:templates/page.pt')
24   def page_view(context, request):
25 1     children = [{'title': page.title,
26                  'url': resource_url(page, request),
27                  }
28                 for (name, page) in context.items()]
29 1     return {'project':'pyracms',
30             'crumbs': _crumbs(context, request),
31             'page': context,
32             'edit_url': resource_url(context, request, 'page_edit'),
33             'add_child_url': resource_url(context, request, 'page_add_child'),
34             'children': children,
35             }
36
37 1 @view_config(name='page_edit',
38               context=Page, renderer='pyracms:templates/page_edit.pt')
39   def page_edit(context, request):
40       # TODO: what if they want to change the 'name' (url)?
41 1     if 'form.submitted' in request.params:
42 0         context.title = request.params['title']
43 0         context.body  = request.params['body']
44 0         return HTTPFound(location=resource_url(context, request))
45 1     return {'project': 'pyracms',
46             'crumbs': _crumbs(context, request),
47             'page': context,
48             'save_url': resource_url(context, request, 'page_edit'),
49             }
50
51   # Reuse the edit page
52 1 @view_config(name='page_add_child',
53               context=Page, renderer='pyracms:templates/page_edit.pt')
54   def page_add_child(context, request):
55 0     if 'form.submitted' in request.params:
56 0         title = request.params['title']
57 0         body  = request.params['body']
58 0         page = Page(title, body)
59 0         context[title] = page
60 0         return HTTPFound(location=resource_url(page, request))
61 0     page = Page('Give Me A Title', 'Add some content.')
62 0     return {'project': 'pyracms',
63             'crumbs': _crumbs(context, request),
64             'page': page,
65             'save_url': resource_url(context, request, 'page_add_child'),
66             }
67
```

The graph shows coverage over time for this file.
The code highlighted in Green has test that exercises it,
but the Red needs new tests to provide coverage.
HAWT! This is a big motivator.

Demo: Adding a Test

Tests and Coverage improved in Build #10



Successful builds make Chuck smile

fin

Questions?

http://www.flickr.com/photos/drewdlecam/3501280430/in/photostream/