Extending Jenkins with Jython

Andrew Neitsch

2015-10-23

github.com/andrewdotn/extending-jenkins-with-jython

Outline

Jenkins

- Industry-standard continuous integration tool
- Regularly and automatically builds and tests your code
- Helps make sure your code works, and stays working

192.168.119.169

Jenkins

search

Jenkins ▸ ENABLE AUTO REFRESH

- New Item
- People
- Build History
- Manage Jenkins
- Credentials

add description

All +

S	W	Name ↓	Last Success	Last Failure	Last Duration	
		test	5 hr 46 min - #4	N/A	1 sec	
		test2	5 hr 45 min - #1	N/A	2 sec	

Icon: S M L

Legend RSS for all RSS for failures RSS for just latest builds

Build Queue

No builds in the queue.

Build Executor Status

1 Idle
2 Idle

Help us localize this page

Page generated: Oct 16, 2015 1:21:05 PM REST API Jenkins ver. 1.633

192.168.119.168

Jenkins

search

Jenkins ▸ ENABLE AUTO REFRESH

- New Item
- People
- Build History
- Manage Jenkins
- Credentials

add description

Welcome to Jenkins!

Please **create new jobs** to get started.

Build Queue

No builds in the queue.

Build Executor Status

1 Idle
2 Idle

Help us localize this page

Page generated: Oct 16, 2015 6:50:18 AM REST API Jenkins ver. 1.633

192.168.119.169

Jenkins ▸ test ▸ configuration

Poll SCM

Configuration Matrix

Add axis

Combination Filter

Run each configuration sequentially

Execute touchstone builds first

Build

Execute shell

Command

```
python /vagrant/foo.py
```

See the list of available environment variables

Delete

Add build step

Post-build Actions

Add post-build action

Save Apply

192.168.119.169

Jenkins

search

Jenkins ▸ test ▸ #3 ▸ python2.7

- Back to Project
- Status
- Changes
- Console Output
  - View as plain text
- Edit Build Information
- Delete Build
- Previous Build

Console Output

```
Started by upstream project "test" build number 3
originally caused by:
 Started by user anonymous
Building in workspace
/var/lib/jenkins/workspace/test/python/python2.7
[python2.7] $ /bin/sh -xe /tmp/hudson8845870300754428442.sh
+ python2.7 /vagrant/foo.py
.
----------------------------------------------------------------------
-------
Ran 1 test in 0.000s

OK
2.7.5 (default, Jun 17 2014, 18:11:42)
[GCC 4.8.2 20140120 (Red Hat 4.8.2-16)]
Finished: SUCCESS
```

If you have tests, you're already running them before you commit, so why bother?

If you have tests, you're already running them before you commit, so why bother?

- Can test regularly forever, not just when there's a change

If you have tests, you're already running them before you commit, so why bother?

- Can test regularly forever, not just when there's a change
- Multi-configuration builds: different Python versions, library versions, platforms, Linux, Windows, and Mac

If you have tests, you're already running them before you commit, so why bother?

- Can test regularly forever, not just when there's a change
- Multi-configuration builds: different Python versions, library versions, platforms, Linux, Windows, and Mac

These are really valuable things that nobody will consistently do by hand!

Code for Python 2 and 3

```
from __future__ import print_function

import sys
import unittest

class TestStringMethods(unittest.TestCase):

  def test_upper(self):
      self.assertEqual('foo'.upper(), 'FOO')

if __name__ == '__main__':
    print(sys.version)
    unittest.main()
```

192.168.119.169

Jenkins

search

Jenkins

- New Item
- People
- Build History
- Manage Jenkins
- Credentials

Build Queue

No builds in the queue.

Build Executor Status

1 Idle
2 Idle

Item name: test

- ○ **Freestyle project**
  This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.
- ○ **Maven project**
  Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.
- ○ **External Job**
  This type of job allows you to record the execution of a process run outside Jenkins, even on a remote machine. This is designed so that you can use Jenkins as a dashboard of your existing automation system. See the documentation for more details.
- ◉ **Multi-configuration project**
  Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

OK

192.168.119.169

Jenkins ▸ test ▸ configuration

User-defined Axis

Name: python

Values: python2.7 python3.4

Delete

Add axis

- [ ] Combination Filter
- [ ] Run each configuration sequentially
- [ ] Execute touchstone builds first

Build

Execute shell

Command

```
${python} /vagrant/foo.py
```

See the list of available environment variables

Delete

Save Apply

192.168.119.169

Jenkins

search

Jenkins ▸ test ▸ #3 ▸ python2.7

- Back to Project
- Status
- Changes
- Console Output
  - View as plain text
- Edit Build Information
- Delete Build
- Previous Build

Console Output

```
Started by upstream project "test" build number 3
originally caused by:
 Started by user anonymous
Building in workspace
/var/lib/jenkins/workspace/test/python/python2.7
[python2.7] $ /bin/sh -xe /tmp/hudson8845870300754428442.sh
+ python2.7 /vagrant/foo.py
.
----------------------------------------------------------------------
-------
Ran 1 test in 0.000s

OK
2.7.5 (default, Jun 17 2014, 18:11:42)
[GCC 4.8.2 20140120 (Red Hat 4.8.2-16)]
Finished: SUCCESS
```

192.168.119.169

Jenkins

search

Jenkins ▸ test ▸ #3 ▸ python3.4

- Back to Project
- Status
- Changes
- Console Output
  - View as plain text
- Edit Build Information
- Delete Build
- Previous Build

Console Output

```
Started by upstream project "test" build number 3
originally caused by:
 Started by user anonymous
Building in workspace
/var/lib/jenkins/workspace/test/python/python3.4
[python3.4] $ /bin/sh -xe /tmp/hudson5943107808054632909.sh
+ python3.4 /vagrant/foo.py
.
----------------------------------------------------------------------
-------
Ran 1 test in 0.000s

OK
3.4.3 (default, Jun 19 2015, 05:46:30)
[GCC 4.8.3 20140911 (Red Hat 4.8.3-9)]
Finished: SUCCESS
```

192.168.119.169

Jenkins

search

Jenkins ▸ test ▸

ENABLE AUTO REFRESH

- Back to Dashboard
- Status
- Changes
- Workspace
- Build Now
- Delete Multi-configuration project
- Configure

Project test

add description

Disable Project

Configuration Matrix	python2.7	python3.4
linux		
windows		
mac		

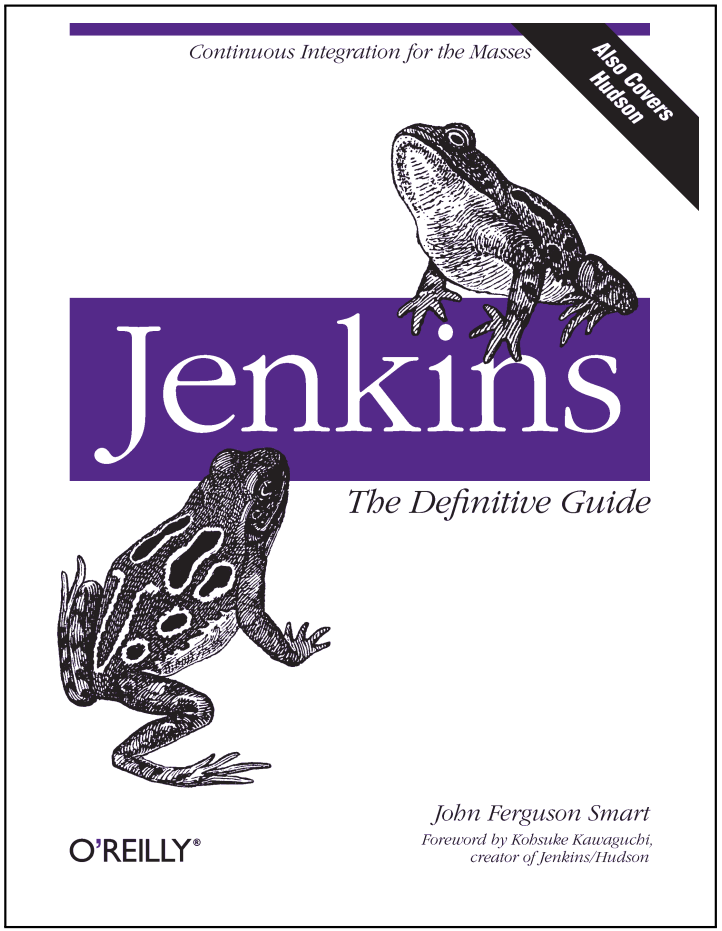
Build History trend

find

#4	Oct 16, 2015 7:34 AM
#3	Oct 16, 2015 7:30 AM
#2	Oct 16, 2015 7:30 AM
#1	Oct 16, 2015 7:28 AM

RSS for all RSS for failures

Advanced uses



- Try servers
- Track betas and release candidates
- Automated deployments

Introduction

Test Isolation

Extending Jenkins

Conclusions


works on
my
machine

works on

jenkins.example.net

and only there

Project goal: “To build the ‘button’ that enables any application to be built and deployed on any server, anywhere.”

Test isolation using docker

- Vagrantfile and setup script in `test-isolation`

Test isolation using docker

- Vagrantfile and setup script in `test-isolation`

```
docker run --rm \
```

Run a container, and clean up afterwards

Test isolation using docker

- Vagrantfile and setup script in `test-isolation`

```
docker run --rm \
  -v /vagrant:/foo:ro \
```

Map a directory into the container, read-only

Test isolation using docker

- Vagrantfile and setup script in `test-isolation`

```
docker run --rm \
  -v /vagrant:/foo:ro \
  centos:centos7 \
```

Use the stock upstream container image

Test isolation using docker

- Vagrantfile and setup script in `test-isolation`

```
docker run --rm \
  -v /vagrant:/foo:ro \
  centos:centos7 \
  python /foo/foo.py
```

And run the tests in the container.

Test isolation using docker

- Vagrantfile and setup script in `test-isolation`

```
docker run --rm \
  -v /vagrant:/foo:ro \
  centos:centos7 \
  python /foo/foo.py
```

Back to Project

Status

Changes

Console Output

View as plain text

Edit Build Information

Delete Build

Console Output

```
Started by upstream project "test-docker" build number 1
originally caused by:
 Started by user anonymous
Building in workspace /var/lib/jenkins/workspace/test-
docker/default
[default] $ /bin/sh -xe /tmp/hudson8437529756374771554.sh
+ docker run --rm -v /vagrant:/foo:ro centos:centos7 python
/foo/foo.py
Usage of loopback devices is strongly discouraged for
production use. Either use `--storage-opt dm.thinpooldev` or
use `--storage-opt dm.no_warn_on_loop_devices=true` to suppress
this warning.
.
----------------------------------------------------------------
-------
Ran 1 test in 0.000s

OK
2.7.5 (default, Jun 24 2015, 00:41:19)
[GCC 4.8.3 20140911 (Red Hat 4.8.3-9)]
Finished: SUCCESS
```

More examples

- Running three hours of Ruby tests in under three minutes

More examples

- Running three hours of Ruby tests in under three minutes

Maybe someone will give a talk on useful things Docker can do for Python developers?

Jenkins is extremely extensible

1,000+ plugins(!)

- Build Failure Analyzer — This plugin analyzes the causes of failed builds and presents the causes on the build page. It does this by using a knowledge base of build failure causes that is built up from scratch.
- Build Monitor Plugin — Build Monitor Plugin provides a highly visible view of the status of selected Jenkins jobs.
- build-metrics-plugin — This plugin uses the results from the Global Build Stats Plugin to generate some basic build metrics. It is really useful in combination with the sidebar links plugin.
- Bumblebee HP ALM Plugin — This plugin integrates Jenkins with HP Quality Center 9, 10, 11, 12, ALM using Agiletestware's Bumblebee product. The plugin updates HP ALM TestPlan, TestLab, TestSet by simple plugin configurations.
- CCCC Plugin — This plugin generates the trend report for CCCC (C and C++ Code Counter).
- CCM Plugin — This plug-in generates reports on cyclomatic complexity for .NET code.
- Checkstyle Plugin — This plugin generates the trend report for Checkstyle, an open source static code analysis program.
- Clang Scan-Build Plugin — This plugin allows you to execute Clang scan-build against Mac or iPhone XCode projects or other scan-build compatible build tools.
- CloudBees Simple Disk Usage Plugin — This plugin offers a simplified disk usage report. It is based on the Unix du command - and as such won't work on Windows masters.
- Clover PHP Plugin — This plugin allows you to capture code coverage reports from PHPUnit. For more information on how to set up PHP projects with Jenkins have a look at the Template for Jenkins Jobs for PHP Projects.
- Clover Plugin — This plugin allows you to capture code coverage reports from Clover. Jenkins will generate and track code coverage across time. This plugin can be used without the need to modify your build.xml.
- Cobertura Plugin — This plugin allows you to capture code coverage report from Cobertura. Jenkins will generate the trend report of coverage.
- Code Dx Plugin — Allows Jenkins to push source and build artifacts to Code Dx and display the aggregated results of its full suite of Static Analysis tools.
- CodeCover Plugin — This plugin allows you to capture code coverage report from CodeCover. Jenkins will generate the trend report of coverage.
- CodeScanner Plugin — This plugin generates the trend report for Codescanner, a tool which uses static analysis to look for bugs, hints and other useful information in Symbian C++ source code.
- Concordion Presenter Plugin — This plugin publishes Concordion test reports for each build
- Coverage Complexity Scatter Plot Plugin — This plugin shows the coverage/complexity scatter plot from Clover or Cobertura plugin results.
- Coverity Plugin — This plugin integrates Jenkins with the Coverity Connect and Coverity Static Analysis tools.
- Cppcheck Plugin — This plugin generates the trend report for CppCheck, a tool for static C/C++ code analysis.
- CPPNCSS Plugin — This plugin allows you to use CppNCSS build reporting tool.
- Cpptest Plugin — This plugin allows you publish Parasoft C++test test results.
- CppUnit Plugin — This plugin enables you to publish CppUnit test results.
- Crap4J Plugin — This plugin reads the "crappy methods" report from Crap4J. Jenkins will generate the trend report of crap percentage and provide detailed information about changes.
- Cucumber Performance Reports Plugin — This plugin reads creates performance reports for jobs running suites of tests using Cucumber-JVM.
- Cucumber Reports Plugin
- Custom History — Show custom text in scm log manner
- Datadog Plugin — This plugin is used to forward metrics, events, and service checks to your account at Datadog, automatically.
- dbCharts Plugin — Add charts based on JDBC database data series.
- Dependency Analyzer Plugin — This plugin parses dependency:analyze goal from maven build logs and generates a dependency report
- Dependency Graph View Plugin — Shows a dependency graph of the projects using graphviz. Requires a graphviz installation on the server.
- Disk Usage Plugin — This plugin records disk usage.
- Downstream buildview plugin — This plugin allows you to view the full status all the downstream builds so that we can graphically see that everything for this build has been completed successfully.
- Doxygen Plugin — This plugin generates Doxygen documentation and publishes HTML reports generated by the Doxygen tool.
- dtmemory plugin
- DRY Plugin — This plugin generates the trend report for duplicate code checkers like CPD or Simian.
- Dynatrace Plugin — This plugin integrates Jenkins with Dynatrace to get performance metrics for tests.
- Emma Plugin — This plugin allows you to capture code coverage report from Emma. Jenkins will generate the trend report of coverage.
- eXtreme Feedback Panel Plugin — This plugin provides an eXtreme Feedback Panel that can be used to expose the status of a selected number of Jobs.
- EzWall Plugin — Add a button to display a fullscreen buildwall on each view.
- FindBugs Plugin — This plugin generates the trend report for FindBugs, an open source program which uses static analysis to look for bugs in Java code.
- FitNesse Plugin — This plugin can be used to both execute and report on FitNesse tests so that they can be integrated into a Jenkins build.
- FlashLog Plugin — Captures Adobe Flash Player logs during a Hudson build in order to save them as build artifacts. Useful to debug unit tests or integration tests.
- Fortify 360 Plugin — Fortify 360 FPR post-processing and uploading to Fortify 360 Server
- Gallio Plugin — This plugin makes it possible to publish Gallio/MbUnit test results
- Git Changelog Plugin — Extracts a changelog out of commit messages between two GIT revisions, the latest release tags are searched automatically. This changelog can be postprocessed and converted to either an human readable git changelog listing all commits, c
- Global Build Stats Plugin
- Gnat Plugin — This plugin makes it possible to integrate Gnat features for Ada languages in Jenkins.
- Grinder Plugin — This plugin reads output result files from Grinder performance tests, and will generate reports showing test results for every build and trend reports showing performance results across builds.
- iceScrum Plugin — This plugin integrates iceScrum Pro to Jenkins.
- Image Gallery Plugin — This plug-in reads a job workspace and collects images to produce an image gallery using colorbox lightbox Javascript library.
- JaCoCo Plugin — This plugin allows you to capture code coverage report from JaCoCo. Jenkins will generate the trend report of coverage.
- Japex Plugin — This plugin adds Japex support so that Jenkins can display trend reports and other useful metrics.
- JavaNCSS Plugin — This plugin allows you to use JavaNCSS build reporting tool.
- JavaTest Report Plugin — This plugin enables Jenkins to load test output from JavaTest test harness, which is commonly used by TCK tests for various JSRs.
- JDepend Plugin — The JDepend Plugin is a plugin to generate JDepend reports for builds.
- JGiven Plugin — This plugin lets you generate JGiven reports as a post-build action.
- JSUnit plugin — This plugin allows you publish JSUnit test results
- JUnit Attachments Plugin — This plugin can archive certain files (attachments) together with your JUnit results.
- JUnit Plugin — Allows JUnit-format test results to be published.
- Klocwork Plugin — This plugin makes it possible to publish Klocwork reports in Jenkins.
- Koji Plugin — This plugin enables integration with Koji build system providing clean-room environment for production builds. Main focus of Koji is on reproducibility, auditability and isolation of build executors which are freshly provisioned each time.
- Label Linked Jobs Plugin — A plugin to facilitate maintenance when using numerous/complex labels.
- LabeledTestGroupePublisher Plugin — This plugin provides the functionality to group tests by types, such as 'unit test', 'smoke test', 'regression test', etc.
- Log Parser Plugin — Parse the console output and highlight error/warning/info lines.
- Logstash Plugin — This plugin pushes logs and build data to a Logstash indexer such as Redis or RabbitMQ.
- Measurement Plots Plugin — Plot measurements embedded into the standard output and error streams of tests.
- Memory Map Plugin
- Mozmill Plugin — This plugin allows you to use Mozmill for automated tests in Gecko based applications such as Firefox and Thunderbird.
- MSTest Plugin — This plugin converts MSTest TRX test reports into JUnit XML reports so it can be integrated with Jenkin's JUnit features.
- MTTR Plugin — This plugin statistics job builds mean time to repair.
- NCover Plugin — Archive and publish .NET code coverage HTML reports from NCover.
- NSIQ Collector Plugin — This plugin shows the lines of code and cyclomatic complexity from from external metric tools named N'SIQ Collector.
- NUnit Plugin — This plugin allows you to publish NUnit test results.
- Performance Plugin — This plugin allows you to capture reports from JMeter and JUnit . Jenkins will generate graphic charts with the trend report of performance and robustness.
- PerfPublisher Plugin — This plugin generates global and trend reports for tests results analysis. Based on an open XML tests results format, the plugin parses the generated files and publish statistics, reports and analysis on the current health of the project.
- Pitmutation — This plugin displays pitmutation report statistics.
- play-plugin — This plugin integrates Play! framework to Jenkins.
- Plot Plugin — This plugin provides generic plotting (or graphing) capabilities in Jenkins.
- Plugin State Plugin — This plugin allows you to keep track of which plugins have been installed and how many projects are using them.
- PMD Plugin — This plugin generates the trend report for PMD, an open source static code analysis program.
- Programming Research Plugin — A plugin for doing static code analysis using PRQA.
- PureCoverage plugin — PureCoverage plugin reports coverage results from Rational PureCoverage tool (c++ coverage tool).
- QF-Test Plugin — Allows users to run QF-Test suites as part of the build step
- Quality Center Plugin — This plugin allows Hudson to trigger HP Quality Center test sets.
- Radiator View Plugin — Provides a job view displaying project status in a highly visible manner. This is ideal for displaying on a screen on the office wall as a form of Extreme Feedback Device.
- reFit Plugin — A plugin for publishing Fit test reports created by reFit, an implementation of the Framework for Integrated Test (Fit).
- Regression Report Plugin — This plugin sends a mail if your test cases find regression (new failure).
- Robot Framework Plugin — This plugin collects and publishes Robot Framework test results
- RubyMetrics plugin — Ruby metric reports for Jenkins. Roov, Rails stats, Reile notes and Flog.
- Russian Salad Test Report Plugin — This plugin enables Jenkins to load test output from Russian Salad test suite.
- Sahagin Plugin — This plugin generates Sahagin HTML report from Sahagin test output data and attaches it to build.
- Scoverage Plugin — This plugin allows you to publish scoverage results in Jenkins as a trend graph.
- Selenium AES Plugin — This plugin is for continuous regression test by Selenium Auto Exec Server (AES).
- Seleniumhtg Plugin — This plugin allows you to run and load HTML Selenese suite result generate by Selenium Server from Selenium-hq.
- seleniumhtmlreport Plugin — This plugin visualizes the results of selenium tests.
- Serenity Plugin — Serenity is a Java code coverage, complexity and dependency library that uses dynamic instrumentation, with a friendly Hudson ui.
- Simple Parameterized Builds Report plugin — This Jenkins plugin makes it easy to visualize the various builds for a parameterized project that were run using the same set of parameters.

Use case: Code metrics

Would be nice to see things like

- Code coverage rates
- Function documentation rates
- Style compliance rates
- Lines of code
- Number of slides in presentation

All right on the dashboard

Jenkins is written in Java

Jenkins is written in Java

What about Jython?

Issues

- Compile-time annotations and extension discovery

Issues

- Compile-time annotations and extension discovery
- Jython scripts can call Java much more easily than Java can call Jython

```
PythonInterpreter interpreter = new PythonInterpreter();
interpreter.exec("from Building import Building");
buildingClass = interpreter.get("Building");
PyObject buildingObject = buildingClass.__call__(
    new PyString(name), PyString(location),
    new PyString(id));
return (BuildingType)buildingObject
    .__tojava__(BuildingType.class);
```

Issues

- Compile-time annotations and extension discovery
- Jython scripts can call Java much more easily than Java can call Jython
- Newer tool clamp, can only invoke interface methods

Success

PythonWrapper and ppsm

Success

PythonWrapper and ppsm

Really well done!

Success

PythonWrapper and ppsm

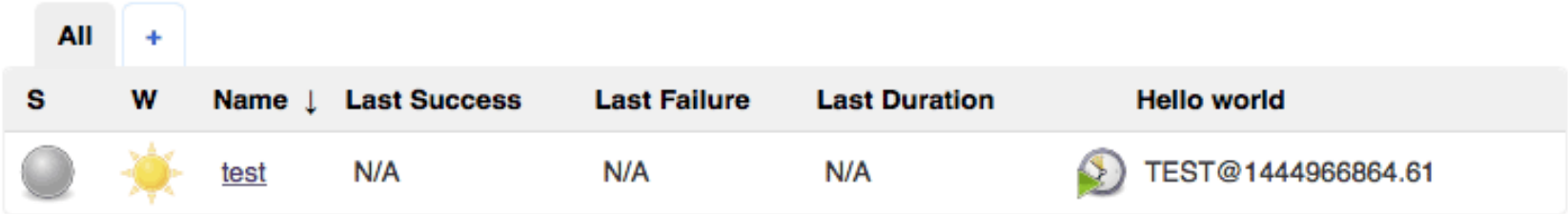
Really well done!

Main development technique:

- View source for existing plugins
- And Jenkins source code

Working example

See the `jython-hello-world` directory.




```
def hello(job):
    return job.getName().upper() + "@" + str(time.time())
...
<j:jelly xmlns:j="jelly:core">
  <td>${it.hello(job)}</td>
</j:jelly>
...
@Extension
public class HelloWorldColumn extends ListViewColumnPW {
    public String hello(Object o) {
        return execPython("hello", o).toString();
    }
}
```

Still need to learn guts

- Jelly XML-based template/data-binding language
- Java data model

Still need to learn guts

- Jelly XML-based template/data-binding language
- Java data model

Writing a whole plugin in Python
doesn't feel like a net win

Still need to learn guts

- Jelly XML-based template/data-binding language
- Java data model

Writing a whole plugin in Python
doesn't feel like a net win
but calling a Python library
from Java definitely could be

Introduction

Test Isolation

Extending Jenkins

Conclusions

Conclusions

- Jenkins runs tests that humans don't have the patience to run by hand: multiple Python versions, multiple platforms, every commit, every day

Conclusions

- Jenkins runs tests that humans don't have the patience to run by hand: multiple Python versions, multiple platforms, every commit, every day
- You can help your fellow developers by making tests that will run anywhere

Conclusions

- Jenkins runs tests that humans don't have the patience to run by hand: multiple Python versions, multiple platforms, every commit, every day
- You can help your fellow developers by making tests that will run anywhere
- You can extend Jenkins with Jython ...

Conclusions

- Jenkins runs tests that humans don't have the patience to run by hand: multiple Python versions, multiple platforms, every commit, every day
- You can help your fellow developers by making tests that will run anywhere
- You can extend Jenkins with Jython ... but there will still be a lot of Java.

Questions?

Call for talks

"w/o interesting talks, there's not a ton of point in 'meeting up'"

Call for talks

"w/o interesting talks, there's not a ton of point in 'meeting up'"

1. Pick something you find interesting
2. Talk about it
3. Include exercises for people to hack on

Suggested exercises

- Install Jenkins, and get it testing some code you care about
- Isolate your tests
- Try extending Jenkins