
django-jenkins Documentation

Release 0.19.0

Mikhail Podgurskiy, Gabriel Le Breton

Jun 04, 2017

Contents

1	Indices and tables	3
2	Getting started	5
2.1	Configuring django project	5
2.2	Configuring jenkins	6
2.3	Results	9

Plug and play continuous integration with django and jenkins

There are a lot of ways to enable continuous integration with your django project. But most of them require a lot of time to configure and setup ci server and make changes in your django project.

This tutorial introduces a new way to enable continuous integration for django project, with minimal projects modifications by means of with django-jenkins.

Previously, ad-hoc integration django with jenkins required using nose testing frameworks instead of django native unittests. Since nose project uses custom test loader it leads to a lot of problems with django code - duplicate signals subscriptions, module loading and other test-only realted errors. Django-jenkins uses a standard unittest runner, so the code under test works like in production. No difference between django-jenkins run and manage.py test so keeps your hands free for real work.

CHAPTER 1

Indices and tables

- genindex
- search

Configuring django project

To enable `django_jenkins` you need only to add `django_jenkins` to `INSTALLED_APPS` in `settings.py`.

Running

```
$ python manage.py jenkins
```

will do the same job as

```
$ python manage.py tests
```

but also will create reports folder in the root of your django project with jenkins parsable pylint, test coverage and tests reports.

Django-jenkins have support for several other testing tools/ To enable it, you should include tools task to `JENKINS_TASKS` settings variable.

```
JENKINS_TASKS = (  
    'django_jenkins.tasks.run_pep8',  
    'django_jenkins.tasks.run_pyflakes',  
    'django_jenkins.tasks.run_jslint',  
    'django_jenkins.tasks.run_csslint',  
    'django_jenkins.tasks.run_sloccount'  
)
```

Please, note that corresponding task tool should be installed on jenkins server manually. Please refer to [django-jenkins README](#) for specific task dependencies list.

In order to get the right coverage, `'django_jenkins'` app should be included as early as possible in `INSTALLED_APPS`

This tutorial doesn't cover the library dependency management and deploying your django project on external server. Basically you could setup the CI server as you did in your local environment.

But if you prefer automatically installation and configuration dependencies on CI server, you could easily add [virtualenv](#) support for your project.

Add requirements to your `requirements.txt` file:

```
Django
django-jenkins
# any other libraries for your project
```

Running

```
$ virtualenv --python=python2.6 env
$ env/bin/pip install -r requirements.txt
```

will create local folder `env` with the required libraries for your project. Running those commands on other servers will help to sync the external dependencies.

Configuring jenkins

After a fresh [Jenkins](#) installation, you'll need to install two required plugins:

- [Violations](#) plugin for parsing `pylint` reports and
- [Cobertura](#) Plugin for showing the code coverage.

Install these plugins via `Manage Jenkins -> Manage Plugins -> Available`.

Start new job with creating free-style project:

The screenshot shows the Jenkins 'New Job' configuration page. The job name is 'django-jenkins-tutorial'. The 'Build a free-style software project' option is selected. The interface includes a sidebar with navigation links, a build queue, and a footer with page generation information.

Build Queue
No builds in the queue.

Build Executor Status	
#	Master
1	Idle

7BitsInternalRadioExchange
1 Idle

MedexportXP (offline)

Page generated: Jan 24, 2012 7:12:43 AM [Jenkins ver. 1.442](#)

<https://sites.google.com/site/kmmbvnr/home/django-jenkins-tutorial>

After configuring the repository access, set up the build triggers. Poll SCM will run the build if any changes in repository are found. The line `* / 5 * * * *` means checking repository every 5 minutes.

Webpage Screenshot

The screenshot shows the Jenkins configuration interface. At the top, there is an 'Add' button and an 'Advanced...' button. Below these is the 'Repository browser' dropdown menu set to '(Auto)'. The 'Build Triggers' section includes checkboxes for 'Build after other projects are built', 'Build periodically', and 'Poll SCM' (which is checked). A 'Schedule' text area contains the cron expression '# MINUTE HOUR DOM MONTH DOW' and '*/*5 * * * *'. The 'Build' section has an 'Add build step' button. The 'Post-build Actions' section includes checkboxes for 'Publish Javadoc', 'Archive the artifacts', 'Aggregate downstream test results', and 'Publish JUnit test result report'. A URL bar at the bottom shows 'http://127.0.0.1:8080/job/hudson_tutorial/configure'.

Select Add build step -> Execute shell. Add commands to setup environment and the run tests command

```
$ python manage.py jenkins --enable-coverage
.. image:: _static/jenkins-2.png
```

Specify the location of test reports - reports/TEST-*.xml and reports/lettuce.xml (in case you are using lettuce tests) files.

CHANGED in 0.13.0: test reports from TEST-*.xml now stored in one file: junit.xml.

Webpage Screenshot

Post-build Actions

- Scan workspace for open tasks
- Aggregate downstream test results
- Console output (build log) parsing
- Post build task
- Publish Cobertura Coverage Report
- Publish JUnit test result report
 - Test report XMLs

Fileset 'includes' setting that specifies the generated raw XML report files, such as 'myproject/target/test-reports/*.xml'. Basedir of the fileset is [the workspace root](#).
 - Retain long standard output/error
- Publish SLOCCount analysis results
- Publish Selenium Report
- Report Violations
- Use publishers from another project

<http://build.fairdevteam.com/job/django-jenkins-tutorial/configure>

Configure locations of violations reports:

Webpage Screenshot

Report Violations

☀️ ☁️ 🟡 XML filename pattern

checkstyle	<input type="text" value="10"/>	<input type="text" value="999"/>	<input type="text" value="999"/>	<input type="text"/>
codenarc	<input type="text" value="10"/>	<input type="text" value="999"/>	<input type="text" value="999"/>	<input type="text"/>
cpd	<input type="text" value="10"/>	<input type="text" value="999"/>	<input type="text" value="999"/>	<input type="text"/>
cpplint	<input type="text" value="10"/>	<input type="text" value="999"/>	<input type="text" value="999"/>	<input type="text"/>
csslint	<input type="text" value="10"/>	<input type="text" value="999"/>	<input type="text" value="999"/>	<input type="text" value="reports/csslint.report"/>
findbugs	<input type="text" value="10"/>	<input type="text" value="999"/>	<input type="text" value="999"/>	<input type="text"/>
fxcop	<input type="text" value="10"/>	<input type="text" value="999"/>	<input type="text" value="999"/>	<input type="text"/>
gendarme	<input type="text" value="10"/>	<input type="text" value="999"/>	<input type="text" value="999"/>	<input type="text"/>
jcreport	<input type="text" value="10"/>	<input type="text" value="999"/>	<input type="text" value="999"/>	<input type="text"/>
jslint	<input type="text" value="10"/>	<input type="text" value="999"/>	<input type="text" value="999"/>	<input type="text" value="reports/jslint.xml"/>
pep8	<input type="text" value="10"/>	<input type="text" value="999"/>	<input type="text" value="999"/>	<input type="text" value="reports/pep8.report"/>
pmd	<input type="text" value="10"/>	<input type="text" value="999"/>	<input type="text" value="999"/>	<input type="text"/>
pylint	<input type="text" value="10"/>	<input type="text" value="999"/>	<input type="text" value="999"/>	<input type="text" value="reports/pyflakes.report, reports/pylint.report"/>
simian	<input type="text" value="10"/>	<input type="text" value="999"/>	<input type="text" value="999"/>	<input type="text"/>
stylecop	<input type="text" value="10"/>	<input type="text" value="999"/>	<input type="text" value="999"/>	<input type="text"/>
Per file limit	<input type="text" value="100"/>			<input type="text"/>
Source Path Pattern	<input type="text"/>			<input type="text"/>
Faux Project Path	<input type="text"/>			<input type="text"/>
Source encoding	<input type="text" value="default"/>			<input type="text"/>

<http://build.fairdevteam.com/job/django-jenkins-tutorial/configure>

Test coverage reports are in `reports/coverage.xml`

Webpage Screenshot

Git Publisher

Publish Cobertura Coverage Report

Cobertura xml report pattern

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.

Coverage Metric Targets

Lines	80	0	0
Condition	70	0	0

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).

E-mail Notification

Save

http://127.0.0.1:8080/job/hudson_tutorial/configure

That's all!

Results

Press `Build Now` and see what you've got:

`Pylint` and other lint tools reports for each builds, shows what warning/errors are new for each build.

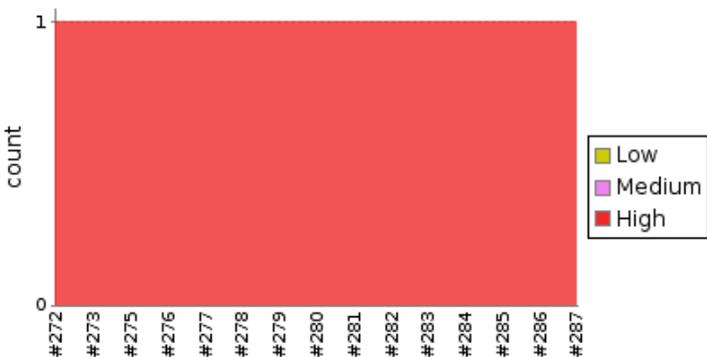
Webpage Screenshot



Violations Report for build 287

Type	Violations	Files in violation
csslint	1	1
jslint	4	2
pep8	52 (+2)	21 (+2)
pylint	80	21

csslint



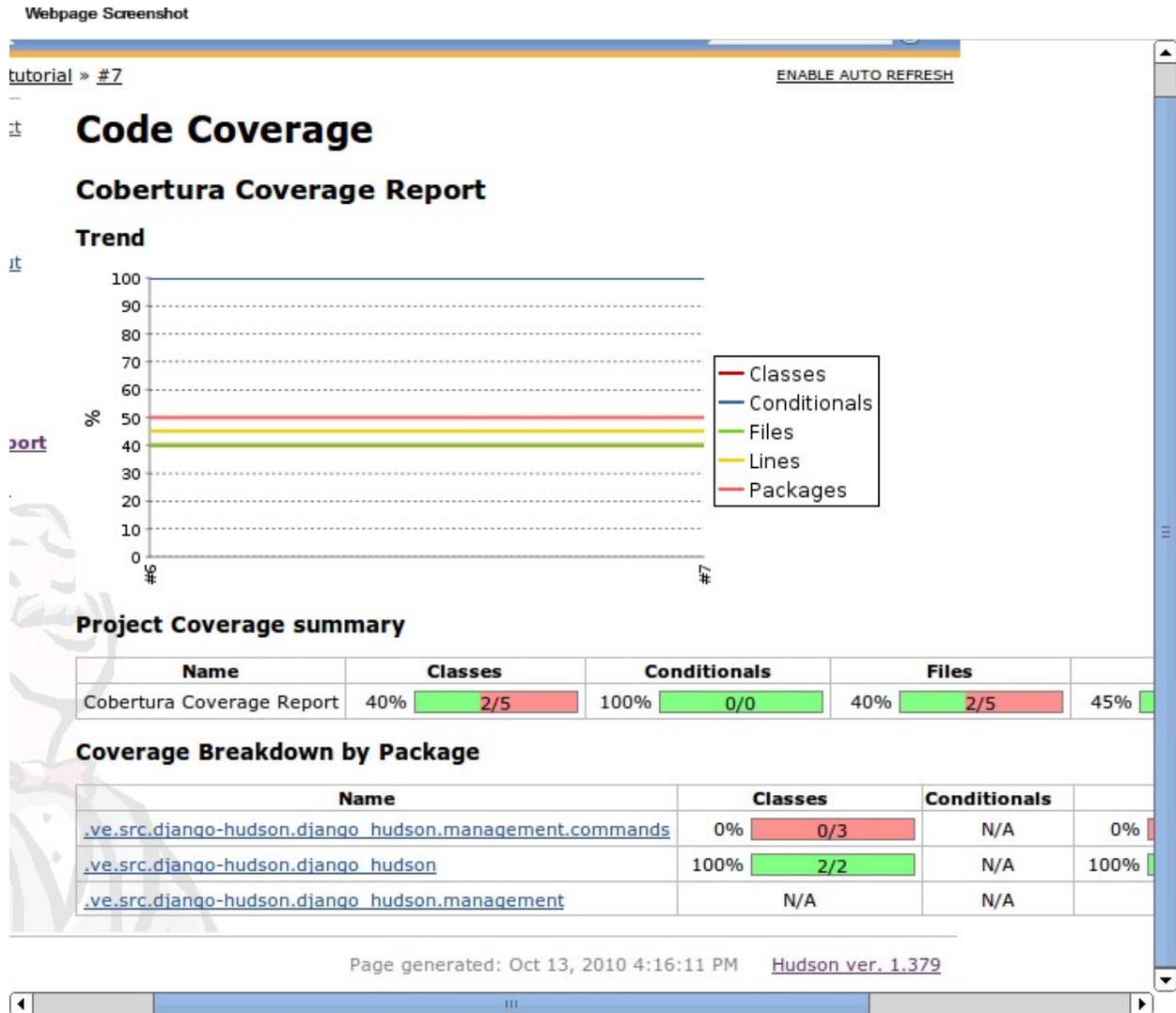
filename	l	m	h	number
tests/static/css/test.css	0	0	1	1

jslint



http://build.fairdevteam.com/job/django_jenkins_python26/violations/#pylint

Nice code coverage reports, showing the total coverage, and colored file listing.



And of course the test results and output.

Webpage Screenshot

Jenkins

mpodgurskiy | [log out](#)

[Jenkins](#) » [django_jenkins_python26](#) » [#169](#) » [Test Results](#)
[ENABLE AUTO REFRESH](#)

- [Back to Project](#)
- [Status](#)
- [Changes](#)
- [Console Output](#)
- [Edit Build Information](#)
- [History](#)
- [Polling Log](#)
- [Git Build Data](#)
- [Test Result](#)
- [Coverage Report](#)
- [Violations](#)
- [SLOCCount](#)
- [Score card](#)
- [Previous Build](#)
- [Next Build](#)

Test Result

0 failures (±0)

115 tests (±0)
 Took 5 min 52 sec.
[add description](#)

All Tests

Package	Duration	Fail	(diff)	Skip	(diff)	Total	(diff)
django.contrib.sessions.tests	5 min 40 sec	0		0		111	
django_jenkins.tests	3 sec	0		0		1	
test_app.tests	3 sec	0		0		1	
test_app.wmtests	6.1 sec	0		0		2	

[Help us localize this page](#)
Page generated: Jan 24, 2012 7:40:19 AM
[Jenkins ver. 1.442](#)

http://build.fairdevteam.com/job/django_jenkins_python26/169/testReport/