Buildbot Tutorial Documentation

Release latest

Thomas Vander Stichele, Lukas Blakk

April 02, 2014

Contents

Contents:

CHAPTER 1

First Run

1.1 Goal

This tutorial will take you from zero to running your first buildbot master and slave as quickly as possible, without changing the default configuration.

This tutorial is all about instant gratification and the five minute experience: in five minutes we want to convince you that this project Works, and that you should seriously consider spending some more time learning the system. In this tutorial no configuration or code changes are done.

This tutorial assumes that you are running on Unix, but might be adaptable easily to Windows.

For the quickest way through, you should be able to cut and paste each shell block from this tutorial directly into a terminal.

1.2 Getting the code

There are many ways to get the code on your machine. For this tutorial, we will use easy_install to install and run buildbot. While this isn't the preferred method to install buildbot, it is the simplest one to use for the purposes of this tutorial because it should work on all systems. (The preferred method would be to install buildbot from packages of your distribution.)

To make this work, you will need the following installed:

- python and the development packages for it
- virtualenv
- git

Preferably, use your package installer to install these.

You will also need a working Internet connection, as virtualenv and easy_install will need to download other projects from the Internet.

Let's dive in by typing at the terminal:

```
cd
mkdir -p tmp/buildbot
cd tmp/buildbot
virtualenv --no-site-packages sandbox
source sandbox/bin/activate
easy_install buildbot
```

1.3 Creating a master

At the terminal, type:

```
cd sandbox
buildbot create-master master
mv master/master.cfg.sample master/master.cfg
```

Now start it:

```
buildbot start $VIRTUAL_ENV/master
tail -f $VIRTUAL_ENV/master/twistd.log
```

You will now see all of the log information from the master in this terminal. You should see lines like this:

```
2009-07-29 21:01:46+0200 [-] twisted.spread.pb.PBServerFactory starting on 9989
2009-07-29 21:01:46+0200 [-] Starting factory <twisted.spread.pb.PBServerFactory instance at 0x1fc8ab
2009-07-29 21:01:46+0200 [-] BuildMaster listening on port tcp:9989
2009-07-29 21:01:46+0200 [-] configuration update started
2009-07-29 21:01:46+0200 [-] configuration update complete
```

1.4 Creating a slave

Open a new terminal, and first enter the same sandbox you created before:

```
cd
cd tmp/buildbot
source sandbox/bin/activate
```

Install buildslave command:

```
easy_install buildbot-slave
```

Now, create the slave:

```
cd sandbox
buildslave create-slave slave localhost:9989 example-slave pass
```

The user:host pair, username, and password should be the same as the ones in master.cfg; please verify this is the case byt looking at the section for c['slaves']:

```
cat $VIRTUAL_ENV/master/master.cfg
```

Now, start the slave:

```
buildslave start $VIRTUAL_ENV/slave
```

Check the slave's log:

```
tail -f $VIRTUAL_ENV/slave/twistd.log
```

You should see lines like the following at the end of the worker log:

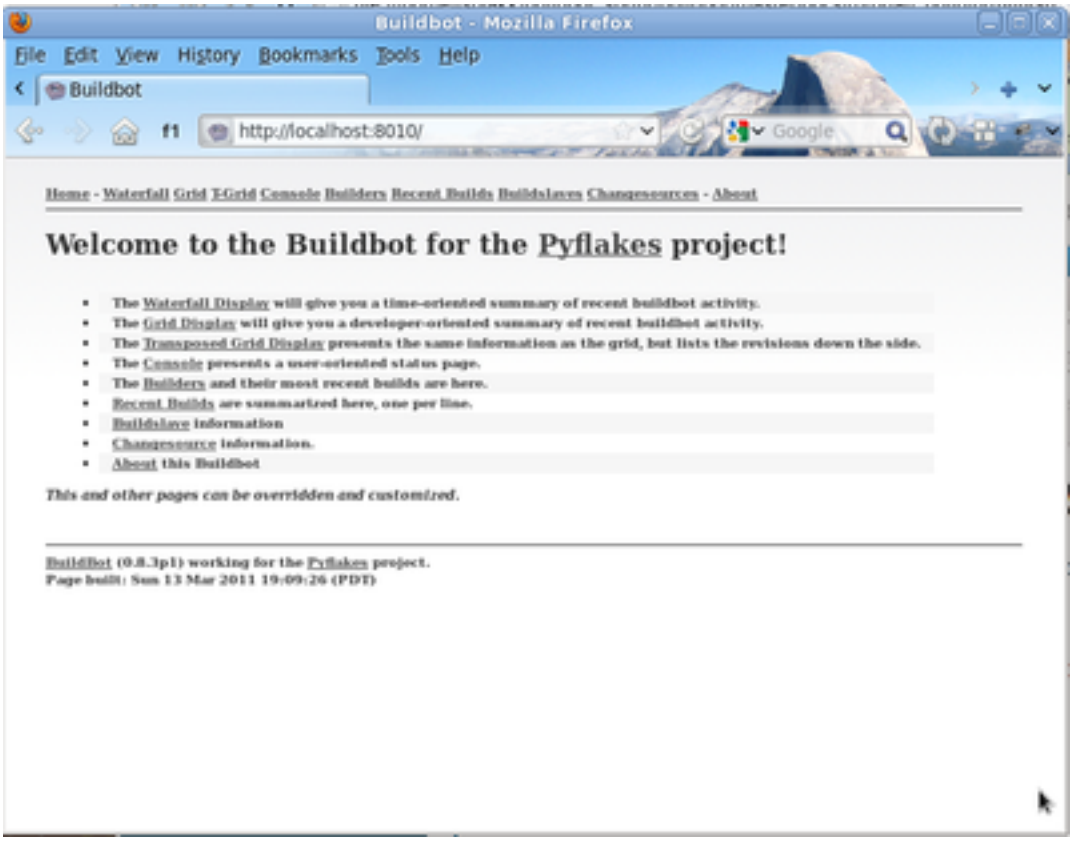
```
2009-07-29 20:59:18+0200 [Broker,client] message from master: attached
2009-07-29 20:59:18+0200 [Broker,client] SlaveBuilder.remote_print(buildbot-full): message from maste
2009-07-29 20:59:18+0200 [Broker,client] sending application-level keepalives every 600 seconds
```

Meanwhile, in the master log, if you tail the log you should see lines like this:

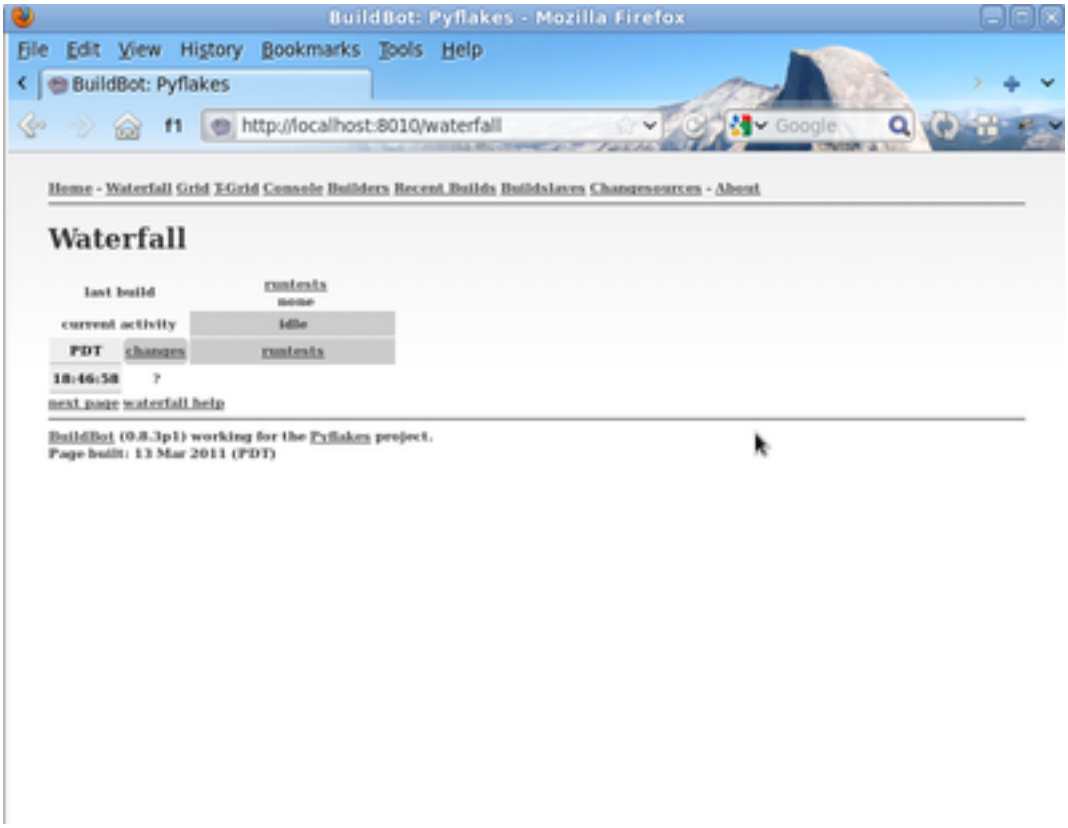
```
tail -f $VIRTUAL_ENV/master/twistd.log

2011-03-13 18:46:58-0700 [Broker,1,127.0.0.1] slave 'example-slave' attaching from IPv4Address(TCP, '
2011-03-13 18:46:58-0700 [Broker,1,127.0.0.1] Got slaveinfo from 'example-slave'
2011-03-13 18:46:58-0700 [Broker,1,127.0.0.1] bot attached
2011-03-13 18:46:58-0700 [Broker,1,127.0.0.1] Buildslave example-slave attached to runtests
```

You should now be able to go to http://localhost:8010, where you will see a web page similar to:



Click on the Waterfall Display link and you get this:



That's the end of the first tutorial. A bit underwhelming, you say ? Well, that was the point! We just wanted to get you to dip your toes in the water. It's easy to take your first steps, but this is about as far as we can go without touching the configuration.

You've got a taste now, but you're probably curious for more. Let's step it up a little in the second tutorial by changing the configuration and doing an actual build. Continue on to *A Quick Tour*

CHAPTER 2

A Quick Tour

2.1 Goal

This tutorial will expand on the *First Run* tutorial by taking a quick tour around some of the features of buildbot that are hinted at in the comments in the sample configuration. We will simply change parts of the default configuration and explain the activated features.

As a part of this tutorial, we will make buildbot do a few actual builds.

This section will teach you how to:

- make simple configuration changes and activate them
- deal with configuration errors
- force builds
- enable and control the IRC bot
- enable ssh debugging

2.2 Setting Project Name and URL

Let's start simple by looking at where you would customize the buildbot's project name and URL.

We continue where we left off in the *First Run* tutorial.

Open a new terminal, and first enter the same sandbox you created before (where $EDITOR is your editor of choice like vim, gedit, or emacs):

```
cd
cd tmp/buildbot
source sandbox/bin/activate
$EDITOR master/master.cfg
```

Now, look for the section marked *PROJECT IDENTITY* which reads:

```
####### PROJECT IDENTITY

# the 'title' string will appear at the top of this buildbot
# installation's html.WebStatus home page (linked to the
# 'titleURL') and is embedded in the title of the waterfall HTML page.
```

```
c['title'] = "Pyflakes"
c['titleURL'] = "http://divmod.org/trac/wiki/DivmodPyflakes"
```

If you want, you can change either of these links to anything you want to see what happens when you change them. After making a change go into the terminal and type:

```
buildbot reconfig master
```

You will see a handful of lines of output from the master log, much like this:

```
2009-08-01 13:38:21+0200 [-] loading configuration from /home/thomas/dev/ext/buildbot/sandbox/master/
2009-08-01 13:38:21+0200 [-] builder buildbot-full is unchanged
2009-08-01 13:38:21+0200 [-] removing IStatusReceiver <WebStatus on port tcp:8010 at 0x11a5290>
2009-08-01 13:38:21+0200 [-] configuration update started
2009-08-01 13:38:21+0200 [-] (Port 8010 Closed)
2009-08-01 13:38:21+0200 [-] Stopping factory <twisted.web.server.Site instance at 0x11a5518>
2009-08-01 13:38:21+0200 [-] adding IStatusReceiver <WebStatus on port tcp:8010 at 0x14f9050>
2009-08-01 13:38:21+0200 [-] twisted.web.server.Site starting on 8010
2009-08-01 13:38:21+0200 [-] Starting factory <twisted.web.server.Site instance at 0x14f90e0>
2009-08-01 13:38:21+0200 [-] WebStatus using (/home/thomas/dev/ext/buildbot/sandbox/master/public_htm
2009-08-01 13:38:21+0200 [-] adding 0 new schedulers, removed 0
2009-08-01 13:38:21+0200 [-] adding 0 new changesources, removing 0
2009-08-01 13:38:21+0200 [-] configuration update complete

Reconfiguration appears to have completed successfully.
```

The important lines are the ones telling you that it is loading the new configuration at the top, and the one at the bottom saying that the update is complete.

Now, if you go back to the waterfall page, you will see that the project's name is whatever you may have changed it to and when you click on the the URL of the project name at the bottom of the page it should take you to the link you put in the configuration.

2.3 Configuration Errors

It is very common to make a mistake when configuring buildbot, so you might as well see now what happens in that case and what you can do to fix the error.

Open up the config again and introduce a syntax error by removing the first single quote in the two lines you changed, so they read:

```
c[title'] = "Pyflakes"
c['titleURL'] = "http://divmod.org/trac/wiki/DivmodPyflakes"
```

This creates a Python SyntaxError. Now go ahead and reconfig the buildmaster:

```
buildbot reconfig master
```

This time, the output looks like:

```
2009-08-01 13:52:23+0200 [-] loading configuration from /home/thomas/dev/ext/buildbot/sandbox/master/
2009-08-01 13:52:23+0200 [-] error while parsing config file
2009-08-01 13:52:23+0200 [-] error during loadConfig
2009-08-01 13:52:23+0200 [-] Unhandled Error
      Traceback (most recent call last):
        File "/home/thomas/dev/ext/buildbot/sandbox/lib/python2.6/site-packages/Twisted-8.2.0-py2.6-l
          reactor.run()
        File "/home/thomas/dev/ext/buildbot/sandbox/lib/python2.6/site-packages/Twisted-8.2.0-py2.6-l
```

```
          self.mainLoop()
        File "/home/thomas/dev/ext/buildbot/sandbox/lib/python2.6/site-packages/Twisted-8.2.0-py2.6-l
          self.runUntilCurrent()
        File "/home/thomas/dev/ext/buildbot/sandbox/lib/python2.6/site-packages/Twisted-8.2.0-py2.6-l
            call.func(*call.args, **call.kw)
        --- <exception caught here> ---
          File "/home/thomas/dev/ext/buildbot/src/buildbot/master.py", line 511, in loadTheConfigFile
            self.loadConfig(f)
          File "/home/thomas/dev/ext/buildbot/src/buildbot/master.py", line 529, in loadConfig
            exec f in localDict
        exceptions.SyntaxError: EOL while scanning string literal (master.cfg, line 191)

2009-08-01 13:52:23+0200 [-] The new config file is unusable, so I'll ignore it.
2009-08-01 13:52:23+0200 [-] I will keep using the previous config file instead.

Reconfiguration failed. Please inspect the master.cfg file for errors,
correct them, then try 'buildbot reconfig' again.
```

This time, it's clear that there was a mistake. in the configuration. Luckily, the buildbot master will ignore the wrong configuration and keep running with the previous configuration.

The message is clear enough, so open the configuration again, fix the error, and reconfig the master.

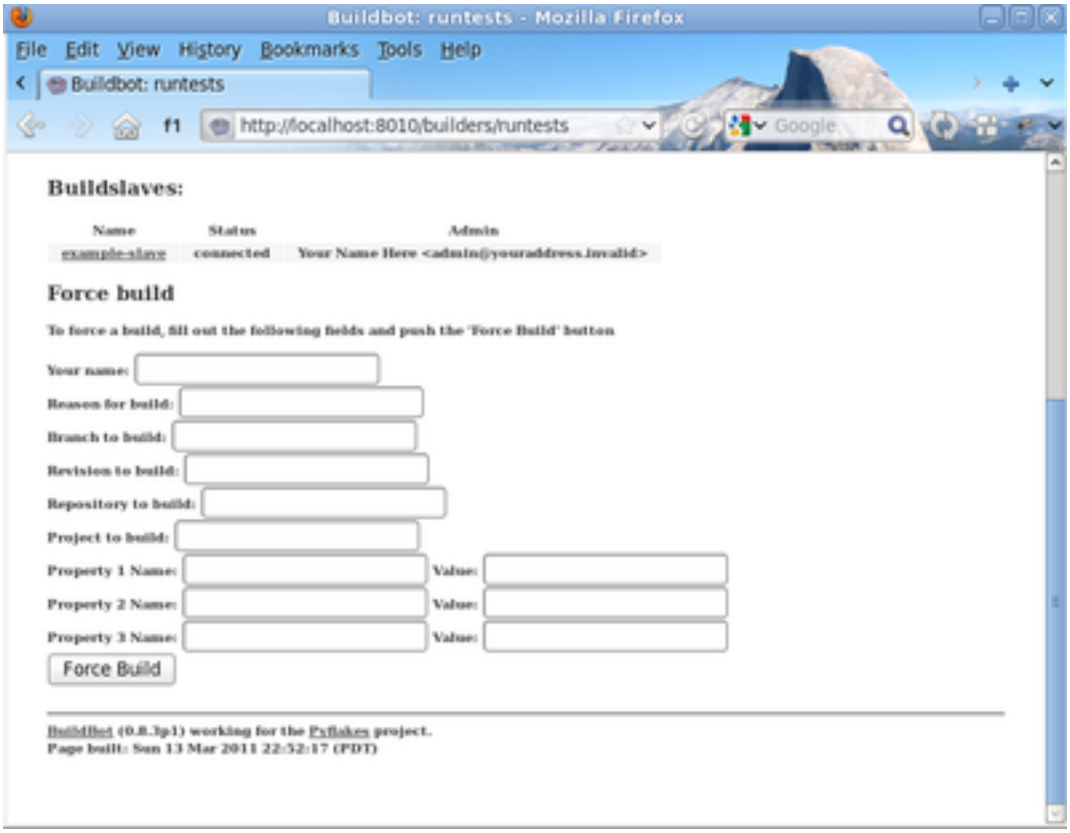
Note that if you are not using 0.8.4 you might experience an error during reconfig with clean configs, this is due to http://trac.buildbot.net/ticket/1757 *which if fixed and will be in the 0.8.4 package. If you come across this, just do start & stop for now to get your buildbot instance updated*:

```
buildbot stop master && buildbot start master
```

2.4 Your First Build

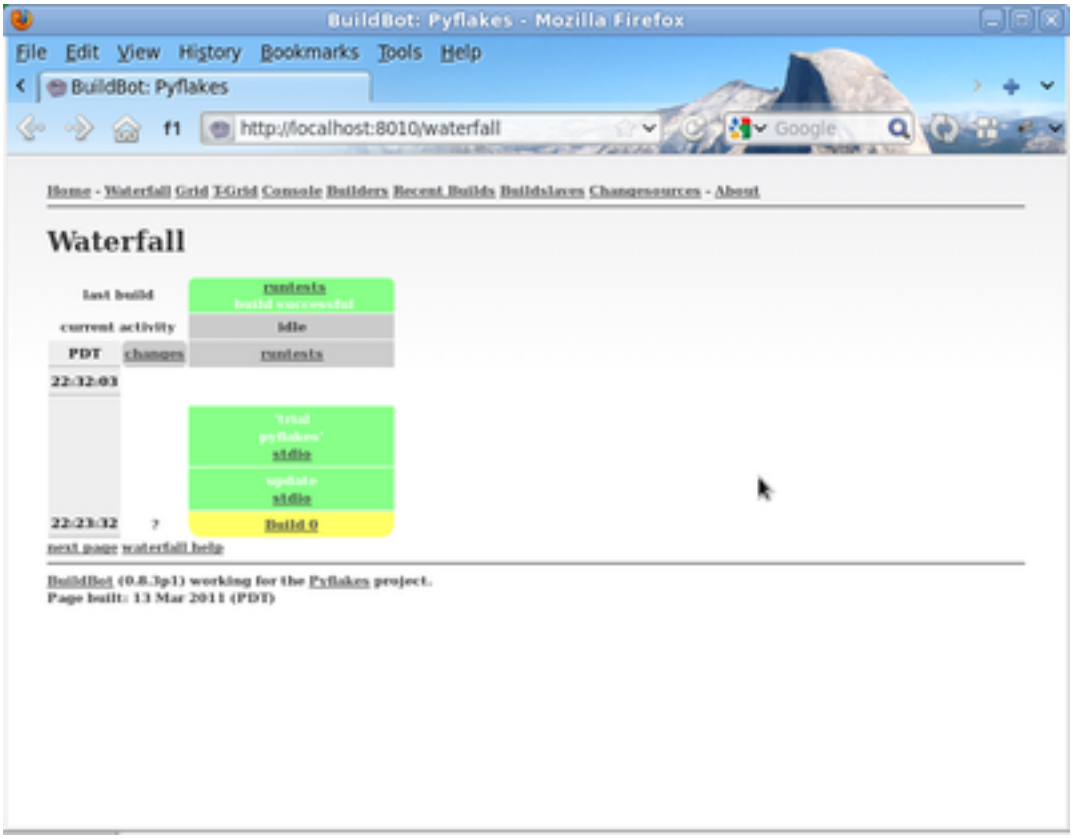
By now you're probably thinking: "All this time spent and still not done a single build ? What was the name of this project again ?"

On the waterfall. page, click on the runtests link, and scroll down. You will see some new options that allow you to force a build:



Type in your name and a reason, then click *Force Build*. After that, click on view in waterfall.

You will now see:



2.5 Enabling the IRC Bot

Buildbot includes an IRC bot that you can tell to join a channel and control to report on the status of buildbot.

First, start an IRC client of your choice, connect to irc.freenode.org and join an empty channel. In this example we will use #buildbot-test, so go join that channel. (*Note: please do not join the main buildbot channel!*)

Edit the config and look for the *STATUS TARGETS* section that you changed before to be able to force the build.

Enter these lines below the WebStatus line in master.cfg:

```
c['status'].append(html.WebStatus(http_port=8010, authz=authz_cfg))

from buildbot.status import words
c['status'].append(words.IRC(host="irc.freenode.org", nick="bbtest",
                             channels=["#buildbot-test"]))
```

Reconfigure the build master then do:

```
cat master/twistd.log | grep IRC
```

The log output should contain a line like this:

```
2009-08-01 15:35:20+0200 [-] adding IStatusReceiver <buildbot.status.words.IRC instance at 0x300d290>
```

You should see the bot now joining in your IRC client. Type:

```
bbtest: commands
```

to get a list of the commands the bot supports.

Let's tell the bot to notify certain events, to learn which EVENTS we can notify on do:

```
bbtest: help notify
```

Now let's set some event notifications:

```
bbtest: notify on started
bbtest: notify on finished
bbtest: notify on failure
```

The bot should have responded to each of the commands:

```
<@lsblakk> bbtest: notify on started
<bbtest> The following events are being notified: ['started']
<@lsblakk> bbtest: notify on finished
<bbtest> The following events are being notified: ['started', 'finished']
<@lsblakk> bbtest: notify on failure
<bbtest> The following events are being notified: ['started', 'failure', 'finished']
```

Now, go back to the web interface and force another build.

Notice how the bot tells you about the start and finish of this build:

```
< bbtest> build #1 of runtests started, including []
< bbtest> build #1 of runtests is complete: Success [build successful]  Build details are at http://l
```

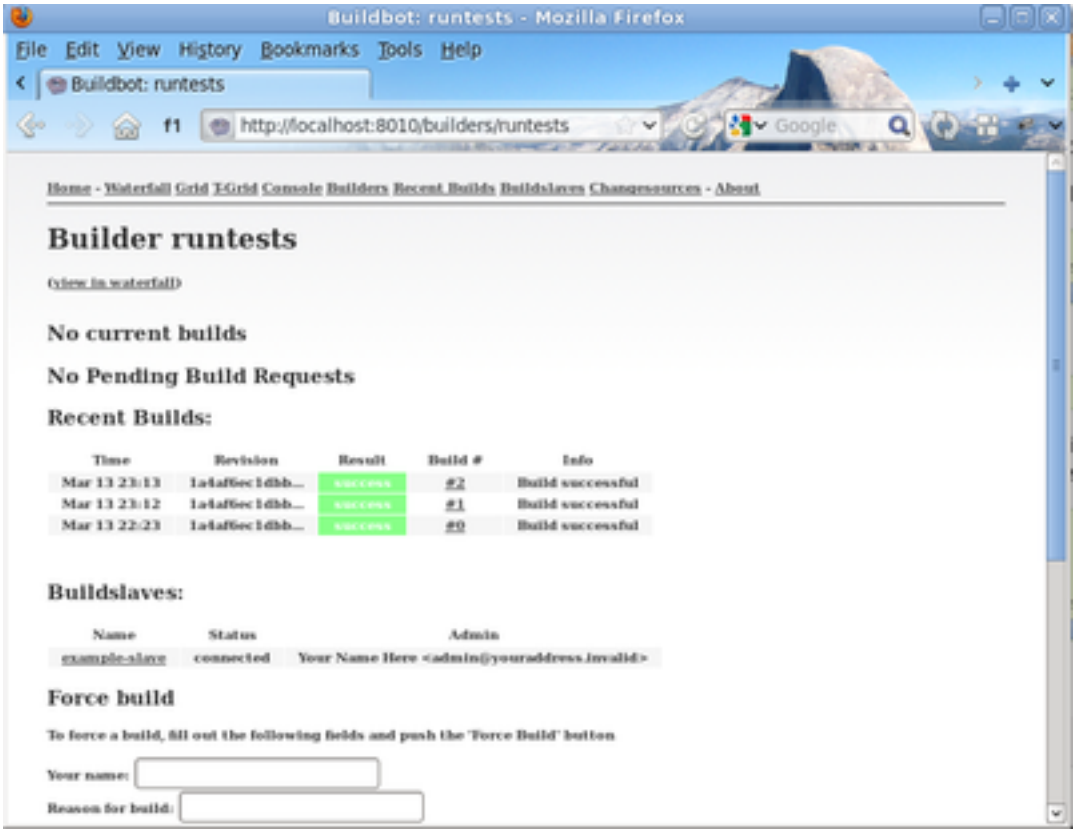
You can also use the bot to force a build:

```
bbtest: force build runtests test build
```

This time, the bot is giving you more output, as it's specifically responding to your direct request to force a build, and explicitly tells you when the build finishes:

```
<@lsblakk> bbtest: force build runtests test build
< bbtest> build #2 of runtests started, including []
< bbtest> build forced [ETA 0 seconds]
< bbtest> I'll give a shout when the build finishes
< bbtest> build #2 of runtests is complete: Success [build successful]  Build details are at http://l
```

You can also see the new builds in the web interface.



2.6 Debugging with Manhole

You can do some debugging by using manhole, an interactive Python shell. It exposes full access to the buildmaster's account (including the ability to modify and delete files), so it should not be enabled with a weak or easily guessable password.

To use this you will need to install an additional package or two to your virtualenv:

```
# make sure your sandbox is activated so this is installed in your virtualenv
pip install pycrypto
pip install pyasn1
```

In your master.cfg find:

```
c = BuildmasterConfig = {}
```

Insert the following to enable debugging mode with manhole:

```
####### DEBUGGING
from buildbot import manhole
c['manhole'] = manhole.PasswordManhole("tcp:1234:interface=127.0.0.1","admin","passwd")
```

Now you can ssh into the master and get an interactive python shell:

```
ssh -p1234 admin@127.0.0.1
# enter passwd at prompt
```

If you wanted to check which slaves are connected and what builders those slaves are assigned to you could do:

```
>>> master.botmaster.slaves
{'example-slave': <BuildSlave 'example-slave', current builders: runtests>}
```

Objects can be explored in more depth using *dir(x)* or the helper function *show(x)*.

CHAPTER 3

Indices and tables

- *genindex*
- *modindex*
- *search*