Programming with Subversion, Trac and Buildbot

version 1.1

Date: June 8, 2006

Olivier Ramonat <olivier.ramonat@netelem.com>
Pascal Obry <pascal@obry.net>

Contents

1 Introduction

This document attempts to desribe a coherent, simple and secure way to install and administrate a suite of project management tools [1][2] :

- Subversion[3] to provide a version control system,
- Trac[4] to get a wiki and an issue/bug tracking system with good integration with Subversion
- and Buildbot to automate the compile/test cycle.

These three tools, installed on the same server, can interact together to improve the quality and speed up software development.

2 How to configure your system

2.1 GNU/Linux and MacOS

2.1.1 Create user accounts

Three user accounts have to be created on the server. This must be done by user *root* with *adduser* command :

```
$ adduser svn
$ adduser trac
$ adduser buildbot
```

2.2 Cygwin/Windows

Subversion, Trac and Buildbot can be used on Windows 2000/XP thanks to Cygwin environment [5].

2.2.1 Install Cygwin

Installing Cygwin is fast and easy :

- Download setup.exe from `http://www.cygwin.com`

[1] Tested on GNU/Linux and MacOs
[2] Tested on Windows/Cygwin, discussed after in this document
[3] Subversion is version control system, the quasi-official replacement for CVS, used by Python, Apache, GCC or KDE for example
[4] Trac is a Sourceforge like
[5] a POSIX emulation layer for Windows

- Install the following modules :
  - Base (selected by default and required)
  - Admin - cygrunsrv
  - Devel - gcc-core
  - Devel - make
  - Devel - Subversion
  - Net - openssh
  - Python - python
  - Utils - rebase
- Launch a Cygwin console and run rebaseall

2.2.2 Create user accounts

Create users svn, trac, buildbot with the standard Windows interface. Create the files '`/etc/passwd`' and '`/etc/group`' from Cygwin command line with :

```
$ mkpasswd -l > /etc/passwd
$ mkgroup -l > /etc/group
```

2.2.3 Configure SSH

This step will allow to remotely connect in a secure way to the Windows computer. It will be used later to connect to the subversion repository.

```
$ ssh-host-config
```

To the questions answer :

- "privilege separation" : *yes*
- "create local user sshd" : *yes*
- "install sshd as a service" : *yes*
- "CYGWIN=" : *ntsec*

2.2.4 Install Trac

Installing Trac is more complex. Cygwin does not provide a binary package and Trac has several dependencies.

- Sqlite

  Download the binary package '`sqlite-2.8.14-1.tar.bz2`' at http://sourceforge.net/project/showfiles.php?group_id=99645&package_id=121049.

```
$ cd /
$ tar xfj sqlite-2.8.14-1.tar.bz2
```

- pysqlite (tested with version 2.1.12)

  To install invoke :

```
$ python setup.py build
$ python setup.py install
```

- Clearsilver (tested with version 0.10.2)

  To install invoke :

```
$ ./configure
$ make
$ make install
```

  You may encounter errors while compiling '`neo_date.c`' on Cygwin. Moreover Cygwin linker is very sensitive to the libraries order. The following patch fixes those problems :

```
*** util/neo_date.c.orig          Mon Mar  6 15:02:23
    2006
--- util/neo_date.c       Mon Mar  6 15:00:17 2006
***************
*** 83,89 ****
--- 83,93 ----
     return ttm->tm_gmtoff;
   #elif defined(HAVE_TZNAME)
     long tz;
+ #ifndef __CYGWIN__
     tz = - timezone;
+ #else
+    tz = - _timezone;
+ #endif
     if(ttm->tm_isdst)
       tz += 3600;
     return tz;
```

```
*** python/setup.py.orig          Mon Mar  6 15:38:03
    2006
--- python/setup.py      Mon Mar  6 15:38:45 2006
***************
*** 5,11 ****

  VERSION = "0.10.2"
  INC_DIRS = ["../"]
! LIBRARIES = ["neo_cgi", "neo_cs", "neo_utl"]
  LIB_DIRS = ["../libs"]
  CC = "gcc"
  LDSHARED = "gcc -shared"
--- 5,11 ----

  VERSION = "0.10.2"
  INC_DIRS = ["../"]
! LIBRARIES = ["neo_cgi", "neo_cs", "neo_utl", "z"]
  LIB_DIRS = ["../libs"]
  CC = "gcc"
  LDSHARED = "gcc -shared"
```

- Trac (tested with version 0.9.3)

  To install invoke :

```
$ python setup.py build
$ python setup.py install
```

Before building Clearsilver, be sure that the Cygwin compiler first appear in your PATH.

2.2.5 Install Buildbot

Cygwin does not provide a package for Buildbot. However the installation is simple and well describe in the INSTALL file provided by Buildbot :

- download the sources archive,
- decompress it,
- build the module(s),

```
$ python setup.py build
```

- install the module(s).

```
$ python setup.py install
```

Before building Twisted and Buildbot, be sure that the Cygwin GCC/C compiler first appears in your PATH.

2.2.6 Run as Windows services

The next step is to run Trac and Buildbot as Windows services to automatically launched them at system start.

First, add c:\cygwin\bin to the system PATH via the Windows standard interface. You need to create a CYGWIN variable with a value of "tty ntsec".

Then use Cygwin *cygrunsrv* tool to create the services. This must be done after the installation of Trac and Buildbot as described in this document.

```
$ cygrunsrv -I buildbot -c /home/buildbot/buildroot -a "--
    no_save -y buildbot.tac" -e CYGWIN="tty ntsec" -p /usr/
    bin/twistd -u buildbot -w buildbot_pwd
```

Start the service :

```
$ cygrunsrv --start buildbot
```

Obtain a description of all parameters :

```
$ cygrunsrv --help
```

2.2.7 Final note

Installation of free tools, developed mainly for GNU/Linux, under Windows with Cygwin is possible and quite simple. However, some adjustments can be necessary according to the computer configuration.

This section main goal is to guide the administrator in his installation by describing the process of a Cygwin installation.

All the following sections are platforms independent.

3 Configure Subversion

All the following commands will be executed by user *svn*

3.1 Create a Subversion repository

Create the repository *my_proj* :

```
$ svnadmin create --fs-type=fsfs my_proj
```

Subversion provide two ways to store its database :

- Berkeley DB
- FSFS (Subversion file system implementation which is the default in recent versions)

Prefer FSFS which is more stable in case of Subversion crash.

3.2 Access to the repository via svn+ssh

All project's users must have a system account and the ability to SSH into the server machine.

To automate connection using scripts, create an RSA key without pass-phrase and use it to authenticate on the server.

Create a local RSA key :

```
$ ssh-keygen -t rsa
```

Copy the public key to *server_addr* :

```
$ ssh-copy-id -i /home/user/.ssh/id_rsa.pub
    user@server_addr
```

On Cygwin ssh-copy-id doesn't exist. You'll have to manually add the key '`/home/user/.ssh/id_rsa.pub`' into user@server_addr:.ssh/authorized_keys.

Then check out project sources *my_proj* :

```
$ svn co svn+ssh://user@server_addr/home/svn/my_proj
```

You do not need to enter your pass-phrase. The connection is automatic and controlled by the SSH RSA key.

3.3 Hook Subversion

Subversion automatically executes the scripts that are placed in the repository's '`hooks`' directory.

3.3.1 pre-commit hook

The script `hooks/pre-commit` is executed before a *commit*. Some conditions can be checked on the committed files. For example, it is possible to test if the files follows the coding guidelines or to check for a valid syntax (if performed by the compiler[6]).

It is possible to prevent SVN check-in unless an *open* Trac ticket number is specified in log message. This cooperation between Trac and Subversion ensures proper tracing of all activities on the repository. Tracing is one of the most important point that leads to software quality.

3.3.2 post-commit hook

The `hooks/post-commit` script is often used to send mail notifications. It can also be used to record the commit log message into the corresponding Trac ticket (section 4.3) or to execute automatic compilations with Buildbot(section 5.1.4).

In the following sections, it is supposed that Subversion repository uses a standard structure :

```
my_proj
  |
  | - trunk
  |
  | - branches
  |     |
  |     | - release_1
  |     | - release_2
```

4 Trac

All the operations will be done by user *trac*

4.1 Create a Trac project

The creation of the project *my_proj* under Trac is done by the tool *trac-admin*.

Project creation under '`/home/trac/rootenv`' :

```
$ trac-admin rootenv/my_proj initenv
Project Name [My Project]> My Project
```

[6] As the -gnatc GNAT's option for example

```
Database connection string [sqlite:db/trac.db]>
Path to repository [/var/svn/test]> /home/svn/my_proj
Templates directory [/opt/local/share/trac/templates]>
```

You only need to name the project and to point to the Subversion repository (on the same server).

4.2 Launch Trac

Trac is launched by :

```
$ tracd -a *,users.htdigest,TracRealm -p 8000 -e /home/trac
    /rootenv/
```

Where users.htdigest is created by :

```
$ htdigest -c users.htdigest TracRealm user
```

Force Trac authentication to have proper tracing of the activities.

The Trac server is running on *my_server* on port 8000. It can be accessed at URL : http://my_server:8000

4.3 From Subversion to Trac tickets

A Trac ticket can be a bug report or a feature request. Trac assigns automatically a number to each ticket.

Use a Subversion hook to link the Subversion transaction to the Trac ticket.

The `scripts/trac-post-commit-hook` python script looks for the string *refs #11, #12* or *fixes #14* in Subversion log and adds the transaction log into Trac ticket history.

When a log contains *fixes* the ticket is automatically marked as closed.

The '`trac-pre-commit-hook`' can reject all transactions that do not refer to a ticket number.

5 Automate with Buildbot

5.1 Configure Buildbot

5.1.1 Requirement

- Twisted http://twistedmatrix.com/trac/wiki/TwistedProject

- Buildbot http://sourceforge.net/projects/buildbot/

Fetch TwistedSumo archive then install :

- ZopeInterface
- Twisted
- TwistedWeb
- TwistedMail (optionnal)

On Debian :

```
$ apt-get install python-twisted python-twisted-web
```

Get buildbot and install it.

Buildbot tested version, Buildbot-0.7.3, does not allow multiple projects on a server. If you want to use multiple projects you have to override a class in master.cfg, (see section 5.1.5). But there are some limitations as it will not be possible to create a page per project.

This feature is expected to be better integrated in future Buildbot versions.

5.1.2 Configure Buildbot server

As buildbot user, run :

```
$ mkdir /home/buildbot/buildroot
$ buildbot master /home/buildbot/buildroot
$ cd /home/buildbot/buildroot
$ cp Makefile.sample makefile
```

Get master.cfg and edit it according to your configuration.

Define the connection parameters : all Buildbot clients must have access to *svnserver*.

```
svnserver = 'svn+ssh://buildbot@my_server/home/svn/'

port = 9989                                    # for slaves
wport = 8010                                   # for web interface
```

Declare Buildbot client : Set a login/password to each buildbot client. Here bot1 and bot2 :

```
# bot-machines used for the builds

c['bots'] = [("bot1", "bot_passwd"),
             ("bot2", "bot_passwd")]
```

Project configuration :

Compile each time a change is committed : Use *RepositoryScheduler* instead of *Scheduler* : a parameter *repository* is defined to allow multiple projects (see 5.1.5).

```
s_trunk=RepositoryScheduler(name="trunk-only",
                            branch="trunk",
                            treeStableTimer=1*60,
                            builderNames=["test-linux",
                                          "test-windows"],
                            repository='/home/svn/test')

s_b1=RepositoryScheduler(name="release-1",
                         branch="branches/release-1",
                         treeStableTimer=1*60,
                         builderNames=["test-linux",
                                       "test-windows"],
                         repository='/home/svn/test')

c['schedulers'].append(s_trunk)
c['schedulers'].append(s_b1)
```

This forces a rebuild each time a change is committed on Subversion. Buildbot waits for the tree to become stable before initiating a build (using timeout *treeStableTimer*).

Note that Buildbot pays attention to branches. The above example has been define to only consider the *trunk* and *branches/release-1*.

Commands to run : If your project url is *baseURL*, compilation process is done by *make build* and tests run by *make test*, you have to write :

```
test_steps = [s(step.SVN, mode="update", baseURL=svnserver
    + 'test/'),
              s(step.Configure),
              s(step.Compile),
              s(step.Test, command=["make", "check"])]
```

```
test_f = factory.BuildFactory(test_steps)
```

Define the builders : The project *test* has two slaves *bot1* and *bot2*. They appear as *test-linux* and *test-windows* in Buildbot web interface.

```
bot1_builder = {'name': "test-linux",
      'slavename': "bot1",
      'builddir': "test-linux",
      'factory': test_f,
      }

bot2_builder = {'name': "test-windows",
      'slavename': "bot2",
      'builddir': "test-windows",
      'factory': test_f,
      }

c['builders'].append (bot1_builder)
c['builders'].append (bot2_builder)
```

Daily tests : A build is triggered every day at 11h45 PM.

```
test_nightly=Nightly('test-nightly', ['test-linux'],
                          hour=23, minute=45, branch='trunk')
c['schedulers'].append (test_nightly)
```

Notification : A notification of builder events will be sent to admin@my_server.

```
c['status'].append(mail.MailNotifier
                   (builders=['test-linux', 'test-windows'
                       ],
                    fromaddr="buildbot@my_server",
                    extraRecipients=["admin@my_server"])
```

Launch the server :

```
$ make start
```

5.1.3 Create a Buildbot slave

Module *python-twisted-web* is not required for Buildbot slave.

```
$ buildbot slave buildroot server_addr:9989 <name> <passwd>
```

Where *server_addr* is the server address, default port is 9989. Slave login/password is set in server configuration.

WARNING : Buildbot slave must be able to SSH into the server automatically (use an RSA key as described in section 3.2 page 8).

5.1.4 From Subversion to Buildbot

`scripts/trac-post-commit-hook` python script must be called in `hooks/post-commit` to notify Buildbot server that a change has been committed in Subversion.

5.1.5 Multiples projects

Create a new class *RepositoryScheduler* that overrides Buildbot *Scheduler*.

When Buildbot receives a notification from the Subversion hook, it will only rebuild the corresponding project.

```
class RepositoryScheduler(Scheduler):
    """Extend Scheduler to allow multiple projects"""
        def __init__(self, name, branch, treeStableTimer,
            builderNames,
                    repository, fileIsImportant=None):
            """
            Override Scheduler.__init__

            Add a new parameter : repository
            """
                Scheduler.__init__(self, name, branch,
                    treeStableTimer,
                                builderNames,
                                    fileIsImportant)
                self.repository = repository

        def addChange(self, change):
            """Call Scheduler.addChange only if the
               repository is modified"""
                # check to make sure the repository matches
                    !
                cs = change.comments.split(' ')
                if len(cs) > 0:
                        repo = cs[0]
                        log.msg('checking %s vs %s' %
                                (repo, self.repository))
```

```
                if repo != self.repository:
                        log.msg("%s ignoring
                            repository %s" %
                                (self, repo))
                        return
        # call our parent since this on the correct
            repository
        Scheduler.addChange(self, change)
```

6 Appendix

6.1 Subversion hooks

scripts/svn-hook-support.sh script contains functions for *pre-commit* and *post-commit* hooks.

6.1.1 pre-commit

Function check_style checks style and syntax of modified files. Uses Style_Checker (http://www.obry.net/contrib.html).

Function log_not_empty checks if log is not empty.

Function trac_pre_commit_record_log checks if a log message contains a reference to a Trac ticket number. Uses scripts/trac-pre-commit-hook.

6.1.2 post-commit

Function trac_post_commit_record_log appends Subversion log to Trac ticket history. Uses scripts/trac-post-commit-hook.

Function send_mail_post_commit sends a mail at each commit.

Function buildbot_post_commit sends a notification to Buildbot server. Usesscripts/svn_buildbot.py.

6.1.3 Examples

Two hooks scripts are provided :

- `hooks/pre-commit`
- `hooks/post-commit`

6.2 Buildbot

You can find an example of Buildbot server configuration here : `scripts/buildbot/master.cgf`.