Buildbot: A continuous integration system

Krzysztof Voss

January, 2013
Outline

• Testing and Continuous Integration

• Introduction to Buildbot

• BuildMaster

• BuildMaster components

• BuildSlave

• Installation and Usage
Testing and continuous integration

Tests:

• the best specification

• safety-net for refactoring

• bug identification

Tests are the most effective if we:

• run them often

• run them on different machines/environments

• can easily see their results
The most straightforward approach would entail:

• logging in to different machines
• fetching the newest source code
• running tests
• analyzing their output

In case we want to test a few environments, repeating the above steps is tedious.

Developers do not focus on the code, instead they run tests.

A continuous integration system performs all of these steps for us, so developers can focus on their code.
Introduction to Buildbot: Features

- run builds on a variety of BuildSlave platforms
- arbitrary build process: handles projects using C, Python, ...
- minimal host requirements: python and Twisted
- BuildSlave can be behind a firewall if they can still do checkout
- status delivery through web page, email, IRC, other protocols
- track builds in progress, provide estimated completion time
- flexible configuration by subclassing generic build process classes
• debug tools to force a new build, submit fake Changes, query BuildSlave status

• released under the GPL

source: http://buildbot.net/buildbot/docs/current/manual/introduction.html
Introduction to Buildbot: Overview

system overview

source: http://buildbot.net/buildbot/docs/0.8.1/full.html
BuildMaster components

source: http://buildbot.net/buildbot/docs/0.8.1/full.html
BuildMaster

BuildMaster:

- holds the configuration of the entire system.
- has information about all BuildSlaves.
- periodically checks the version control system to see if any new changes are committed to a repository.
- can watch more than one repository.

Example configuration file
Feedback

tests passed

BuildMaster Components

**Definition. [Scheduler]** A Scheduler can be periodic or can be run on events like a commit. There is also a ForceScheduler, so we can explicitly run builds.

After a Scheduler adds a BuildRequest to a Builder queue, BuildSlaves are given command to perform their BuildSteps.

**Definition. [Builder]** A Builder is a pair of a list of BuildSlaves and a Factory.

**Definition. [Factory]** Factory object has a list BuildSteps specifying how to perform a build.

For example:

- pull changes from a repository
- run script to prepare environment variables
• run tests in directory A

• run tests in directory B
BuildSlave

BuildSlave knows only about its BuildMaster. It connects to it through TCP. Because it is initiating the connection, it can stay behind a NAT.

It also initiates a connection to a version control repository.

It is a good practice to add a command for starting a BuildSlave to a crontab. That is useful in case your system was subject to an unexpected reboot.
Installation and Usage

virtualenv --no-site-packages sandbox
source sandbox/bin/activate
easy_install buildbot
easy_install buildbot-slave
To create a BuildMaster:
buildbot create-master master
To run it:
buildbot start master
To create a BuildSlave that connects to a BuildMaster running on localhost:9989:
buildslave create-slave slave localhost:9989 example-slave pass
To run a BuildSlave:
buildslave start slave
Resources

• Buildbot website

• Travis website

• Github