www.spkaa.com
Ph: 888-310-4540

SPK and Associates
900 E Hamilton Ave, Ste.100
Campbell, CA 95008

Continuous Integration in 8 Easy Steps with Buildbot

Several weeks ago, Ron provided some great insight regarding how to install the Hudson continuous integration tool on Windows. This week, I'll be discussing a different tool: Buildbot.

My initial exposure to continuous integration tools in general was only a few years ago, when one of our customers had requested an Apache Continuum installation. It worked well for the single Java application that they were developing -- however, it included many features that we did not end up using. In particular, we never leveraged its role-based security or its release management features. Fast-forward a couple years later, and the customer's development environment has changed significantly.

The Java app is now a handful of smaller C++ and Python based projects. Source Code Management has changed several times -- from CVS to SVN, then to Bazaar, and then finally GIT -- and builds are now being managed by Buildbot.

I was apprehensive to change, since I knew nothing about Buildbot. It turns out however, that Buildbot's simplicity is one of its biggest strengths. Buildbot's architecture is a simple master-slave setup. Slaves reside on one or more machines and simply wait for instructions from the master. The master hosts the configuration file that defines who the slaves are, what the build schedules are, and where to pull the source code from.

In the 8 simple steps below, you will learn how Buildbot helps with your automated builds:

1. Create the master

```
%>  Buildbot create-master /buildbot/buildmaster/
```

This creates our initialmaster directory structure, along with a sample configuration file to get us started.

2. Define the slaves

```
 ####### BUILDSLAVES
from Buildbot.buildslave import BuildSlave

c['slaves'] = [BuildSlave("builder", "password123")]

c['slavePortnum'] = 9999
```

www.spkaa.com
Ph: 888-310-4540

SPK and Associates
900 E Hamilton Ave, Ste.100
Campbell, CA 95008

Although the slaves are listeners, they authenticate with the master prior to establishing a connection. We define a shared secret so that other slaves can't connect.

3. Define where the source code will be pulled from

```
####### CHANGESOURCES

from Buildbot.changes.gitpoller import GitPoller

c['change_source'] = GitPoller(

'ssh://build@code.widgetsinc.local/home/git/repo/Platform.git',

pollinterval=1200)
```

Our change source is a GIT repository. Unfortunately there's no way for Buildbot to know when a change has been committed to the repo, so we poll for changes on a regular interval. When a change is seen, we can trigger a build to start.

4. Define the schedules

```
####### SCHEDULERS

c['schedulers'] = []

c['schedulers'].append(Scheduler(name="onsubmit",

branch="master",

treeStableTimer=2*60,

builderNames=["MyProject submit"]

)

)

c['schedulers'].append(timed.Nightly(name="nightly-widgetsinclibs-build",

hour=15, minute=15,

builderNames=["Daily widgetsinclibs"]
```

www.spkaa.com
Ph: 888-310-4540

SPK and Associates
900 E Hamilton Ave, Ste.100
Campbell, CA 95008

```
)
```

In addition to builds being triggered via polling, we can define a regularly scheduled build.

5. Define the builders

```
#########   BUILDERS

from Buildbot.process.factory import BuildFactory

from Buildbot.steps.source import Git

from Buildbot.steps.shell import ShellCommand

f_MyProject = factory.BuildFactory()

f_MyProject.addStep(shell.Compile(command="cd ..; rm -rf build", timeout=600))

f_MyProject.addStep(Git(repourl=MyProjectroot, mode="update"))

f_MyProject.addStep(shell.Compile(command="cd workingdir; export COMPILE_FLAGS=1; make
clean -j9; bash -e ./test.sh; omake --no--progress", timeout=6600))

f_MyProject.addStep(shell.Compile(command="/usr/local/bin/git-tag", timeout=600))

f_MyProject.addStep(shell.Compile(command="git remote add origin
ssh://build@code.widgetsinc.local/home/git/repo/Platform.git; git push --tags",
timeout=600))

c['builders'] = [

{'name':'MyProject submit',

'slavename':'localhost',

'builddir':'5.0.0_MyProject',

'factory':f_MyProject

},
```

www.spkaa.com
Ph: 888-310-4540

SPK and Associates
900 E Hamilton Ave, Ste.100
Campbell, CA 95008

```
{'name':'Daily MyProject release candidate',

'slavename':'localhost',

'builddir':'5.0.0_MyProject_rc',

'factory':f_MyProject_nightly

},
```

Our builders let us define how we want the builds to run. For instance, we may need some prior setup, as well as some post-build steps that we need to complete. In our situation, we wanted to tag the build in GIT.

We used some simple shell scripts as build steps to accomplish this.

6. Setup notifiers

```
c['status'].append(MailNotifier

(builders=['MyProject submit', 'Daily MyProject release candidate', 'Daily
widgetsinclibs'],

fromaddr='code-review@widgetsinc.com',

lookup='widgetsinc.com',

extraRecipients=['msolinap@widgetsinc.com'],

sendToInterestedUsers=False))
```

Once the build is complete, we can define how we want to be notified.

7. Create the slave(s)

```
%>  buildslave create-slave /Buildbot/buildslave localhost:9999 builder password123
```

Our configuration is complete, and we can tell Buildbot to create the working directory structure, and to start listening for the master process.

8. Start Buildbot!

www.spkaa.com
Ph: 888-310-4540

SPK and Associates
900 E Hamilton Ave, Ste.100
Campbell, CA 95008

```
%>  Buildbot start /Buildbot/buildmaster
%>  buildslave start /Buildbot/buildslave
```

Once Buildbot is running, it can be managed via the web GUI available at http://localhost:8888/. Your initial builds may appear to fail, but you can easily debug them by looking at each of your build step's output. Each build step must complete with a successful return code, or else the build will display as failed. Once you've modified your build, you can force an immediate rebuild, instead of waiting for a trigger or schedule to fire.

Hopefully this helps you get up and running quickly with Buildbot. Its flexibility allows for a number of customizations that let you tailor it for your specific development process. Have an interesting build situation? Let me know!

Mike Solinap

Sr Systems Integrator

SPK & Associates