Particulars of the use of buildbot for ABINIT

Triggering the execution of the automatic tests

Although the ETSF test farm might be accessed independently of buildbot, the usual mode of interaction with the test farm is through the buildbot system. In the following, we suppose that you are a well-identified developer of ABINIT, hence, you have your own git branches.

There are two ways to trigger the tests on the test farm:

- a merge request of your branch to trunk/develop will trigger nightly (the bunch starts around 10:30pm European time) the launch of automatic tests on all the “nightly” slaves;
- a general “on-demand” Web page on bbportal. You can contact Xavier if you think you are sufficiently expert to use it, provided (of course) you need it for your development.

You will receive by mail the results of the execution of the tests, concerning each nightly slave, one for each slave. Congratulation if your compilation succeeded, all the tests succeeded, etc!

How to handle failures?

Let’s now analyze what to do in case of failure. The mail from buildbot is rather self-explanatory, and one is quickly lead, through hyperlinks (“further details”) to a list of the steps that were executed by buildbot, the failing one being indicated in red.

If it is step “10 make_core”, you have to click on the “10.2 make” or “10.3 stderr” hyperlink, and you will see the usual log/err file produced by the compiler. In case you would like to have direct access on the machine, this is possible, see below.

If it is step “12 testsuite”, you have to click on the “12.4 summary” hyperlink, to identify whether tests “passed” or “failed”, and to identify the test directories where such things happened. Remember: on the reference machine “abiref”, all tests must “succeed”, while on the other machines, no one test can “fail”, but a “pass” is OK. More execution information is found in the “12.2 xreport” hyperlink.

In order to correct the test in the other series, use “12.5 All Results” hyperlink to identify precisely the tests that must be corrected.

The links “12.1 stdio” and “12.3 testbot” are not very often used in the debugging process by standard users, but more by the maintainers.

12.1 stdio = the stdout of buildbot (Usually NOT useful for debugging)
12.2 xreport = detailed output of execution
12.3 testbot = output of the configuration use to drive // tests (Usually NOT useful for debugging)
12.4 summary = a short table, summarizing the status (succeeded, passed, failed) of the tests, per directory
12.5 All Results = all results grouped by number of "cpu" (np=1,2,4,10,...).
In order to correct the test in the “22. abirules” or “23. buildsys” series executed with abiref GNU 5.3_debug, the “diff” hyperlink will likely be the most useful for you to correct your source code. Perhaps you will need also to complete the information from “22.4 log” by a look to the “20.2 make” hyperlink. Do not hesitate to contact Xavier Gonze to find some reasonably elegant way to get rid of the “unused variable” or “unused arguments” problems in delicate cases.

**Accessing directly some slave.**

If you have git branches, you are entitled to have interactive access to each of the slaves. Moreover, you can login as “buildbot” user, and place yourself as if you were the one who just made the run reported in the mail that you received. Hence, you are directly placed in an environment ready to debug.

1. Login on a gateway (well, the first time, contact Jean-Michel Beuken, to allow your machine to connect to the private network).
   
   ssh USERNAME@hall.abinit.org

2. To take the hand over buildbot, execute:
   
   sudo su - buildbot

3. Now, you need to access other machines of the testfarm, use the supplementary step
   
   ssh <name_of_the_machine>
   
   where <name_of_the_machine> might be abiref, ubu, max2, etc. e.g. ssh abiref
   
   The first part of the name of the bot is the name_of_the_machine.

4. cd ABINIT
5. cd to the directory for the particular builder that you want to debug (the machine might support more than one builder) ( ex: cd abiref GNU 9.2_openmpi )
6. cd to the directory for the particular branch that you want to debug (the one bearing your name) ( ex : cd trunk_develop )
7. Use the command “module load NAME_OF_BUILDER” to select the compiler that you want to use (type “module avail” to see the list of options).

   Try to solve your problem ... ! You can modify the files, recompile, etc ...

8. The slave can handle usual git commands. So you might make modifications, commit and push to your branch. Of course, after commit and push, it is also advised to test your modifications on your own machine, before using again the test farm, although this is no mandatory.