Breaking Selected Packages out of Trilinos and Importing other Packages

Motivations, concerns, workflows, examples, ...

Roscoe A. Bartlett

Oak Ridge National Laboratories

October 14, 2014

Outline, Motivations, and Concerns

Outline:

- Motivations for breaking packages out of Trilinos and hosting on github
- Concerns/challenges when breaking out a Trilinos package
- The TriBITS move to github using a snapshotting approach
- Comparing snapshotting and submodules workflows for splitting out Kokkos
- Variations on version control and snapshotting
- Applications to Trilinos?

Motivations:

- Smooth development and integration with projects and customers only needing a single package (or small number of packages)
- Encourage contributions outside of SNL and outside of Trilinos
- Take advantage of github features like Issues, pull requests, restricted push access
- Moving your changes around through Trilinos requires updating all Trilinos packages! (Trilinos is large and hard to push changes and takes an hour of work and half a day in many cases to push changes due problems in ssg/master).

Concerns/Challenges:

- Keeping packages on github synced with the rest of Trilinos and each other
- Ensuring existing use cases and workflows for Trilinos by developers and customers are not disturbed or damaged.

Current status of Trilinos

- Many important (paying) Trilinos customers never touch a Trilinos release branch or a release tarball
  - Examples: SIERRA, CASL VERA
- Many important (paying) Trilinos customers do automated (daily) testing of their software against Trilinos dev versions in ssg/master
  - Examples: SIERRA, Alegra, Xyce, CASL VERA, internal CRADA
- External clones of Trilinos are maintained for some projects taking advantage of distributed version control: Example, CASL VERA:
  - CASL VERA maintains a Trilinos clone on casl-dev.ornl.gov.
  - Changes to Trilinos for CASL made directly to casl-dev/master
  - Updates of Trilinos from ssg/master but only made after detailed analysis (see http://trac.trilinos.org/wiki/VERAIntegrationTriBITSTrilinos#trilinos_cdash_examination)
  - CASL VERA customers and collaborators pull Trilinos from casl-dev/master, not ssg/master.
  - SCALE maintains a mercurial copy of Trilinos git repo!

Like it or not, Trilinos git repo is the deployment mechanism for may important Trilinos customers!

Trilinos is Big

Source repository size: 2.0G

```
$ du -sh Trilinos/
2.0G    Trilinos

$ cd Trilinos
$ du -sh * | sort -rh
1.2G    packages
20M     cmake
14M     doc
2.0M    commonTools
1.4M    SIERRA
1.2M    demos
952K    sampleScripts
…
```

```
$ cd Trilinos/packages/
$ du -sh * | sort -rh

210M    zoltan
79M     seacas
63M     teko
56M     muelu
46M     mesquite
41M     Sundance
26M     stk
20M     ml
20M     belos
17M     xpetra
16M     epetra
15M     zoltan2
14M     moocho
13M     tpetra
…
```

Testing many ST packages:

```
Final set of enabled packages:
TriBITS … Panzer Sundance
Optika TrilinosCouplings 46

Final set of non-enabled
packages:  ThreadPool Pliris
Claps Amesos2 Trios ShyLU
TriKota STK Phdmesh Aristos
CTrilinos ForTrilinos
PyTrilinos WebTrilinos Didasko
NewPackage Mesquite
MeshingGenie FEApp 19
```

Build Directories: 52G!

```
$ cd BUILDS/CHECKIN/
$ du -sh * | sort -hr | less
29G     MPI_DEBUG_ST
16G     MPI_DEBUG
3.8G    SERIAL_RELEASE_ST
2.6G    SERIAL_RELEASE
```

Checkin test PT and some ST packages: 3 hours 40 minutes (16 processes)

```
0) MPI_DEBUG => passed: passed=1591,notpassed=0 (48.56 min)
1) SERIAL_RELEASE => passed: passed=1581,notpassed=0 (28.06 min)
2) MPI_DEBUG_ST => passed: passed=1973,notpassed=0 (101.09 min)
3) SERIAL_RELEASE_ST => FAILED: passed=1909,notpassed=1 => Not ready to push! (42.82 min)
```

The TriBITS move to github using a snapshotting approach

Directory Structure and Initial Setup: TriBITS Devs

Directory Structure (Snapshotting)

```
~/Trilinos.base/
  Trilinos/
    cmake/tribits/  # snapshotted dir
    TriBITS/        # cloned from github
  BUILDS/
    CHECKIN/
      checkin-test-<machine>.sh
    TRIBITS_CHECKIN/
      checkin-test.sh
  GCC-4.6.1/
      MPI_DEBUG/
        do-configure
```

See full process at:

http://trac.trilinos.org/wiki/TriBITSTrilinosDev

Initial setup for TriBITS devs (Snapshotting)

```
$ cd ~/Trilinos.base/
$ git clone software.sandia.gov:/space/git/Trilinos
$ cd Trilinos/
$ git clone git@github.com:TriBITSPub/TriBITS
$ echo TriBITS > .gitdist  # Optional
```

Updating to develop TriBITS (Snapshotting)

```
$ cd ~/Trilinos.base/Trilinos/
$ git pull
$ cd TriBITS/
$ git pull
```

Updating to develop TriBITS using gitdist (Snapshotting)

```
$ cd ~/Trilinos.base/Trilinos/
$ gitdist pull
```

Local TriBITS develop, publish, test, and push

Local dev after update (Snapshotting)

```
$ cd ~/Trilinos.base/Trilinos/TriBITS/
  * make changes to TriBITS *
$ cd ~/Trilinos.base/Trilinos/
  * make changes to rest of Trilinos *
$ cd ~/Trilinos.base/BUILDS/GCC-4.6.1/MPI_DEBUG/
$ ./do-configure ... && make && ctest
$ cd ~/Trilinos.base/Trilinos/TriBITS/ ; git commit
$ cd ~/Trilinos.base/Trilinos/ ; git commit
  # Iterate above steps until ready to push
```

Configure, build, test (Snapshotting)

```
$ cd Trilinos.base/BUILDS/GCC-4.6.1/MPI_DEBUG/
$ ./do-configure \
   -DTrilinos_TRIBITS_DIR:STRING=TriBITS \
   -DTrilinos_ENABLE_TriBITS=ON
$ make -j8 && ctest -j8
```

- NOTE: Develop against the TriBITS version cloned from github *not* the one snapshotted in Trilinos!

Publish, test, and push (Snapshotting)

```
# Publish TriBITS changes to Trilinos
$ cd ~/Trilinos.base/Trilinos/cmake/tribits/
$ ../../TriBITS/snapshot-dir.py # Does commit!
  # NO MERGE CONFLICITS!!!
# Test and push to Trilinos
$ cd ~/Trilinos.base/BUILDS/CHECKIN/
$ ./checkin-test-<mymachine>.sh --do-all #Must pass!
$ cd ../TRIBITS_CHECKIN/
$ ./checkin-test.sh --do-all --push
$ cd ../CHECKIN/
$ ./checkin-test-<mymachine>.sh --push
```

- **checkin-test.py uses snapshotted cmake/tribits/**

- Host TriBITS on github (issues, pull requests, external usage, etc.)
- Allows co-development of TriBITS with Trilinos (keep in sync)
- Guarantees that the TriBITS version snapshotted into Trilinos works with Trilinos
- Provides traceability of TriBITS versions in Trilinos
- Allow for automatic merging when two or more TriBITS change sets are pushed around the same time to Trilinos
- **Not break existing tools and processes for non-TriBITS Trilinos developers and users!!!**
- Maintain distributed version control and mirroring for Trilinos **and TriBITS**
- Don't force everyone to have to clone TriBITS from github to avoid firewall issues on some systems (e.g. ORNL, WEC, etc.)
- Allow for small local changes to TriBITS (especially in mirrored clones of Trilinos not at SNL)
- Allow for small infrequent changes to TriBITS directly in Trilinos to fix urgent problems.

Minor (rare) changes to TriBITS in Trilinos

Motivation:

- Some critical build is broken when is breaking Trilinos for everyone and needs to be fixed ASAP!
- Someone finds some small typos and just wants to fix it.

Non-TriBITS developer making small/rare change to TRiBITS

```
$ cd ~/Trilinos.base/Trilinos/cmake/tribits/
  * make some small change to some tribits file *
$ git commit –m "TriBITS: Some small fix"
$ git push    # using checkin-test.py
```

Apply small/rare change to TriBITS

```
$ cd ~/Trilinos.base/Trilinos/
$ git pull
$ git format-patch -1 -o ../patches <sha1>
../patches/0001-TriBITS-some-small-fix.patch
$ cd ~/Trilinos.base/Trilinos/TriBITS/
$ git pull
$ $ git am -p3 \
  ../../patches/0001/TRiBITS-some-small-fix.patch
```

Reject a small/rare change to TriBITS

- **Do nothing**. The next snaphsot of TriBITS will wipe out the change. No merge conflicts!!!!
- But change can always be pulled off at any time later with git format-patch and applied so it is never lost!

This process has already been executed at least once since TriBITS was moved to github and snapshotting started being used!

Trilinos commit:

```
58552d5 "Tribits: Fixed spelling error ("disaled")
in check-in test script."
Author: Mark Hoemmen <mhoemme@sandia.gov>
Date:   Wed Sep 10 17:00:23 2014 -0600 (4 weeks
ago)
```

TriBITS commit:

```
e7ae2af "Tribits: Fixed spelling error ("disaled")
in check-in test script."
Author: Mark Hoemmen <mhoemme@sandia.gov>
Date:   Wed Sep 10 17:00:23 2014 -0600 (4 weeks
ago)
```

More substantial changes to TriBITS (github)

Non-TriBITS developer proposes non-trivial change to TriBITS (github workflow)

- Get a account on github
- Fork TriBITS git repo on github to your github account
- Clone github TriBITS repo off of github.com/TriBITSPub/TriBITS
- Create remote in local github clone of TriBITS pointing to your forked github repo of TriBITS
- Create new local topic branch for the change
- Make the change in the local topic branch
- Push the topic branch to your github repo
- Make a pull request for the new branch on github.com/TriBITSPub/TriBITS

- This process has already been followed may times by Nico S. and Ross B.
- See examples at: https://github.com/TriBITSPub/TriBITS/pulls?q=is%3Apr+is%3Aclosed

TriBITS pusher/developer, fetch, merge, review, and fix issues (github workflow)

```
$ cd ~/Trilinos.base/Trilinos/
$ git pull
$ cd TriBITS/
$ git pull
$ git remote add <other-user>-github \
    git@github.com:<other-user>/TriBITS
$ git fetch <other-user>-github
$ git merge <other-user>-github/<topic-branch>
```

- Review commits, make new commits to fix issues, etc.
- Snapshot changes to Trilinos and push to github and Trilions as with any regular change to TriBITS.

Version Traceability of TriBITS in Trilinos Snapshot

TriBITS snapshot commit in Trilinos:

```
$ cd ~/Trilinos.base/Trilinos/
$ git log -- cmake/tribits/

commit 4cf1d72a1602dc4bf57fb18b8e10e22d6b5ecf44
Author: Roscoe A. Bartlett <bartlettra@ornl.gov>
Date:   Sat Oct 11 12:12:53 2014 -0400

    Automatic snapshot commit from TriBITS at a5cada6

    Origin repo remote tracking branch: 'origin/master'
    Origin repo remote repo URL: 'origin = git@github.com:TriBITSPub/TriBITS'

    At commit:

    a5cada6 Added support for <PACKAGE_NAME>_SOURCE_DIR_OVERRIDE
    Author: Roscoe A. Bartlett <bartlettra@ornl.gov>
    Date: Sat Oct 11 12:10:39 2014 -0400
```

- Snapshot commits created automatically using the TriBITS snapshot-dir.sh script:

```
$ cd ~/Trilinos.base/Trilinos/cmake/tribits/
$ ../../TriBITS/snapshot-dir.py
```

- Provides all the necessary info to trace back to TriBITS origin repo:
  - Git SHA1: a5cada6 (WARNING: This can change if repo is later filtered!)
  - Summary: Added support for <PACKAGE_NAME>_SOURCE_DIR_OVERRIDE
  - Author: Roscoe A. Bartlett <bartlettra@ornl.gov>
  - Author Date: Sat Oct 11 12:10:39 2014 -0400

Seeing Corresponding Commits in TriBITS

TriBITS snapshot commits in Trilinos:

```
$ cd ~/Trilinos.base/Trilinos/
$ git log --oneline -- cmake/tribits/
…
4cf1d72 Automatic snapshot commit from TriBITS at a5cada6
8fbbce2 Automatic snapshot commit from TriBITS at 9918d90
…
```

Corresponding commits in TriBITS:

```
$ cd ~/Trilinos.base/Trilinos/TriBITS/
$ git log-short --name-status a5cada6 ^9918d90

a5cada6 "Added support for <PACKAGE_NAME>_SOURCE_DIR_OVERRIDE"
Author: Roscoe A. Bartlett <bartlettra@ornl.gov>
Date:   Sat Oct 11 12:10:39 2014 -0400 (18 hours ago)

M       doc/examples/UnitTests/CMakeLists.txt
M       package_arch/TribitsProcessPackagesAndDirsLists.cmake

4feb929 "Regenerated"
Author: Roscoe A. Bartlett <bartlettra@ornl.gov>
Date:   Sat Oct 11 12:10:25 2014 -0400 (18 hours ago)

M       doc/build_quick_ref/TribitsBuildQuickRef.html
M       doc/build_quick_ref/TribitsBuildQuickRef.pdf
M       doc/build_quick_ref/TribitsBuildQuickRef.rst

09ed04c "Removing datatype from cache vars, adding warning about XXX_ENABLE_YYY vars"
Author: Roscoe A. Bartlett <bartlettra@ornl.gov>
Date:   Sat Oct 11 12:08:21 2014 -0400 (18 hours ago)

M       doc/build_quick_ref/TribitsBuildQuickRefBody.rst
```

Comparing Snapshotting and Submodules Workflows for Splitting out Kokkos

Directory Structure and Initial Setup: Kokkos Devs

Directory Structure (Snapshotting)

```
~/Trilinos.base/
  Trilinos/
    imported_packages/kokkos/   # snapshotted dir
    kokkos/                     # cloned from github
  BUILDS/
    CHECKIN/
      checkin-test-<machine>.sh
  GCC-4.6.1/
      MPI_DEBUG/
        do-configure
```

Directory Structure (Submodules)

```
~/Trilinos.base/
  .gitsubmodules
  Trilinos/
    packages/kokkos/  #cloned from github
  BUILDS/
    CHECKIN/
      checkin-test-<machine>.sh
  GCC-4.6.1/
    MPI_DEBUG/
      do-configure
```

Initial setup for Kokkos devs (Snapshotting)

```
$ cd ~/Trilinos.base/
$ git clone software.sandia.gov:/space/git/Trilinos
$ cd Trilinos/
$ git clone git@github.com:<orgname>/kokkos
$ echo kokkos > .gitdist  # Optional
```

Initial setup for Kokkos devs (Submodules)

```
$ cd ~/Trilinos.base/
$ git clone --recursive
    software.sandia.gov:/space/git/Trilinos
```

Updating to develop Kokkos (Snapshotting)

```
$ cd ~/Trilinos.base/Trilinos/
$ git pull
$ cd kokkos/
$ git pull
```

Updating to develop kokkos (Submodules)

```
$ cd ~/Trilinos.base/Trilinos/
$ git pull
$ git submodule update --init --recursive
  # ABOVE: Throws you off all of your branches!
# Get back on your branch!
$ cd packages/kokkos/
$ git checkout master  # or some other branch?
$ git pull
```

- git submodule update **kicks you off your local Kokkos branch**! As a Kokkos developer, I don't know why you would want to put up with the above behavior?

Updating to develop Kokkos using gitdist (Snapshotting)

```
$ cd ~/Trilinos.base/Trilinos/
$ gitdist pull
```

- **Keeps you on your local Kokkos branch**!
- Scalable to multiple repos!

Local Kokkos develop, publish, test, and push

Local dev after update (Snapshotting)

```
$ cd ~/Trilinos.base/Trilinos/kokkos/
  * make changes to kokkos *
$ cd ~/Trilinos.base/Trilinos/
  * make changes to rest of Trilinos *
$ cd ~/Trilinos.base/BUILDS/GCC-4.6.1/MPI_DEBUG/
$ ./do-configure ... && make && ctest
$ cd ~/Trilinos.base/Trilinos/kokkos/
% git commit …
  # Iterate above steps until ready to push
```

Local dev after update (Submodules)

```
$ cd ~/Trilinos.base/Trilinos/packagse/kokkos/
  * make changes *
$ cd ~/Trilinos.base/Trilinos/
  * make changes to rest of Trilinos *
$ cd Trilinos.base/BUILDS/GCC-4.6.1/MPI_DEBUG/
$ ./do-configure ... && make && ctest
$ cd ~/Trilinos.base/Trilinos/packages/kokkos/
% git commit …
  # Iterate above steps until ready to push
```

Configure, build, test (Snapshotting)

```
$ cd Trilinos.base/BUILDS/GCC-4.6.1/MPI_DEBUG/
$ ./do-configure
   -DKokkos_SOURCE_DIR_OVERRIDE=kokkos \
   -DTrilinos_ENABLE_Kokkos=ON
$ make -j8 && $ ctest -j8
```

- NOTE: Develop against the Kokkos version cloned from github *not* the one snapshotted in Trilinos!

Configure, build, test (Submodules)

```
$ cd Trilinos.base/BUILDS/GCC-4.6.1/MPI_DEBUG/
$ ./do-configure \
   -DTrilinos_ENABLE_Kokkos=ON
$ make -j8 && $ ctest -j8
```

Publish, test, and push (Snapshotting)

```
# Publish Kokkos changes to Trilinos
$ cd Trilinos/imported_packages/kokkos/
$ ../../kokkos/snapshot-dir.py # Does commit!
  # NO MERGE CONFLICITS!!!
# Test and push to Trilinos
$ cd ~/Trilinos.base/BUILDS/CHECKIN/
$ ./checkin-test-<mymachine>.sh --do-all #Must pass!
$ cd ~/Trilinos.base/Trilinos/kokkos/
$ git push # Or use checkin-test.py for kokkos
$ cd ~/Trilinos.base/BUILDS/CHECKIN/
$ ./checkin-test-<mymachine>.sh --push
```

- **Will merge** correctly inside of Trilinos pull if there is another change to Kokkos merged in!
- **checkin-test.py uses snapshotted packages/kokkos/**

Publish, test, and push (Submodules)

```
# Publish Kokkos changes to Trilinos
$ cd ~/Trilinos.base/Trilinos/
$ git add kokkos
$ git commit …
# Test and push to Trilinos
$ cd ~/Trilinos.base/BUILDS/CHECKIN/
$ ./checkin-test-<mymachine>.sh --do-all #Must pass!
$ cd ~/Trilinos.base/Trilinos/packages/kokkos/
$ git push # Or use checkin-test.py for kokkos
$ cd ~/Trilinos.base/BUILDS/CHECKIN/
$ ./checkin-test-<mymachine>.sh --push
```

- **Will result in merge conflict** and stop test and/or push if any change for Kokkos is pushed by another developer after the checkin-test.py script starts running!
- Requires TriBITS tool checkin-test.py to be updated to work with submodules!

Non-Kokkos Trilinos dev/user access, test, and push

Initial setup devs/users (Snapshotting)

```
$ cd Trilinos.base/
$ git clone software.sandia.gov:/space/git/Trilinos
```

Initial setup devs/users (Submodules)

```
$ cd Trilinos.base/
$ git clone --recursive
  software.sandia.gov:/space/git/Trilinos
```

- The need to call the new --recursive option will break many existing tools (e.g. TriBITS, SIERRA's build, etc.)
- The hidden clone of Kokkos on github.com will fail on many machines with restricted internet access (e.g. ORNL and WEC).

Updating to dev/install (Snapshotting)

```
$ cd Trilinos.base/Trilinos/
$ git pull
```

Updating to dev/install (Submodules)

```
$ cd Trilinos.base/Trilinos/
$ git pull
$ git submodule update --init --recursive
```

- Needing command git submodule update --init --recursive will break many tools (e.g. TriBITS, SIERRA's build, etc.) and the hidden clone/pull to github will fail on many systems with restricted access (e.g. ORNL and WEC).

Local dev after update (Snapshotting)

```
$ cd ~/Trilinos.base/Trilinos/
  * make changes to rest of Trilinos *
$ cd ~/Trilinos.base/BUILDS/GCC-4.6.1/MPI_DEBUG/
$ ./do-configure ... && make && ctest
$ cd ~/Trilinos.base/Trilinos/
% git commit …
  # Iterate above steps until ready to push
```

Local dev after update (Submodules)

```
$ cd ~/Trilinos.base/Trilinos/
  * make changes to rest of Trilinos *
$ cd Trilinos.base/BUILDS/GCC-4.6.1/MPI_DEBUG/
$ ./do-configure ... && make && ctest
$ cd ~/Trilinos.base/Trilinos/packages/kokkos/
% git commit …
  # Iterate above steps until ready to push
```

Test and push (Snapshotting)

```
$ cd ~/Trilinos.base/BUILDS/CHECKIN/
$ ./checkin-test-<mymachine>.sh --do-all #Must pass!
$ ./checkin-test-<mymachine>.sh --push
```

Test and push (Submodules)

```
$ cd ~/Trilinos.base/BUILDS/CHECKIN/
$ ./checkin-test-<mymachine>.sh --do-all #Must pass!
$ ./checkin-test-<mymachine>.sh --push
```

Miscellaneous issues with git submodules (git 2.1.0)

Pulling updated repos with "git submodule update --init –recursive" throws you off your branch!

$ git help submodules # git 2.1.0

update

Update the registered submodules, i.e. clone missing submodules and checkout the commit specified in the index of the containing repository. This will make the submodules HEAD be detached …

$ git help pull # git 2.1.0

BUGS

Using --recurse-submodules can only fetch new commits in already checked out submodules right now. When e.g. upstream added a new submodule in the just fetched commits of the superproject the submodule itself can not be fetched, making it impossible to check out that submodule later without having to do a fetch again. This is expected to be fixed in a future Git version.

http://codingkilledthecat.wordpress.com/2012/04/28/why-your-company-shouldnt-use-git-submodules/

"There's a reason why I know a lot of people who have nicknamed these things "sobmodules" in their frustration."

http://stackoverflow.com/questions/6714785/git-submodule-alternative

"Basically, submodules should be your last choice when dealing with your own code. They're great for dealing with third party libraries, but end up being a royal pain for your own code."

Snapshotting vs. Submodules Workflows: Kokkos

Requirements for splitting out and developing Kokkos with Trilinos:	Snapshotting	Submodules
1. Host Kokkos on github (issues, pull requests, external usage, etc.)	√	√
2. Allow co-development of Kokkos with Trilinos (keep in sync)	√	√
3. Guarantee that Kokkos in Trilinos works with all of Trilinos	√	√
4. Provide traceability of Kokkos versions in Trilinos	√	√
5. Not break existing tools and processes for non-Kokkos Trilinos developers and users	√	X
6. Maintain distributed version control and mirroring for Trilinos **and Kokkos**	√	X
7. Preserves old versions of Trilinos when Kokkos github repo goes away or is filtered.	√	X
8. Don't force everyone to have to clone Kokkos from github to avoid firewall issues on some systems (e.g. ORNL, WEC, etc.)	√	X
9. Allow for small local changes to Kokkos (especially in mirrored clones of Trilinos not at SNL)	√	X
10. Allow for small infrequent changes to Kokkos directly in Trilinos to fix urgent problems.	√	X
11. Allow for automatic merging when two or more Kokkos change sets are pushed around the same time to Trilinos	√	X
12. Robust to forgetting to push Kokkos to github	√	X
13. Scalable workflow to multiple external packages (stay on branch)	√	X
14. Not forcing usage of what is widely considered a controversial and problematic git feature (i.e. submodues) on all Trilinos developers and users!	√	X

Variations on version control and snapshotting

Snapshotting in Branch then Merge: E.g. MOOSE

Branches in casl-dev/MOOSE.git repo:

- inl_clean_svn: Direct snaphot commits for MOOSE SVN repo
- master: Local changes and merges from inl_clean_svn

Updating MOOSE snapshot:

```
$ cd MOOSE
$ git fetch origin
$ git checkout --track origin/inl_clean_svn
$ ./create_snapshot_commit    # update from current MOOSE SVN repo
$ git push                    # push to origin/inl_clean_svn
$ git checkout master
$ git pull
$ git merge inl_clean_svn     # deal with merge conflicts if they occur
$ git push                    # push to origin/master (TEST FIRST!)
```

Example:

```
commit bacba1ed219d9fba4a50132a3173efcf3f469d18 (master~2^2~1^2)
Author: moosetest <moosetest@dd8cd9ef-2931-0410-98ca-75ad22d19dd1>
Date:   Tue May 28 15:03:57 2013 +0000

r18996 | permcj | 2013-05-28 08:46:46 -0600 (Tue, 28 May 2013) | 1 line
    Holy Warnings Batman! refs-#1777
```

- Advantage: Can maintain local changes without overwriting ☺
- Disadvantage: May have to deal with merge conflicts ☹
- REALITY: Your only choice when you cannot affect changes in origin repo!

Issues and Solutions for Snapshotting Workflow

- Mistakenly making changes in the snapshotted dir (Trilinos/packages/kokkos/)?
  - ⇒ Make Trilinos/imported_packages/<packageDir>/ (kokkos/) read-only
    - snapshot-dir.sh script adjust the permissions automatically
    - Reminder to non-package devs not to change?
    - Still allow people to make files readable and change if critical.
  - ⇒ Use gitolite to restrict pushes to only certain people

- Forgetting to update snapshot before running checkin-test.py?
  - ⇒ Add check to package (Kokkos) dev's checkin-test.py wrapper to abort if Trilinos/imported_packages/<packageDir>/ and Trilinos/<packageDir>/ diff?
  - ⇒ Add support for snapshotting to TriBITS itself?
  - ⇒ If a backward compatible change to the package (Kokkos) is not snapshotted, then there will be nothing changed and no push will occur!
  - ⇒ If a non-backward compatible change is made (to Kokkos) requiring changes to downstream packages, then build/tests will fail with checkin-test.py and no push will occur!

Adding Support for Snapshotting to TriBITS?

- Add --snapshotted-packages=<pkg0>:<dir1>,… argument to checkin-test.py?

```
$ checkin-test.py \
    --snapshotted-packages=TriBITS:TriBITS,Kokkos:kokkos,...
    --do-all --push
```

- After pull from repos but before configure, for each <pkgi>:

```
$ cd <base-dir>/Trilinos
$ ./cmake/tribits/core/python/snapshot_dir.py --orig-dir=<diri>/ \
    --dest-dir=<pkgi_native_dir>/
  # Or if package has its own specialized snapshot script
$ cd <diri>
$ <base-dir>/Trilinos/<pkgi_native_dir>/update_snapshot
```

- If any snapshot fails for any reason (dirty dirs, etc.) then abort!

Applications to Trilinos?

Classifications of Trilinos Packages?

- **Internal Package:** Trilinos/packages/<packageDir>/
  - Developed exclusively inside of the main Trilinos git repo and exports not supported
  - Examples: NOX, AztecOO, Stratimikos, …
- **Exported Package**: Trilinos/packages/<packageDir>/
  - Developed natively in Trilinos git repo
  - Supports snapshotting out to other projects/repos
  - Examples: Zoltan, Teuchos
- **Imported Package:**
  - Developed primarily/exclusively in an external git repo and then imported into Trilinos
  - **Snapshotted Package**: Trilinos/imported_packages/<packageDir>/
    - Typically has no (or few) upstsream package dependencies
    - Source copied (rsync) into Trilinos and committed
    - Examples: TriBITS, Kokkos?, Zoltan?, DataTransferKit?
  - **Inserted Package**: Trilinos/<extraRepo>/<packageDir>/
    - Package listed Trilinos/PackagesList.cmake but source kept in external git repo
    - Not required for pre-push testing (Secondary Tested, ST)
    - Examples: Packages in preCopyrightTrilinos?, MOOCHO?, DataTransferKit?
- **Extra Package**: Trilinos/<extraRepo>/<packageDir>/
  - Developed and managed in an external repository and pulled in as extra TriBITS repo
  - Has no downstream package dependencies in main Trilinos repo
  - Examples: Sundance? Mesquite?

All these packages are included in Trilinos post-push CI an Nighty testing and tarball releases of Trilinos!

Restricting push access for Trilinos Packages?

Gitolite Basics:

- Special account "git" controls access to repos under /home/git/repositories/
- Users register public ssh keys: Public SSH key => <userid>
- Access to repos using git@trilinos.org:<repo-name>
- Flexible repo access rules based on gitolite groups
- Repo git@trilinos.org:gitolite-admin: SSH keys, group definitions and repo access rules:

```
gitolite-admin/
    keysdir/
    conf/gitolite.conf
```

Advantages:

- Provide repo access without providing accounts on the machine
- Define access groups right in gitolite.conf
- User can see repos and permissions using ssh git@trilinos.org info
- Flexible access control by repo, by directory, etc.
- Force code reviews before pushing to some important packages (Teuchos, Epetra, …)
- Supports custom git push hooks (e.g. use our existing git custom hooks)
- Add new repos by adding to them to gitolite.conf and pushing

Disadvantages:

- Some initial setup

Recommendations:

- Set up gitolite on trilinos.org and define reasonable push restrictions
- Provide read-only clone on github (pull request, user facing issues)

Value Proposition for Trilinos?

1. Trilinos provides a (almost) continuously integrated collection of software packages that can be built in a single build or can be built and installed in chunks of packages of various sizes. (This is what TriBITS will allow soon.)

2. Trilinos defines a consistent lifecycle model and quality metrics that provides better aligned expectations between developers and users and supports usage of Trilinos packages from basic research projects through usage of selected packages in high-consequence applications. (This is the TriBITS lifecycle model.)

3. Trilinos provides automated testing of Trilinos packages on a number of platforms and configurations. (This is a big deal and a big driver to add new packages to Trilinos.)

4. Trilinos provides a well-defined and efficient release process that delivers integrated components on a regular schedule or through stable branches in the public repo. (This is also a big for some customers.)

THE END