Handling Git Push Problems

Problem recap: you work on a ticket, commit, and attempt to push, only to be greeted with:

Try #1

```
$ git push
....
To /Users/mjuric/projects/tmp/meas_algorithms
   de7c4d7..849f074  tickets/1831 -> tickets/1831
 ! [rejected]        master -> master (non-fast-forward)
error: failed to push some refs to '/tmp/meas_algorithms'

To prevent you from losing history, non-fast-forward updates were
rejected. Merge the remote changes (e.g. 'git pull') before pushing
again.  See the 'Note about fast-forwards' section of 'git push --help'
for details.
```

Then, as the error message says, you try a **`git pull`**:

Try #2

```
$ git pull
Already up-to-date.
$ git status
# On branch tickets/1831
nothing to commit (working directory clean)
```

So try **`git push`** again, but:

Try #3

```
$ git push
To /Users/mjuric/projects/tmp/meas_algorithms
 ! [rejected]        master -> master (non-fast-forward)
error: failed to push some refs to
'/Users/mjuric/projects/tmp/meas_algorithms'
To prevent you from losing history, non-fast-forward updates were
rejected. Merge the remote changes (e.g. 'git pull') before pushing
again.  See the 'Note about fast-forwards' section of 'git push --help'
for details.
```

and you are left utterly confused (especially if you haven't touched the master at all).

The problem occurs because of how **git** push/pull work, combined with the poorly chosen default behavior of **`git push`**, compounded by the misleading (incomplete) error message.

By default, **`git push`** attempts to sync *all* branches upstream, not just the current branch. Here "sync" means "upload all necessary changes, and make the tip of the remote branch the same as the tip of the local branch." Secondly, this is *not* atomic -- some branches may sync successfully, others may fail. That is what happens in #1 above. The syncing of `tickets/1831` succeeds, but the syncing of master doesn't. That's why in #3 there is no message about `tickets/1831` any more.

The sync of `master` failed because someone else has pushed a new commit to the remote:

```
Remote       Local
master:      master:

D
|
C            C
|            |
B            B
|            |
A            A
```

As **git** translates your **`git push`** quite literally as '*Do everything necessary to make the commit C the tip of the remote master*', it refuses to do so as there's no fast-forward path from D to C. And that is what it (tersely) reports.

So while **git** advises you to do **`git pull`**, it does not warn you that **`git pull`** checks *only the current branch*. That is why in # 2 above, **`git pull`** will say that everything is up to date (when the `master` clearly isn't), because **`git push`** has *succeeded* for `tickets/1831` in # 1.

So there's a subtle, annoying asymmetry between **`git pull`** and **`git push`**. Knowing all that, the workaround is easy:

```
$ git checkout master
Switched to branch 'master'
Your branch is behind 'origin/master' by 4 commits, and can be
fast-forwarded.
$ git pull
Updating d9ae10d..eedb85f
Fast-forward
 python/lsst/meas/algorithms/utils.py |   12 +++++++-----
 src/shapes/SdssShape.cc              |   20 ++++++++++++++--------
 2 files changed, 19 insertions(+), 13 deletions(-)
$ git push
Everything up-to-date
```

Or, just ignore the **`git push`** error message for branches you're not working on.

Can anything be done about this? The root cause of the problem is the unexpected asymmetry of **`git pull`** and **`git push`**. The former syncs down only the current branch, while the latter attempts to sync up all branches. Everything would be much better if either

- **`git pull`** fetched+merged all branches or
- **`git push`** would push only the current branch upstream

The former cannot be done. The latter can be:

```
git config --global push.default tracking
```

Note: **`--global`** will set this as default for all repos that you work with on that account/machine.

So if we try it now:

```
$ git push
Counting objects: 3, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 224 bytes, done.
Total 2 (delta 1), reused 0 (delta 0)
Unpacking objects: 100% (2/2), done.
To /Users/mjuric/projects/tmp/meas_algorithms
   849f074..7d0918c  tickets/1831 -> tickets/1831
```

Now there are no more annoying messages about branches we're not working on (master). Voilà!

The downside is that now you have to manually push each branch upstream, but, realistically, this *is* what one wants 99.99% of the time. It's unclearn why this isn't the default; perhaps it's related to history and backwards compatibility.

Helpful on-line references:

- http://longair.net/blog/2011/02/27/an-asymmetry-between-git-pull-and-git-push/
- http://stackoverflow.com/questions/832222/why-is-git-pushing-to-two-branches-in-this-git-push