Submodules

Creating Submodules

This is the equivalent to SVN externals. Submodules are subdirectories of a Git project that point to another Git project.

Starting in an existing Git repository, add a submodule.

```
git submodule add git://somehost/someproject.git mysubproj
```

This will create a subfolder named `mysubproj` in the current directory and a new file called `.gitmodules` that contains mappings to the repositories.

Git tracks the submodule and the parent module separately. To change, update, or modify the submodule, `cd` into its directory.

Git helps the parent project precisely track the commit (not a symbolic `master` or `HEAD`) hash that the submodule is at for this parent project. This commit point can be different than another parent project that points to this submodule at a different commit point.

Updating Submodule Pointers

Start any submodule updating by executing `cd <SUBMODULE>` to navigate into the submodule directory.

If there's new code to be pulled, it can be retrieved via `git fetch`. There's no need to merge it into a branch as submodules are meant to represent a point in time and a hash (sans branch or tag name) is exactly that — a point in time.

To visualize if all the new code has been retrieved, type `git log --all -5` top see the last five commits regardless if they are merged into any local branch. Copy the hash of the desired submodule snapshot.

Next, `git checkout <HASH>` to the desired commit. The submodule's working directory now reflects the new snapshot state.

Lastly, `cd ..` back to the master repository hosing the submodule. The `git status` command will show that the subdirectory has been modified. `git add <SUBMODULE>` will add the submodule's update HASH pointer to the staging area. Git commit will seal it into the master repository.

Restoring Submodules

To check out a project and set up the submodules:

```
git clone <git://url/yourrepo.git>
cd <yourrepo>
# Setup the submodule .gitmodules file
git submodule init
# Retrieve the submodules' code
git submodule update
```

If another developer changes the submodule, then commits the parent project and you pull and retrieve that, you'll need to update the submodules again to stop your `git status` from telling you the tree is dirty.

```
git submodule update
```

The main concept is just to put the submodule in whatever state you want it to be in, then add and commit from the parent project. If a colleague does that and then you pull the parent project, you need to `update` to get the latest submodule code.

Coding in Submodules

Remember that submodule code changes need to be committed and pushed from within the submodule repository. In fact, it is better if submodule coding happens from a separate clone of the repository. If a submodule-coding developer fails to push updated code to the team-accessible host for the submodule code, her colleagues won't be able to retrieve the commits; the parent project will be pointing to a submodule hash that isn't publicly available.