

Bitbucket Direct Collaboration

[OBSOLETE] Thoughts about supporting direct (and informal) collaboration without the need for pull requests.

- [Introduction](#)
- [Alternatives](#)
- [Experiment](#)
- [Details](#)
- [Addendum](#)

Introduction

[John Blackford](#) and I were talking about how best to use Bitbucket to support direct informal collaboration between members without the need for pull requests. For example a member creates some code and wants other members to be able to see it, use it, comment on it, modify it etc. This code is a "contribution" not it's not yet been discussed or approved by the group.

Alternatives

There are various ways to achieve this but some work better than others.

Approach	Discussion
Access collaborators' forks directly	This works, but forks are private by default, so their owners will have to allow access to the repositories (which could include work that isn't to be shared). In order to scale, one collaborator would need to be nominated as the "primary" one.
Use pull requests to collaboration branches on the project repository	This works, but requires the formality of a pull request. Until the pull request has been approved and merged, collaborators can't see the code.
Use direct access to collaboration branches on the project repository	This works well. Collaborators create an "upstream" remote to the project repository and can push/pull directly to/from collaboration branches. Branch permissions prevent direct access to the master, develop, feature/xxx and release/xxx branches.



Experiment

I adjusted the [WT-369 project repository's branch permissions](#). For those of you who can't access them, they now look like this. The key point is that only the branches that we care about (i.e. that are part of our process) are restricted. So we can freely create and use collaboration branches (I suggest branches called `collab/xxx`).

Branch permissions

[Add permissions](#)

With branch permissions you can control the actions users can perform on a single branch, branch type or branch pattern. [Learn more](#)

Branch	Prevent	Exemptions
 master	Rewriting history	
	Deletion	
	Changes without a pull request	
	All changes	
 develop Development branch from branching model	Rewriting history	
	Deletion	
	Changes without a pull request	
	All changes	
feature/ Feature branches from branching model	Rewriting history	
	Deletion	
	Changes without a pull request	
	All changes	
release/ Release branches from branching model	Rewriting history	
	Deletion	
	Changes without a pull request	
	All changes	

Details

To make use of this you need to go to your fork's local clone and add an `upstream` remote that points back to the project repository (this is the usual convention for such a remote).

```
% git remote -v
origin      ssh://git@code.broadband-forum.org:7999/~wlupton_broadband-forum.org
/wt-369.git (fetch)
origin      ssh://git@code.broadband-forum.org:7999/~wlupton_broadband-forum.org
/wt-369.git (push)

% git remote add upstream ssh://git@code.broadband-forum.org:7999/usp/wt-369.git

% git remote -v
origin      ssh://git@code.broadband-forum.org:7999/~wlupton_broadband-forum.org
/wt-369.git (fetch)
origin      ssh://git@code.broadband-forum.org:7999/~wlupton_broadband-forum.org
/wt-369.git (push)
upstream    ssh://git@code.broadband-forum.org:7999/usp/wt-369.git (fetch)
upstream    ssh://git@code.broadband-forum.org:7999/usp/wt-369.git (push)
```

Now create a collaboration branch, do some work, and commit it. Here I've called the branch `collab/feature/for-discussion` in the expectation that it might eventually become a `feature/for-discussion` branch, but the name really doesn't matter (as long as it doesn't match one of the names already used by our process).

```
% git checkout develop
Switched to branch 'develop'
Your branch is up-to-date with 'origin/develop'.

% git checkout -b collab/feature/for-discussion
Switched to a new branch 'collab/feature/for-discussion'

% vi for-discussion.txt
% git add for-discussion.txt
% git commit -m"Added 'for-discussion' discussion for discussion"
[collab/feature/for-discussion f734c89] Added 'for-discussion' discussion for discussion
1 file changed, 1 insertion(+)
create mode 100644 for-discussion.txt
```

Now push the collaboration branch to the project repository.

```
% git push upstream
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 316 bytes | 0 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote:
remote: Create pull request for collab/feature/for-discussion:
remote:   https://code.broadband-forum.org/projects/USP/repos/wt-369/compare
/commits?sourceBranch=refs/heads/collab/feature/for-discussion
remote:
To ssh://git@code.broadband-forum.org:7999/usp/wt-369.git
* [new branch]      collab/feature/for-discussion -> collab/feature/for-discussion
```

The really nice thing is that, assuming that all the forks were created with "Enable fork synching" checked (this is the default), the collaboration branch will now be immediately propagated to all the forks!

Addendum

You might be wondering what will happen now that the `collab/feature/for-discussion` branch is available via both of my remotes, i.e. in the project repository via my `upstream` remote and in my fork via my `origin` remote. Let's take a look.

```
% git branch --all | grep for-discussion
* collab/feature/for-discussion
remotes/origin/collab/feature/for-discussion
remotes/upstream/collab/feature/for-discussion
```

That's perfect. We get to choose which remote to push to, and we can also set an explicit upstream branch. However, one thing is worrying me: why does `git push upstream` work but `git push origin` not work? I can't see which piece of config is causing this. Any ideas?

```
% git push origin
fatal: The current branch collab/feature/for-discussion has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin collab/feature/for-discussion

% git push upstream
Everything up-to-date

% git config --list | grep origin
remote.origin.url=ssh://git@code.broadband-forum.org:7999/~wlupton_broadband-forum.org/wt-369.git
remote.origin.fetch=+refs/heads/*:refs/remotes/origin/*
branch.master.remote=origin
branch.develop.remote=origin
branch.feature/re-generate-usp-proto-python.remote=origin
branch.feature/fix-readme-files-for-new-structure.remote=origin
% git config --list | grep upstream
remote.upstream.url=ssh://git@code.broadband-forum.org:7999/usp/wt-369.git
remote.upstream.fetch=+refs/heads/*:refs/remotes/upstream/*
```