Pushing / Merging Branches

Pushing

If we run ... `git push <remote> serverfix`
Git automatically expands the `serverfix` branchname out to `refs/heads/serverfix:refs/heads/serverfix`, where the sytax is `local:remote`..

You can use this same syntax when pusshing a local branch into a remote branch that is named differently. If you didn’t want it to be called `serverfix` on the remote, you could instead run `git push origin serverfix:awesomebranch` to push your local `serverfix` branch to the `awesomebranch` branch on the remote project.

Should you see the errors below when attempting to push, see the Pull / Merge > Resolving Conflicts section of this page for steps on merging your branches, resolving the conflicts, and then completing your push.

```
git push origin v0.2


To github.com:shaunrd0/CMake.git
! [rejected]        v0.2 -> v0.2 (non-fast-forward)
error: failed to push some refs to 'git@github.com:shaunrd0/CMake.git'
hint: Updates were rejected because the tip of your current branch is behind
hint: its remote counterpart. Integrate the remote changes (e.g.
hint: 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

Shared Remote Server Workflow

*You should also be able to share your branches by pushing them to a shared server, working with others on shared branches and rebasing your branches before they are shared. This being said, it’s possible to have a workflow where each developer has write access to their own public repository and read access to everyone else’s -

1. The project maintainer pushes to their public repository.
2. A contributor clones that repository and makes changes.
3. The contributor pushes to their own public copy.
4. The contributor sends the maintainer an email asking them to pull changes.
5. The maintainer adds the contributor’s repository as a remote and merges locally.
6. The maintainer pushes merged changes to the main repository.

Branching

Basic git branch commands -

Checkout and create new branch if it doesnt exist

```
git checkout -b branchname
```

`git log --all --graph --decorate --oneline --simplify-by-decoration` will output your history in a format similar to the Network Graph on GitHub

```
git log --all --graph --decorate --oneline --simplify-by-decoration


* 8221652 (HEAD -> master, origin/master, origin/HEAD) merge v0.4 into master
* 27e6e1c (origin/v0.4, v0.4) Fix for tab spacing in vim
| * feb1da1 (refs/stash) WIP on master: 807e0b3 Reorganized C problem 4
|/
* 807e0b3 (origin/v0.3) Reorganized C problem 4
* 2fc4266 (origin/v0.2) Finishing up v0.2
| * 9187276 (origin/v0.1) Cleaned up the README.
|/
* f1b858b Initial commit of first CMake project
```

The output can be formatted further, and linked with aliases within git -

```
git config --global alias.lg "log --all --graph --pretty=format:'%Cred%h%Creset -%C(yellow)%d%Creset %s %Cgreen(%cr) %C(bold blue)<%an>%Creset' --abbrev-commit"
```

There are also other simpler options for similar output with less information -

```
git show-branch
git show-branch --all
```

Alternatively, if you would rather a GUI - run `gitk` - provided it's configured correctly.

To delete a branch, local or remote, see the commands below -

Remove a local branch

```
git branch -d the_local_branch
```

Remove a remote branch. Be careful with this command! Be sure you know that you want to delete the branch forever.

```
git push origin :the_remote_branch
git push origin --delete the_remote_branch
```

Pull / Merge

Before merging, commit and push a 'checkpoint' to your version or feature branch. If you do not, git will squash the history from your branch into master - this commit can serve as a reference for changes merged into master later on. Should you forget to do this, the merge could still be traced with more effort.

merge `master` into the `test` first to resolve any conflicts on the `test` branch itself. After the `test` branch is clean, up-to-date, and pushed to origin, I'll `git checkout master` and `git merge test`.

`git merge origin/master`. If you want to fast-forward, run `git merge --ff-only origin/master`

The `--squash` option takes all the work on the merged branch and squashes it into one changeset producing the repository state as if a real merge happened, without actually making a merge commit.

Also the `--no-commit` option can be useful to delay the merge commit in case of the default merge process.

Resolving Conflicts

Problems pushing your local changes to a remote (origin) ?
`git pull <remote> <branch>` and resolve the conflicts by following the instructions below .

When attempting to pull or merge branches, there can sometimes be new changes to the same content within the same files on the two different branches. Since git wants to be sure that you retain the changes you want, it pauses our merge and prompts us to resolve these conflicts before creating a final commit to finish our merge.

```
git pull origin v0.2


From github.com:shaunrd0/CMake
* branch            v0.2       -> FETCH_HEAD
Auto-merging 4-Ch2-course-laucher/CMakeLists.txt
CONFLICT (content): Merge conflict in 4-Ch2-course-laucher/CMakeLists.txt
CONFLICT (add/add): Merge conflict in 4-Ch2-course-laucher/4-problems/CMakeLists.txt
Auto-merging 4-Ch2-course-laucher/4-problems/CMakeLists.txt
Automatic merge failed; fix conflicts and then commit the result.
```

Below, we can see that my branch has conflicts with `4-Ch2-course-launcher/CMakeLists.txt` - So, to resolve these, we would run `vim 4-Ch2-course-launcher/CMakeLists.txt`

```
On branch v0.2
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)

  added:     4-Ch2-course-laucher/4-problems/CMakeLists.txt
	both modified:   4-Ch2-course-laucher/CMakeLists.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

`vim <path/to/conflict/file>` and you will notice syntax similar to the below has been added to your file -

```
Some text in a file that has no conflict.

<<<<<<< HEAD
Some text that was changed on the local HEAD.
=======
Some text that was also changed on the remote we are attempting to merge with
>>>>>>> feature-branch

Some more text in a file that has no conflict.
```

All that Git is asking us to do here is delete the changes that we don't wish to keep, and then `git commit -m "Commit message"` to complete our merge. If you want to abort the merge, run `git status` to see how, or just run `git merge --abort`.

So, in this case if we wish to keep the changes that are on our local `HEAD`, and overwrite the changes on our `feature-branch`. Just modify the file, deleting all of the added syntax from the merge conflicts described by git, and any changes that may go with them -

```
Some text in a file that has no conflict.

Some text that was changed on the local HEAD.

Some more text in a file that has no conflict.
```

Our merge conflict is resolved. check `git status`, stage your changes with `git add` and make the commit to finish the merge.

Revision #6
Created 28 June 2020 14:23:16 by Shaun Reed
Updated 18 December 2021 22:44:22 by Shaun Reed