Git Workflow

Introduction

This article explains the workflow that should be followed to contribute to the Tiki GitLab repository at https://gitlab.com/tikiwiki/tiki. Following are the main steps:

1. **Fork** the main Tiki GitLab repository to your GitLab account
2. **Clone** the forked repository to your computer
3. **Checkout** the correct target branch on your computer
4. **Create a new branch** on your computer based on the target branch
5. **Commit** changes to your local clone
6. **Push** changes to your forked repository on GitLab
7. Create a new **Merge Request** from your forked repository on GitLab

Each of these steps is explained in more detail below.

1. **Fork the main Tiki GitLab repository to your GitLab account**
   1. Create an account at https://gitlab.com
   2. Go to Tiki project page, https://gitlab.com/tikiwiki/tiki
   3. Click on fork badge at top right corner
   4. Select the target group you want to place your fork. It will take 15 minutes or more.

2. **Clone the forked repository to your computer**

   In this step, the repository you forked to your account on GitLab is cloned onto your local computer.

   Examples of how to clone

   The examples below use Git command line, but it can be performed in any Git client embedded in an IDE. Also fabiomontefuscolo is the GitLab user in this example and it should be replaced by GitLab user who forked the repository. Only one of the following examples (or another variation not shown here) should be applied.

   ```bash
   git clone https://gitlab.com/fabiomontefuscolo/tikiwiki.git
   ```
Clone to a newly created directory named tiki

```bash
$ git clone git@gitlab.com:fabiomontefuscolo/tiki.git tiki
```

Clone to current folder

```bash
$ git clone git@gitlab.com:fabiomontefuscolo/tiki.git .
```

As Git will checkout all revisions since the beginning of Tiki (over 70 000 commits), will take quite a while, so you may want to limit this as in the following example:

Clone to current folder with 5 last revisions only

```bash
$ git clone git@gitlab.com:fabiomontefuscolo/tiki.git . --depth=5
```

3. Checkout the correct target branch on your computer

For each major version of Tiki, there is a working branch. If the contribution is a fix for version 20.1 for example, the contributor must checkout the branch 20.x.

Note that unlike with SVN, checking out a branch with git does not mean downloading all of the tiki files for the branch into a separate folder. The folder where you cloned your fork will still be used, and the files in the folder will changed to reflect the branch that you've checked out.

Checkout branch 20.x on your computer

```bash
$ git checkout 20.x
```

4. Create a new branch on your computer

This is a very important step. It is a good idea to create another branch for the contribution. According to earlier steps, the repository now is set to branch 20.x and a new branch will start from this point. Suppose branch with the name fixing-left-sidebar, the following command will create this branch.

Create a local branch

```bash
$ git checkout -b fixing-left-sidebar
```

If for some reason the contributor needs to stop his work and start a new one for the same branch 20.x, he can simply commit or stash his changes, switch back to 20.x and create the new contribution he needs. For example, a new urgent task appeared to fix a security issue:

Create a second local branch

```bash
$ git checkout 20.x
$ git checkout -b fixing-xss-issue-on-register-form
```

In this way, the contributor can have two ore more unrelated jobs (fixing-left-sidebar and fixing-xss-issue-on-register-form) for the same target branch (20.x). It will also help on Merge Request creation. Please see the atomic commit convention
Similar to checking out a branch, a new local branch that you've created will still reside in the same folder that you cloned your forked repository into. As explained above, creating local branches is a simple and quick way to isolate different sets of changes so that you are always able to quickly switch back to a clean copy or between sets of changes.

5. Commit changes to your local clone

The community standards for commit messages should be respected please see: Commit Message.

```
Commit changes to your local clone

$ git add tiki-file.php
$ git commit -m "[FIX] Removing bad characters from registration form to avoid XSS attacks"
```

6. Push changes to your forked repository on GitLab

In order to have changes available to create a new Merge Request, it is necessary to push the created branch to your forked repository on GitLab.

```
Push changes to your GitLab fork

$ git push -u origin fixing-xss-issue-on-register-form
```

7. Create a new merge request from your forked repository on GitLab

The purpose of this step is to merge the changes pushed to your GitLab Tiki fork to a specific branch in the main Tiki GitLab repository.

1. Go to your forked repository page in GitLab https://gitlab.com/fabiomontefuscolo/tiki
2. Then put mouse over Repository and click on Branches entry
3. Locate the branch desired (eg.: fixing-xss-issue-on-register-form) and then click on Merge Request button
4. On top right corner, click on Change branches and select the correct target branch (eg.: 20.x)
5. Write a detailed description. Include steps to reproduce a bug if needed or how to test a new feature

Check your merge request via the web interface

Here is an example: https://gitlab.com/tikiwiki/tiki/merge_requests/18/diffs
Find yours here: https://gitlab.com/tikiwiki/tiki/merge_requests/

Make sure you are respecting the atomic commit convention.

If there is an issue, fix quickly or indicate in the comments that you will do later, or that you need help.

Wait and interact

Maintainers may ask you changes on that pull request. In this case, just checkout the branch related to the merge request, add new commits and push it again.

Once merge request is accepted

Via the magic of SubGit, you’ll soon see your commits here: https://sourceforge.net/p/tikiwiki/code/HEAD/log/

To backport a commit

first check that we have the branch on which we want to backport the commit if not track it
to check
git branch -a

to track and clone the branch's commits
git remote add --track 25.x 25.x-track https://gitlab.com/keutyche/tiki.git
git fetch 25.x-track --depth=500
move to a new branch with the elements we just downloaded
git checkout -b fixCorruption 25.x-track/25.x

do the cherry pick

git cherry-pick a12f4017cf6b13eefab06f8110ff5aa2516439d3
a12f4017cf6b13eefab06f8110ff5aa2516439d3 This is the hash of the commit after being merged
then make a push and create a new merge request to the desired branch
git push -u origin fixCorruption

Important
When commands like git reset --hard or git push --force is needed, it may be that there is a problem with the workflow and it should be planned again.

Manually Syncing Your Fork to the main Tiki Repository
An alternative to mirroring your fork as described in Mirroring your fork to the main Tiki repository, is to manually sync it as follows:

1. Add the main Tiki repository as a remote with the command git remote add upstream https://gitlab.com/tikiwiki/tiki.git
   Note, you only need to do this step once
2. Apply commits from the main Tiki repository to your local repository with the command git pull upstream master --rebase
3. Apply those commits from your local repository to your forked repository with the command git push origin master
   1. If you get an error message after attempting this push to your forked repository and there is no other activity on that repository (e.g., it is only used by you for merge requests), try git push origin master --force

You should sync often, especially before making changes locally, and after committing but before pushing to your fork.

Fix Out-of-Sync Fork
Sometimes your forked repository can get out of sync with the main Tiki repository and your local copy. Symptoms are when you frequently have to force push to the fork, or unwanted other commits from other authors are combined into a merge request made from your fork to the main repository. Follow these steps to fix:

```bash
git fetch upstream
git checkout master
git reset --hard upstream/master
git push origin master --force
```

See also
- Backport changes using Git
- Tiki Unit Testing