UsingGit

Apache Tika uses the Git version control system. Apache provides writeable Git repositories hosted at Github and mirrored to https://gitbox.apache.org.

Overview

This page contains few migrations guides.

1. If your source code was checkout from SVN repo, scroll down for a guide to migrate to git repo. 2. If your source code was cloned from a git repo at git-wip-us.apache.org for primary development. Please scroll down to find a guide to migrate to github repo. 3. Currently, the repository at https://github.com/apache/tika can be used as a primary repo. If you are looking for a fresh getting started instructions, skip migration guides, start with Developer Workflow and/or Contribution workflow using Github's pull requests.

Migrating from an existing SVN checkout of Tika to Git

If you need to migrate from an SVN checkout of Tika to Git, follow these instructions below.

1. `svn status` (ensure no local changes) 2. `mv .svn .svn.old` (or simply `find . -name "*.svn" -exec rm -rf {} \;`) 3. `git init` 4. `git remote add origin git@github.com:apache/tika.git` 5. `git checkout -b merge-branch` 6. `git fetch --all` 7. `git reset --hard origin/master` 8. `git checkout master`

And on my Tika 2.x checkout the last two steps were changed to:

1. `git reset --hard origin/2.x` 2. `git checkout 2.x`

Migrating to Github from git-wip-us.apache.org

Previously the repo was hosted on git-wip-us.apache.org, if you are one of those who cloned it from git-wip-us.a.o and wants to migrate to Github as a primary repo, follow these instructions:

1. please update your repository's remote url:

```
git remote set-url origin git@github.com/apache/tika.git
```

2. If not already enabled - Goto your github account settings and please do 2-Factor-Auth for your account. It's a MUST do, no kidding. 3. Connect your apache account to github account. Click on https://gitbox.apache.org/setup/ If you type the URL manually, please note there is a '/' at the end of setup/
4. Authorize your Apache account, and then Authorize your github account.

   a. If you see all three boxes green and a tick inside them, your setup is complete 🙂 2. Infra team had instructed in another mailing list that sometimes 'MFA status' might not tick in first pass. They said that the MFA status updater runs hourly so wait for an hour atleast. This usually happens when you authorize github account without 2FA, it won't go through in first pass. 3. If you face issues report to infra team!

Checking out a copy of Tika and modifying it

To check out a copy of Tika, perform the following command:

```
git clone git@github.com:apache/tika.git
```

This will check out a copy of the code.

Once you have the code, you can modify a file, or two, then add the files for staging/commit, and then commit them. Once done, you can also decide to push the files up to the master repository if you have write access and are a member of the PMC and/or a committer. If you don't have write access to the repository, you can follow this guide for issuing pull requests that committers/PMC members can merge.

Here are the steps to achieve the above if you have commit access, assuming you have 2 files, `file1` and `file2` in the repository local copy that you modified.

1. `git add file1 file2` 2. `git commit -m "message describing change."` 3. `git push -u origin <branch_name>`

Suggested Developer Workflow

As a longtime SVN user, it took me a while to figure out Git and realize its great support for use cases. Here is the one I most commonly use in Git that I think fits the Tika workflow as a PMC member/committer quite well. Imagine that you are working on an issue, call it TIKA-xxx. Either you have filed it yourself, or someone has filed it. My suggestion is to create a feature/working branch for that issue, named exactly after the JIRA issue, e.g., create a branch TIKA-xxx. Commit your changes and modifications there. Then, after the changes are applied, tested, and verified, then you can switch back to the master branch, and push your local copy to the remote. If you'd like a review before pushing (e.g., attaching a patch you have two options). You can create a patch file by diffing the branches, and then attach that to the JIRA issue. This is common and already in use using Subversion in Tika in years past. Let's take this use case to start. Inside of your tika clone, do the following:

1. `git branch TIKA-xxx` 2. modify files, change them, test, etc. 3. `git add <changed files>` (to see them, try `git status` or simply to stage **all** changes, `git add *`) 4. `git commit -m "Fixes for TIKA-xxx contributed by <your first name> <your last name> <your email>" 5. `git checkout master` 6. `git diff trunk..TIKA-xxx > TIKA-xxx.<your last name>.<yyMMdd>.patch.txt` 7. Attach the patch created in 6 to JIRA.

If you are **not** looking for a review or you have already had a review and are ready to commit the changes, push them:

0. (if you haven't already merge, the branch into master) `git checkout master && git merge TIKA-xxx`

1. `git push -u origin master`

Suggested User Contribution Workflow

Users contribute patches to Tika using JIRA and/or Github. Let's take the use case for contributing via JIRA, then we'll tackle the Github one.

JIRA contribution

Grab the user's patch file, and then apply it locally. Before doing so, create the feature branch by following step 1 from the Developer workflow section. Then, assuming the patch file is named according to the conventions from step 6 in the Developer workflow section, perform the following steps (make sure you are on the TIKA-xxx feature branch):

1. `git apply < TIKA-xxx.<your last name>.<yyMMdd>.patch.txt` 2. Steps 3-4 from Developer workflow guide. 3. Final 2 steps from Developer workflow guide (aka the merge of feature branch step and then the push to origin trunk)

Github contribution

If they are contributing using Github they are submitting a Pull Request for review you can easily merge their pull request into your local feature branch using Git. Assuming the Github user is "user01" and the branch they have created is "fix-tika-stuff" (you can find this information in the Tika Pull request, for example in this pull request #65, the username is `smahda` and branch name is `TIKA-1803`), you can merge it into your local feature branch like so (again, make sure you are on the local feature branch for your issue, TIKA-xxx first by typing `git checkout TIKA-xxx` if you aren't already):

1. `git pull https://github.com/user01/tika.git fix-tika-stuff` (will find the remote github branch and merge locally) 2. test and make sure PR works, etc. Fix conflicts, etc. 3. Final 2 steps from Developer workflow guide (aka the merge of feature branch step and then the push to origin trunk)