1. Information References
2. Conventions
3. Environment Setup
4. Help
5. Tasks
6. Workflow models
    1. The Integration-Manager Workflow Model and its specific adoption in the Github Workflow for Contributer's Model
        1. Integration-Manager workflow (Chacon and Straub 2014, ch. 5 Distributed Git, sec. Integration-Manager Workflow)
        2. Github workflow for contributors. Based on Integration-Manager Workflow. Github calls it the "Fork & Pull Model".
            1. See
                1. (Chacon and Straub 2014)
                    1. Ch. 5. Distributed Git, Sec. Integration-Manager Workflow
                    2. Ch. 6. GitHub, Sec. Contributing to a Project.
            2. Setup
                1. Under "Create and delete repositories (repos)" above follow
                    1. "Create a remote github respository (repo) when no project files exist anywhere else."
                    2. "Create a local repo from an existing remote repo: github "forking" a project."
                2. @Local you should end up with a local repo and a mapping to two remote repos: origin, and upstream.
                    1. local.
                        1. C:\Users\You\Git\example\.git
                    2. remotes.
                        1. origin. https://github.com/you/example.git
                        2. upstream. https://github.com/owner/example.git
                    3. Verify in git clients.
                        1. $ git remote --verbose show
                        2. @SmartGit. Branches Window. Observe:
                            1. local. (observe the "local branches" misnomer. Should just be "local").
                            2. origin. https://github.com/you/example.git
                            3. upstream. https://github.com/owner/example.git
            3. General operations
                1. @Local. Create a local topic branch from local master.
                    1. @SmarGit
                        1. We are on the master branch (we need to checkout *from* the master).
                        2. Menu > Branch > [Add Branch...]
                            1. Branch: (Make your topic branch descriptive. E.g. "issue315-fruit-handling")
                            2. [Add Branch]
                    2. @$
                        1. $ git branch topic-branch
                        2. $ git show-ref
                2. @Local. Checkout local topic branch.
                    1. @SmartGit
                        1. Window Branches > Right Click on "topic-branch" > Checkout; or double click on "topic-branch"
                        2. Observe triangle against tropic branch is now black.

2. @$
    1. $ git status
    2. $ git checkout topic-branch
    3. $ git status
3. @Local. Ensure our local topic-branch is up-to-date with upstream/master.
    1. Summary: Pull upstream/master branch into local topic. Resolve any merge conflicts. Push updates to origin/topic-branch.
    2. See
        1. (Chacon and Straub 2014, Ch. 6 Github, Sec. Advanced Pull Requests, Subsec. Keeping up with upstream)
            1. If you want to merge in the target branch to make your Pull Request mergeable, you would add the original repository as a new remote, fetch from it, merge the main branch of that repository into your topic branch ...
            2. ... fix any issues and finally push it back up to the same branch you opened the Pull Request on.
        2. https://help.github.com/articles/syncing-a-fork/
        3. https://www.syntevo.com/doc/display/SG/GitHub%3A+keeping+in+sync+wit
        4. http://stackoverflow.com/a/16810513/872154
    3. Pull upstream/master branch into local topic.
        1. Verify we are on local/topic-branch.
            1. @SmartGit
                1. In the Branches window observe the triangle.
            2. @$
                1. $ git status
                2. Observe "On branch topic-branch"
        2. Fetch from upstream repo
            1. @SmartGit
                1. Branches Window > Click on upstream to highlight.
                2. [Pull |
                3. Configure Pull [This will only appear initially]
                    1. Choose: Rebase
                    2. Remember as default for other repositories: unticked.
                    3. [Configure |
                4. Fetch from all remotes: Untick
                5. Fetch From: upstream (https://github.com/owner/example.git)
                6. More Options
                    1. Overwrite choice (Should be disabled)
                        1. Merge fetched remote changes: Chosen.
                        2. Rebase local branch onto fetched chagnes
                    2. Update existing and fetch new tags: unticked
                    3. Remember as default for repository: unticked.
                7. [Fetch Only]
            2. @$ git fetch upstream
        3. Verify all relevant repos and branches are displayed
            1. @SmartGit
                1. Main Environment > Journal window > Right hand Hamburger Menu > Show Auxillary branch ... > Select "upstream/master".
                2. [Log] ... > Branches Windows > Select All.
        4. Merge upstream/master branch into local topic branch

1. @SmartGit
    1. [Log]
    2. Branches Window > Tick all repos and branches (So you have complete picture).
    3. Graph Window > Click on upstream/master (whichs points to the last "Merge pull request" commit) to select.
    4. [Merge]
    5. [Fast-Forward]. General rule: if you can fast-foward merge, do it.
2. @$
    1. $ git merge upstream/master
    2. Observe "Fast-forward" merge done; or "Already up-to-date." if there where no changes to merge.

4. Push updates to origin/topic-branch.
    1. Verify we are on local/topic-branch.
        1. @SmartGit
            1. In the Branches window observe the triangle.
        2. @$
            1. $ git status
            2. Observe "On branch topic-branch"
    2. @SmartGit
        1. [Push] > [Push to]
            1. Target Repository: origin
            2. Push to: Tracked or matching branch
            3. (This will create origin/topic-branch when done for the first time)
            4. [Push]
            5. "Enter the master password". This is the (in AppsWindows.txt as) "Smartgit Master password".
    3. @$
        1. $ git push
        2. Observe something like
            1. To https://github.com/JohnLukeBentley/example.git [Which is the origin repo]
            2. adc3ab3..a4ebcfa topic-branch-03 -> topic-branch-03

4. @Local. Make one or more commits to improve the project.
    1. In your code editor make a coherent set of changes: edit files, create files, delete files, etc. This often will entail a single small change to a single file.
    2. Verify we are on local/topic-branch.
        1. @SmartGit
            1. In the Branches window observe the triangle.
        2. @$
            1. $ git status
            2. Observe "On branch topic-branch"
    3. @SmartGit
        1. $ git diff --check # Warn if changes introduce conflict markers or whitespace errors. https://git-scm.com/book/en/v2/Distributed-Git-Contributing-to-a-Project, under "Commit Guidelines"
        2. Files Window > Click on modified files, one at a time, to review changes.
        3. (Optional) [Stage |

4. [Commit]... (this will offer for commit even files not yet staged and even not yet tracked)
    1. Commit Message
        1. "As a general rule, your messages should start with a single line that’s no more than about 50 characters and that describes the changeset concisely, followed by a blank line, followed by a more detailed explanation."
            1. (https://git-scm.com/book/en/v2/Distributed-Git-Contributing- under "Commit Guidelines")
        2. See also GitReference.md, "Pull Request and commit message formats".
    2. [Commit](Don't | Commit & Push | as it's a good idea to review your commit before pushing to remote repos).
5. [Log]...
    1. Branch Window > Verify all repos and branches are selected (So you have complete picture)
    2. Commits Window > Click on topic-branch (which will coincide with the latest commit); and/or click, in turn, on each of the commits you've made.
        1. Review that the commit is (or the commits are) good: the correct commit message; and the set of changes is as desired.
4. @$
    1. $ git diff --check #Warn if changes introduce conflict markers or whitespace errors. https://git-scm.com/book/en/v2/Distributed-Git-Contributing-to-a-Project, under "Commit Guidelines"
    2. $ git diff # view staged changes.
    3. $ git commit --all --message "All my changes."
    4. Or
    5. $ git commit --all
        1. Enter VIM environment > Keyboard [Insert |
        2. Make changes > [Esc] x: | Enter |
    6. $ git log -1 --patch
        1. Review that the commit is (or the commits are) good: the correct commit message; and the set of changes is as desired.
    5. Optionally repeat for another commit.
5. Repeat "@Local. Ensure our local topic-branch is up-to-date with upstream/master." above, without any push ... we'll do that next.
6. @Local. Push your local topic-branch to origin/topic-branch
    1. Verify we are on local/topic-branch.
        1. @SmartGit
            1. In the Branches window observe the triangle.
        2. @$
            1. $ git status
            2. Observe "On branch topic-branch"
    2. See "Observe the state of references in the log." bellow.
    3. Push
        1. @SmartGit
            1. [Push |
                1. Current branch 'topic-branch'
                2. (This will create origin/topic-branch when done for the first time)
            2. Output Window. Verify this was pushed to origin remote

2. @$
    1. $ git push
    2. Observe something like
        1. To https://github.com/JohnLukeBentley/example.git [Which is the origin repo]
        2. a4ebcfa..eed4bf6 topic-branch-03 -> topic-branch-03
4. Repeat "Observe the state of references in the log." bellow.
    1. Observe HEAD -> topic-branch, and origin/topic-branch both point to the last commit.
7. @Github. Open a Pull Request on Github. This will entail a request to pull from you/example(origin)/topic-branch into them/example(upstream)/master branch.
    1. Verify: Code [Tab] > Branch: topic-branch
    2. Verify a message like "This branch is x commit(s) ahead of them:master
    3. [Compare & pull request] (or | Pull request | ....
    4. Open a pull request
        1. Verify
            1. [Base fork: them/example| |base:master| ... |head fork: you/example| |compare:topic-branch|. Able to merge.
        2. Complete heading and body details on pull request.
            1. See also GitReference.md, "Pull Request and commit message formats".
            2. Optionally use task list markdown format in body. Repeat first line of current and anticipated commits. E.g.
                1. * [x] task1
                2. * [ ] task2
        3. [Create Pull Request |
        4. Observe you are redirected to the pull request in the owner's account. E.g. https://github.com/owner/example/pull/2
            1. Observe: "This branch has no conflicts with the base branch"
8. @GithubOwner. Evaluates pull request. (If you are simulating this, then open another browser (side-by-side) and log in as the owner)
    1. Pull requests [tab]: e.g. https://github.com/owner/example/pulls
        1. Click through on the pull request.
            1. [Merge pull request] ; or
                1. Generelly accept the default "Create a merge commit" as this best preserves the history.
                2. [Confirm merge |
            2. [Close pull request] ; or
            3. [Comment] request further changes from the contributor.
9. Contributer responses to Owner's Pull request decision
    1. If the owner Merges the Pull request.
        1. Do "@Local. Ensure our local topic-branch is up-to-date with upstream/master."
        2. Update your local and origin masters.
            1. Checkout master
            2. Click on commit pointed to by upstream/master and local/topic-branch (the last commit) > Merge > Fast-forward.
            3. Push to origin/master
        3. Delete the topic-branch (its commit history is kept safe by being merged into local/master (and upstream/master).
            1. @SmartGit

1. Main Environment > Branches > Double click local branches, master to checkout.
2. Right click topic-branch > Delete ...
    1. Delete tracked branch: ticked
    2. Delete from remote 'origin': ticked
    3. [Delete |
2. @$
    1. Delete tracked remote reference topic-branch
        1. $ git branch --delete --remotes origin/topic-branch
    2. Delete local topic-branch
        1. $ git branch --delete topic-branch
    3. Delete remote topic-branch
        1. $ git push --porcelain --progress origin :refs/heads/topic-branch
        2. or
        3. ? $ git push --prune

2. If the owner requests further changes with respect to the current pull request.
    1. Repeat
        1. "@Local. Make one or more commits to improve the project." (in the same local/topic-branch)
        2. "@Local. Push your local topic-branch to origin/topic-branch"
    2. This results in further commits being appended to the original Pull Request.
    3. Repeat: "@GithubOwner Evaluates pull request."
3. If the owner endorses some of the commits but rejects others (and is going to keep the pull request open)
    1. If all of the commits rejected are from the last, contiguously going backwards, then revert those commits.
        1. @SmartGit
            1. Select all the commits you want to revert (a contiguous selection from the last commit to the target undesired commit) > [Revert |
        2. @$
            1. $ git status # verify you are on the topic-branch
            2. $ git x-log -6 # Observe which commits you want to revert
            3. $ git revert 904580e cf0301c
    2. Otherwise make your changes to the file manually, in your editor, and push another commit (or series of commits).
        1. See "@Local. Make one or more commits to improve the project."
4. If the owner closes the pull request ("with unmerged commits"), the implication is that you need to undo your changes somehow. You have various strategies available.
    1. [Universal Methods]
        1. If you don't want to use your topic-branch again:
            1. Optionally Tag. Delete topic-branch locally. If you change your mind and now want to continue to use the topic-branch: recreate the topic-branch at the commit rolled back to.
                1. Check: master points to the commit before those you want to discard.
                    1. @Smartgit:
                        1. Log Environment > Branches Windows: Ensure all branches ticked, including any "Tags" > Observe master location in Commits log.

2. @$
    1. $ git x-log -10 # An aliased log command.
2. [Optional] Tag the last commit with "pr###-closed-topic-branch", where ### is some number. This is handly has a historical marker.
    1. Verify you are on topic-branch
        1. @SmartGit
            1. Main or Log Environment > Observe triangle in UI points to topic branch
        2. @$
            1. $ git status
            2. Observe > "On branch topic-branch"
    2. Do tagging
        1. @SmartGit
            1. Log Environment > Right click on last commit > Add Tag (don't annotate it) > Tag: "pr###-closed-topic-branch", where ### is some number. > [Add Tag & Push |
            2. Log Environment> F5 to refresh > Commits window: observe new tag.
        2. @$
            1. $ git tag "pr19-closed-topic-branch"
            2. $ git x-log -10
            3. Observe new tag.
3. Checkout master branch
    1. @SmartGit:
        1. Log Environment > Right click on master commit > Checkout ... > Checkout local branch: master > [Checkout |
    2. @$
        1. $ git checkout master
4. Delete topic-branch whose changes have not been merged into master.
    1. View topic-branch on remote, topic-branch on local, remote tracking topic-branch on local
        1. @$
            1. View topic-branch on remote
                1. ps> git ls-remote origin [Select-String refs/heads
            2. View topic-branch on local, remote tracking topic-branch on local
                1. $ git branch –all
    2. Do delete
        1. @SmartGit
            1. Main Environment > Right click on topic-branch > Delete ...
            2. Delete
                1. Delete tracked branch: ticked
                2. Delete from remote 'origin': ticked
                3. [Delete |

2. @$
    1. [The following order does things cleanly, although doing it out of order will probably eventually return matters to a consistent state]
    2. Delete topic-branch on local
        1. $ git branch --delete topic-branch
    3. Delete remote tracking topic-branch on local
        1. $ git branch --delete --remotes origin/topic-branch
    4. Delete topic-branch on remote
        1. $ git push origin :refs/heads/topic-branch
3. Repeat "View topic-branch on remote, topic-branch on local, remote tracking topic-branch on local" to verify

5. Result:
    1. If you tagged it.
        1. The rejected commit history preserved in the local repo log, because you tagged it.
        2. The tag is pushed to origin so the history is backed up.
        3. The upstream repo doesn't get corrupted with your tags as the pull requests are only with respect to particular (active) branches.
    2. If you didn't tag it
        1. The rejected commit history is lost in your local repo log, but
            1. It is preserved in the reflog (until garbage collection?)
            2. It is preserved on Github under Pull requests (but now your log doesn't record where the closed pull request occured).

6. Optionally create a new topic-branch at the commit now pointed to by master. Either use a new name or by recycle the name of the just deleted topic-branch. The now taged and closed branch will be by-passed on further commits.
    1. Choose to either:
        1. Recreate the topic branch with the prior name e.g. "topic-branch"
        2. Create an entirely new name. E.g. "Issue214-loop-bug"
    2. @SmartGit
        1. Log Environment > Select last desired commit (should be pointed to by master, origin/master, and upstream/master) > [Check Out] ...
            1. Create local branch: topic-branch-new
            2. [Checkout |
        2. Main Environment > [Push] ... > Current branch 'topic-branch-new' > | Push |
        3. Log Environment. Observe all the references point to the last commit. master, master/origin, topic-branch, origin/topic-branch, upstream/master
    3. @$

1. Verify we are on master
    1. $ git status
    2. $ git x-log -10 #verify we are on master
2. Create topic-branch on local
    1. $ git checkout -b "topic-branch-new"
3. Create topic-branch on remote
    1. $ git push origin refs/heads/topic-branch-new:refs/heads/topic-b
4. Set remote tracking reference for topic-branch.
    1. $ git push --set-upstream origin topic-branch-new
5. View topic-branch on remote, topic-branch on local, remote tracking topic-branch on local
    1. @$
        1. View topic-branch on remote
            1. ps> git ls-remote origin [Select-String refs/heads
        2. View topic-branch on local, remote tracking topic-branch on local
            1. $ git branch --all

2. If you want to use your topic-branch again:
    1. Optionally Tag. Hard reset. If you change your mind and now want to rid yourself of the topic-branch: delete it.
        1. Verify: master points to the commit before those you want to discard.
            1. @Smartgit:
                1. Log Environment > Branches Windows: Ensure all branches ticked, including any "Tags" > Observe master location in Commits log.
            2. @$
                1. $ git x-log -10 # An aliased log command.
        2. Verify: you are on topic-branch
            1. @SmartGit
                1. Main or Log Environment > Observe triangle in UI points to topic branch
            2. @$
                1. $ git status
                2. Observer > "On branch topic-branch"
        3. [Optional] Tag the last commit with "pr###-closed-topic-branch", where ### is some number. This is handly has a historical marker.
            1. Verify you are on topic-branch
                1. @SmartGit
                    1. Main or Log Environment > Observe triangle in UI points to topic branch
                2. @$
                    1. $ git status
                    2. Observer > "On branch topic-branch"
            2. Do tagging
                1. @SmartGit

1. Log Environment > Right click on last commit > Add Tag (don't annotate it) > Tag: "pr###-closed-topic-branch", where ### is some number. > [Add Tag & Push |
2. Log Environment> F5 to refresh > Commits window: observe new tag.

2. @$
    1. $ git tag "pr19-closed-topic-branch"
    2. $ git x-log -10
    3. Observe new tag.

4. While still checkout on topic-branch, hard reset to local/master (you select the commit you want to commit: all the earlier commits are discarded).
    1. @SmartGit
        1. Verify you have topic-branch checkedout
            1. Observe the triangle in the UI.
        2. Log Environment > Commits > Observe master and topic-branch are the the right relative places.
        3. Log Environment > Commits > Right click local/master > Reset ... 'hard' > [Reset]
        4. Edit > Prefrences > Commands > Push [section] > Allow modifying pushed commits (e.g. forced-push): ticked.
        5. Main Environment > [Push] > Current branch 'topic-branch' > | Push |
            1. 'Do you want to replace the remote Branch?' > [Replace |
        6. Edit > Prefrences > Commands > Push [section] > Allow modifying pushed commits (e.g. forced-push): unticked. (For future safety).
    2. @$
        1. Verify you have topic-branch checked out
            1. $ git status
        2. $ git x-log -10 # Observe master and topic-branch are the the right relative places.
        3. $ git reset --hard master
        4. $ git push --force
        5. Observe that the following refernces point to an ancestor of pr21-closed-topic-branch: head, topic-branch, master, origin/topic-branch, origin/master.
            1. $ git x-log pr21-closed-topic-branch -10

5. Optionally delete the topic branch. Either.
    1. Keep the topic-branch if you want to make further commits against a new pull request under that name.
    2. Delete the topic-branch.
        1. @SmartGit
            1. Main Environment > Branches > Double click local branches, master to checkout.
            2. Right click topic-branch > Delete ...
                1. Delete tracked branch: ticked

2. Delete from remote 'origin': ticked
3. [Delete |

2. @$
    1. Delete tracked remote reference topic-branch
        1. $ git branch --delete --remotes origin/topic-branch
    2. Delete local topic-branch
        1. $ git branch --delete topic-branch
    3. Delete remote topic-branch
        1. $ git push --porcelain --progress origin :refs/heads/topic-branch
        2. or
        3. ? $ git push --prune

6. Cons:
    1. More convoluted than "Tag then delete topic-branch"
    2. Prone to reset errors.

2. If you want to preserve commit history in local log.
    1. [Don't] Keep but abandon topic branch (if you are never going to work in the topic branch again).
        1. High level steps:
            1. Merge local master to the commit made before the pull request was initiated.
            2. Checkout a new topic branch, e.g. 'topic-branch-03' from the commit where the master is.
            3. Continue work on new topic branch.
            4. Don't delete abandoned topic branch (unless you are happy to lose that history in your repo).
        2. Reason not to use:
            1. Topic branches can proliferate. Generally you want keep your repo's lean, only having branches you are actively using and switching bewteen.
    2. [Don't] Revert then push reversions (to preserve commit history). Keep or delete the topic branch.
        1. Check: master points to the commit before those you want to revert.
            1. @Smartgit:
                1. Log Environment > Branches Windows: Ensure all branches ticked, including any "Tags" > Observe master location in Commits log.
            2. @$
                1. $ git x-log -10 # An aliased log command.
        2. Verify: you are on the topic-branch
            1. @SmartGit:
                1. Log Environment > Commits window > Observe the triangle.
            2. @$
                1. $ git status
                2. Observe "On branch topic-branch"
        3. Tag the last commit with "pr###-closed-topic-branch", where ### is some number. This is handly has a historical marker.

1. @SmartGit
    1. Log Environment > Right click on last commit > Add Tag (don't annotate it) > Tag: "pr###-closed-topic-branch", where ### is some number.
    2. Log Environmetn > F5 to refresh > Commits window: observe new tag.
2. @$
    1. $ git tag "pr19-closed-topic-branch"
    2. $ git x-log -10
    3. Observe new tag.
4. Keep the topic branch but revert the changes.
    1. @SmartGit
        1. Log Environment > Select all the commits you want to revert (contiguously from the last commit) > Right Click > Revert .. > [Revert & Commit |
        2. Log Environment > Commits > Observe new revert commits.
    2. @$
        1. $ git status # verify you are on the topic-branch
        2. $ git x-log -10 # Observe which commits you want to revert
        3. Practice Revision selection
            1. $ git x-log master..head # All commits from head, unreachable by master
            2. $ git x-log head~2..head # The last two commits (assumes a linear history for the range)
            3. $ git show –oneline 44b3 2ec6
        4. Do revert using chosen revision selection
            1. $ git revert master..head
                1. Enter VIM Environment > : x [Enter] > [Repeat for each commit reverted]
        5. $ git x-log -10
    3. This creates a new commits with the changes undone, but preserves the history of changes.
5. Push the reverted commits to origin/topic branch.
    1. @SmartGit
        1. Main Environment > [Push] ... > Current branch 'topic-branch' > | Push |
        2. Log Environment > Commits >
        3. Observe log: topic-branch and origin/topic-branch are now even on the last commit.
    2. @$
        1. $ git push
        2. $ git x-log -10
        3. Observe log: topic-branch and origin/topic-branch are now even on the last commit.
6. Checkout local/master
    1. @SmartGit
        1. Main Environment > Branches > Double click on Local Branches/ master
        2. Log Environmet > Commits.

3. Observe topic-branch ahead of master and HEAD

2. @$
    1. $ git checkout master
    2. $ git x-log topic-branch -10
    3. Observe topic-branch ahead of master and HEAD

7. Merge (fast-forward) reverted topic-branch into master

1. @SmartGit
    1. Log Environment > Click on last commit (a revert pointed to by topic-branch and origin/topic branch) to select.
    2. [Merge] > | Fast Foward |
    3. Observe master now points to last commit (with topic-branch and origin/topic-branch)

2. @$
    1. $ git merge topic-branch
    2. $ git x-log -10
    3. Observe master now points to last commit (with topic-branch and origin/topic-branch)

8. Either:
    1. Keep the topic-branch if you want to make further commits against a new pull request under that name.
    2. Delete the topic-branch.
        1. @SmartGit
            1. Main Environment > Branches > Double click local branches, master to checkout.
            2. Right click topic-branch > Delete ...
                1. Delete tracked branch: ticked
                2. Delete from remote 'origin': ticked
                3. [Delete |
        2. @$
            1. Delete tracked remote reference topic-branch
                1. $ git branch --delete --remotes origin/topic-branch
            2. Delete local topic-branch
                1. $ git branch --delete topic-branch
            3. Delete remote topic-branch
                1. $ git push --porcelain --progress origin :refs/heads/topic-branch
                2. or
                3. ? $ git push --prune
    9. Push to origin
    10. Setup new pull request to sync reversions into upstream/master
        1. @Github > [New Pull Request |
        2. @Github Owner merges new pull request
        3. @Local > Fetch then fast forward merge changes from upstream/master into local/master
        4. @Local > Push merge to origin/master to bring everything up to date.
    11. Reason not to use
        1. It requires an additional pull request to close a previous one.

3. If you want to lose commit history in local log.

1. [Recommended: second choice] Delete topic-branch without tagging (rejected commits are lost to the repo history). Delete locally first.
    1. Check: master points to the commit before those you want to discard.
        1. @Smartgit:
            1. Log Environment > Branches Windows: Ensure all branches ticked, including any "Tags" > Observe master location in Commits log.
        2. @$
            1. $ git x-log -10 #An aliased log command.
    2. Checkout master branch
        1. @SmartGit:
            1. Log Environment > Right click on master commit > Checkout ... > Checkout local branch: master > [Checkout |
        2. @$
            1. $ git checkout master
    3. Delete topic-branch whose changes have not been merged into master.
        1. @SmartGit
            1. Main Environment > Right click on topic-branch > Delete ...
            2. Delete
                1. Delete tracked branch: ticked
                2. Delete from remote 'origin': ticked
                3. [Delete |
        2. @$
            1. Delete remote tracking topic-branch
                1. $ git branch --delete --remotes origin/topic-branch
            2. Delete local topic-branch
                1. $ git branch --delete topic-branch
            3. Delete remote topic-branch
                1. $ git push --porcelain --progress origin :refs/heads/topic-branch
                2. or
                3. ? $ git push --prune
    4. Result:
        1. Observe that the rejected commits (that constituted the pull request) are now gone from your local history. Github, however, keeps the commits in the Pull request. And the commits are in your reflog (at least until garbage collection?).
            1. @SmartGit
                1. Log Environment > Commits
            2. @$ git reflog -10
    5. Optionally create a new topic-branch, either with a new name or by recycling the name of the just deleted topic-branch. This now taged and closed branch will be by-passed.
        1. Choose to either:
            1. Recreate the topic branch with the prior name e.g. "topic-branch"
            2. Create an entirely new name. E.g. "Issue214-loop-bug"

2. @SmartGit
    1. Log Environment > Select last commit (should be pointed to by master, origin/master, and upstream/master) > [Check Out] ...
        1. Create local branch: topic-branch
        2. [Checkout |
    2. Main Environment > [Push] ... > Current branch 'topic-branch' > | Push |
    3. Log Environment. Observe all the references point to the last commit. master, master/origin, topic-branch, origin/topic-branch, upstream/master

2. [Discouraged] Delete topic-branch without tagging (rejected commits are lost to the repo history). Delete remotely first.
    1. @Local
        1. Check: master points to the commit before those you want to discard.
            1. @Smartgit:
                1. Log Environment > Branches Windows: Ensure all branches ticked, including any "Tags" > Observe master location in Commits log.
            2. @$
                1. $ git x-log -10 # An aliased log command.
    2. @GitHub
        1. Logged in as contributer > Pull Request that has been closed >
        2. Observe
            1. Closed with unmerged commits
            2. This pull request is closed, but the YourContributorUsername:topic-branch branch has unmerged commits.
            3. [Delete branch |
        3. Hit [Delete branch |
    3. @Local
        1. Checkout master branch
            1. @SmartGit:
                1. Log Environment > Right click on master commit > Checkout ... > Checkout local branch: master > [Checkout |
            2. @$
                1. $ git checkout master
        2. Delete local topic-branch
            1. @SmartGit
                1. Main Environment > Right click on topic-branch > Delete ...
                2. "Are you sure to delete the local branch 'topic-branch'? > [Delete |
            2. @$
                1. Examine current state of branches
                    1. $ git show-ref
                    2. $ git branch --all
                2. Delete remote tracking branch topic-branch

1. $ git branch --delete --remotes origin/topic-branch

3. Delete local topic-branch
    1. $ git branch --delete --force topic-branch
4. The topic-branch on the remote has already been deleted
5. Verify current state of branches
    1. $ git branch --all

4. Result:
    1. Observe that the rejected commits (that constituted the pull request) are now gone from your local history. Github, however, keeps the commits in the Pull request. And the commits are in your reflog (at least until garbage collection?).
        1. @SmartGit
            1. Log Environment > Commits
        2. @$ git reflog -10
5. Optionally create a new topic-branch, either with a new name or by recycling the name of the just deleted topic-branch. This now taged and closed branch will be by-passed.
    1. Choose to either:
        1. Recreate the topic branch with the prior name e.g. "topic-branch"
        2. Create an entirely new name. E.g. "Issue214-loop-bug"
    2. @SmartGit
        1. Log Environment > Select last commit (should be pointed to by master, origin/master, and upstream/master) > [Check Out] ...
            1. Create local branch: topic-branch
            2. [Checkout |
        2. Main Environment > [Push] ... > Current branch 'topic-branch' > | Push |
        3. Log Environment. Observe all the references point to the last commit. master, master/origin, topic-branch, origin/topic-branch, upstream/master
6. Con
    1. Perhaps a bit more tricky than doing all the the topic-branches locally

10. @Local. Make further changes and create a new pull request.
    1. Make local change in same topic branch, local/topic-branch
    2. Commit.
    3. Push to origin (ContributorPublicFork)
    4. @Github > ContributorPublicFork > Branch > [Select topic branch] topic-branch > New Pull Request
    5. The New Pull Request will include all commits not previously merged into master.

4. Handling merge conflicts.
    1. @SmartGit
        1. [Pull] > Fetch from > upstream > | Fetch Only |
        2. Log > Select upstream/master > [Merge] ... > | Merge to Working Tree |
            1. Exectuing a command has failed. merge failed (return code 1) > [OK |
        3. Main > Double-Click on file with "Conflicted (Both modified)"
        4. Conflict Solver

1. Ensure [All] is selected for the best veiw of the conflict.
2. Either
    1. [Take Left] > | Save |
    2. [Take Right] > | Save |
    3. Edit your file manually in your editor.
3. Close the conflict solver.

5. [Commit] ... the resolved changes.
6. [Push] to origin/topic-branch

2. Do "@Github. Open a Pull Request on Github." above
3. "@GithubOwner. Evaluates pull request." above
4. Do "Contributer responses to Owner's Pull request decision" above.