

Git Training Part-2

CONTENTS

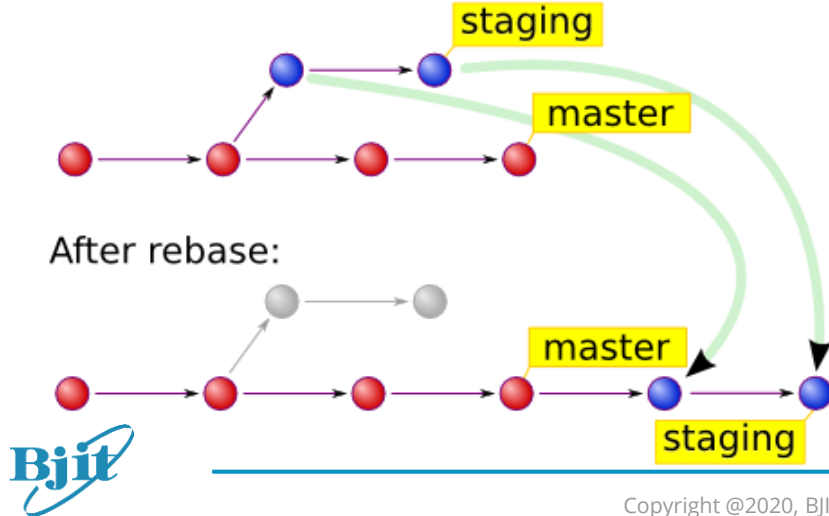
- Git Rebase
- Git Hooks
- Git Commit message guideline
- Git Commit using GUI tool
- Update your commit – commit amend using GUI tool
- Important Git Command
- Working style – best practices

GIT REBASE

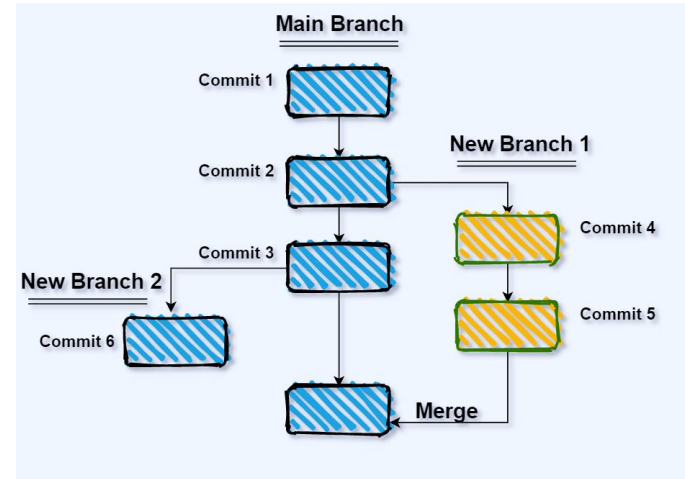
Git rebase allows you to integrate changes from one branch onto another. It is used to modify the commit history of a branch. Once the rebase is complete, your branch's commit history will be updated with the changes from the source branch, resulting in a more linear sequence of commits. Here's how the basic process of using git rebase:

Git rebase command:

`$ git rebase <branchname>`



Git Merge



GIT Hooks

Git Hooks are build-in feature in Git. Git Hooks are scripts that run before/after certain events or actions. If Git hooks script failed, then it prevents the actions from happening. For example, if commit message does not maintain conventional commit message, it prevents to commit the code.

There are two groups for Git hooks. One is server-side hooks, and another is client-side hooks. Server-side hooks runs on your remote repository. It is triggered before/after certain events/actions happened on your remote repository. Client-side hooks are triggered on your local repository.

Why we use Git hooks?

- Increase productivity
- Save time
- Run custom scripts
- Enforce standards that prevents error

GIT COMMIT MESSAGE GUIDELINE

This is a way to tell Git about the staged changes you made to the repository using the commit command. Follow these guidelines when writing good commit messages:

- Capitalized, short (50 chars or less) summary of changes.
- Put a blank line between the summary and description.
- Wrap the description 72 characters or less. Use a hyphen (-), followed by a single space with blank lines in between for bullet points if needed.
- Put a blank line.
- Footer line. The `footer` is usually an external reference, like a Jira/Redmine ID or issue number.
- Write the summary line and description of what you have done in the imperative mood.
- Imperative mood: a verb form that makes a command including orders, request, advice, instructions and warnings.

GIT COMMIT MESSAGE GUIDELINE

- A standard commit should contain the following sections:
 - <Sub-system>: <Summary>
Sub-system: Name of the system, Optional if there is no sub-system in particular.
Summary: Concise/Core description of the commit.
 - <Body>
Detail of the commit message. If the summary can completely express everything, there is no need for a message body.
 - <Footer>
Fixes #redmine-id , represents the bug
Refs #redmine-id , represents a continuation of a task. A task may have several commit to complete.
Closes #redmine-id , represents the last commit of the task. Task should be closed.
- Full Conventional Commit Example.
Calendar: add calendar.h file to fix the build error

Without this file, it gives error - calendar.h file not found.

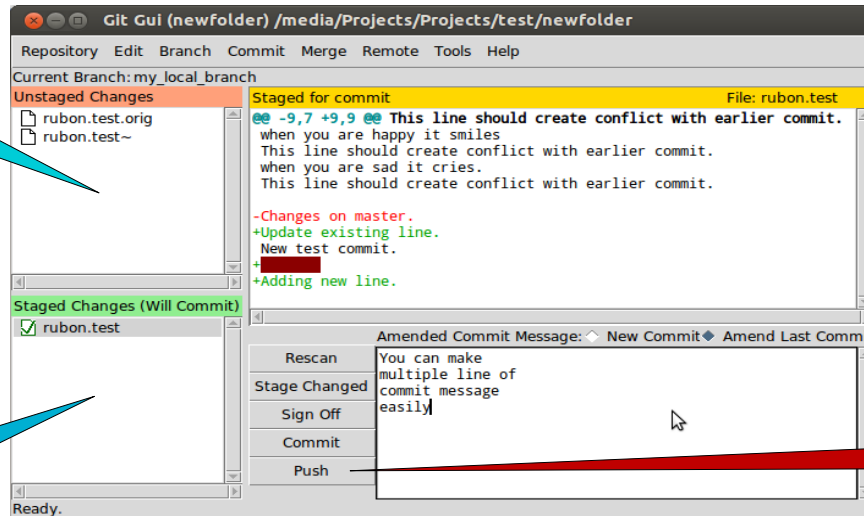
Fixes: Refs. #1234
- An example of a git command or you can use GUI tool.
\$ git commit -m <Summary> -m <Description> -m <Footer>

GIT COMMIT USING GUI TOOL

- You may need to update your last commit. You may don't want to make new commit, instead you need to update your last commit. Following command would work.
\$ git commit --amend -m "<Your commit message here>"
\$ git commit --amend --no-edit
- You can also use gui tool for this purpose, **Git GUI**.
- Using git gui you can easily add/remove files from stage, can see your changes, red marks, write multiple line of commit message and can commit. **But you should not push from git gui.**
\$ git gui

Unstaged files
Will not commit

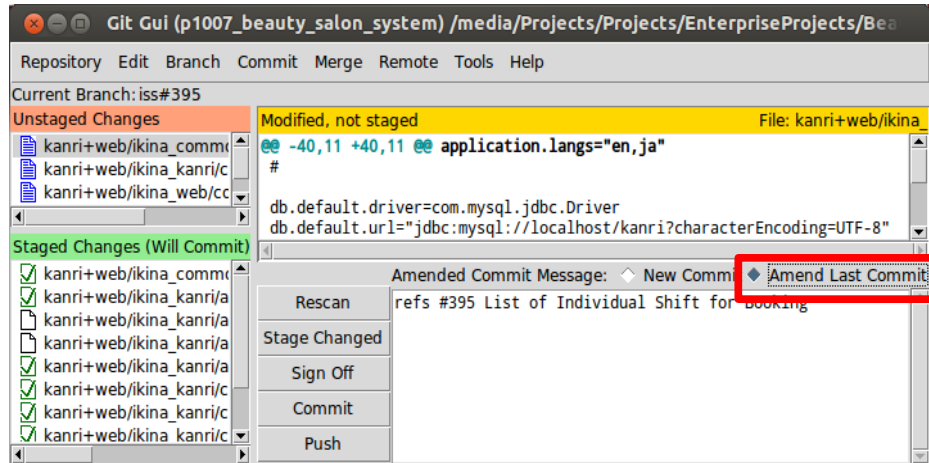
Staged files
Will commit



Don't push
from here

UPDATE YOUR COMMIT – commit amend (1/2)

- You may need to update your commit for various reasons:
 - You may find a problem/bug by yourself. So, you need to update your commit.
 - Reviewer might give review feedback that you need to fix. So, you need to update your commit.
 - For any reason you checkout to different branch without finishing your current task. So, you committed partial task. Now you come back to that work, finish your task. So, you need to update your commit.
- You can update your last commit very easily. Here is the command.
 - \$ git commit --amend -m "Write your comment here"
 - \$ git commit --amend --no-edit // keep the last commit message
- Using the Git GUI makes the process much simpler:



GIT IMPORTANT COMMAND

- To Undo a Conflict
 - \$ git merge --abort
 - \$ git rebase --abort
- To copy specific commit to different branch
 - \$ gitk // Copy the sha1 value
 - \$ git checkout <branchName>
 - \$ git cherry-pick sha1
- To remove last commit/specific commit
 - \$ git checkout <branchName>
 - \$ git reset --hard HEAD~1
- To recover deleted commit
 - \$ git reflog
 - \$ git checkout <sha>/HEAD@{N}
- To find Hash ID of local/remote branch
 - \$ git rev-parse <branchname>
 - \$ git rev-parse origin/<branchname>
- To recover deleted branch
 - \$ git reflog
 - \$ git branch <branchname> ID
 - \$ git checkout -b <newBranchName> ID

GIT IMPORTANT COMMAND

- To check merged branches
`$ git branch --merged`
- To check no-merged branches
`$ git branch --no-merged`
- To search by date --before/--after
`$ git log --after="2022-11-1"`
`$ git log --after="2022-11-1" --before="2022-11-30"`
- To search by message --grep
`$ git log --grep="resolve"`
- To search by author --author
`$ git log --author="Ujjal"`
- To search by file -- <filename>
`$ git log -- README.md`

WORKING GUIDELINE – BEST PRACTICES

- Don't work on master branch, develop branch, or your main branch. Work only in your local branch.
- Create your local branch for a specific task.
- After finishing your task, commit it and push it to remote repository.
- For every separate tasks or tickets create separate local branches, work on that branch, commit and push to remote repository.
- If there is any review feedback, switch to that branch, fix and commit additional patch-set.
- Let's see the steps of a total workflow:
 - \$ git clone or git pull // to take latest update
 - \$ git checkout -b iss#303 // create and checkout to local branch
 - Finish your task
 - \$ git commit // only commit, before push, you should take latest update and merge, push
- Always keep your master or main development branch clean. This will help you to take latest updated source by Git pull.
- Let's think about another scenario. Suppose you are working on a feature in master branch. During your development you receive another high priority bug fixing task. What will you do?
- The simple solution would be if you work on your local branch. Commit your current incomplete task in your local branch. Create/Checkout to the high priority bug task branch, finish it, push it. Now back to your earlier feature task local branch, finish your task, amend a commit and push.

A white L-shaped graphic consisting of a horizontal line and a vertical line meeting at a right angle, positioned to the left of the "Thank you for watching" text.

Thank you for watching

A white L-shaped graphic consisting of a horizontal line and a vertical line meeting at a right angle, positioned to the right of the "Thank you for watching" text.

Get in touch with us:

www.bjitgroup.com

