

Intro to Git

Joseph Gan

9 Nov 2022

Commit Message

- Based on [conventionalcommits](#)
- Structure

```
<type>[optional scope]: <description>  
<BLANK LINE>  
[optional body]  
<BLANK LINE>  
[optional footer(s)]
```

```
feat(profile): implement profile API
```

```
With this commit, it adds profile CRUD API with OpenAPI documentation and  
controller sliced test
```

```
Close #1
```

Commit Message

- Types
 - build, ci, docs, feat, fix, perf, refactor, style, test
- Scopes
 - To be defined at a later stage
- Description
 - Short summary (≤ 90 characters)
- Body
 - As much as you want, be as specific as possible
- Footer
 - To reference or close an issue

Enforcement

- Maven [git-build-hook](#) plugin
- Setup a pre-commit hook to enforce

Tools

- CLI
 - git-bash
- GUI
 - [x] [GitExtensions](#)
 - [SourceTree](#)
 - [Git-Fork](#)

Tools

- IDE (Built-in)
 - IntelliJ
 - [x] [gittoolbox](#)
 - VSCode
 - [x] [gitlens](#)
 - [x] [gitextensions](#)

Tools

- [x] `kdiff3`
- `diffmerge`
- IDE built-in diff

Concept / Terminology

- .gitignore
- hooks (.git/hooks)
 - pre-commit, post-commit, pre-rebase, etc

Feature Demo

- Amend
- Checkout
- Switch
- Reset
 - Mixed, Soft, Hard
- Reflog
- Revert
- Stash

Feature Demo

- Merge
 - No Fast Forward
 - Fast Forward
 - Dealing with Conflicts
- Rebase
 - Rebase Interactively
- Cherry Pick
- Worktree

Feature No Demo

- Bisect
- Submodules

Warning

- **Never** rebase + force push in `main` (or any public branch)

Best Practices

- Write good commit message
- Prefer Clean History
- [Therefore] Rebase >>> Merge FF >>> Merge No FF
- Git pull (set rebase as default)
- Practice local feature-branch (easy to switch to main)
 - Practice small commits locally, squash when necessary prior to push or merge back to main

Best Practices

- Gitlab
 - FF Merge as default [who looks at merge commit anyway]
 - Configure default ff merge in Gitlab

Merge method

Determine what happens to the commit history when you merge a merge request. [How do they differ?](#)

- ☐ Merge commit
Every merge creates a merge commit.
- ☐ Merge commit with semi-linear history
Every merge creates a merge commit.
Merging is only allowed when the source branch is up-to-date with its target.
When semi-linear merge is not possible, the user is given the option to rebase.
- ☒ Fast-forward merge
No merge commits are created.
Fast-forward merges only.
When there is a merge conflict, the user is given the option to rebase.
If merge trains are enabled, merging is only possible if the branch can be rebased without conflicts. [What are merge trains?](#)

Best Practices

- Pull/Merge Request
 - Team dependent
 - Small >>> Huge [reduce overhead + context]
 - Pair programming eliminates it (above)
 - Trunk based development
 - Protected by well defined pipeline