LAB II LECTURE
Developing with Git

Seoul National University
Graphics & Media Lab

Today's Contents

- Introduction
- Setting up Git / Git Basics
  - Install and Config
  - Init local
  - Clone
  - Make changes in git repository

- Basic Branching / Merging

Big Game Company

Graphics

User Interface

Simulation

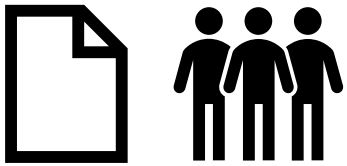
Game Management

Battle Management

Etc…

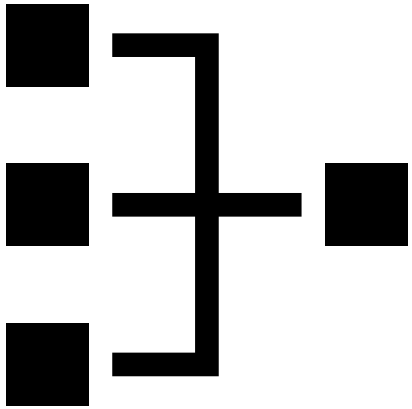
Working Together

Obtain well-cleaned source files without any conflicts to other source files



Graphics.cpp

- Graphics
- User Interface
- Simulation
- Battle Management
- Game Management
- Etc…

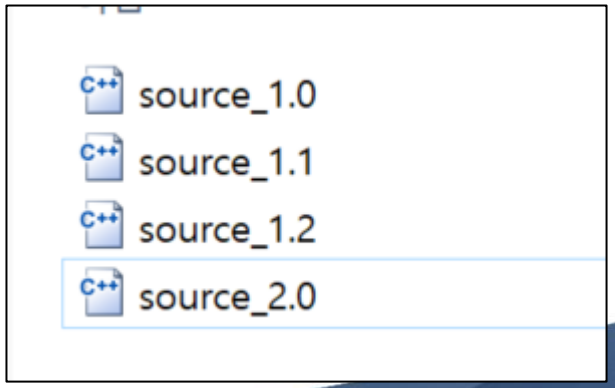


Version Control

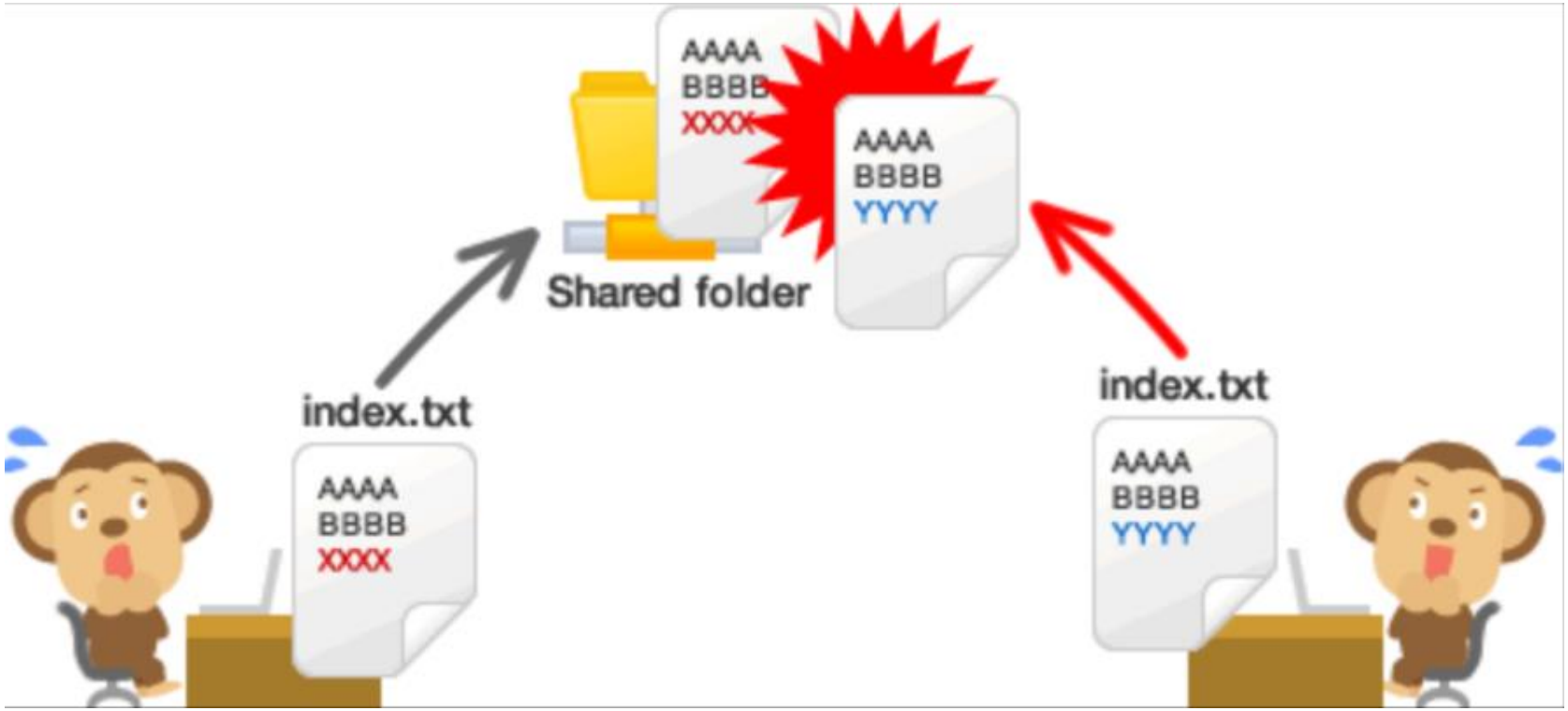
- We want to keep previous works
  - We may want to restart from previous work
- The simplest way => Make copies
  - Too many copies in the end: 100+ revisions = 100+ copies?
  - Hard to distinguish between versions

- Things will get uglier in a team projects

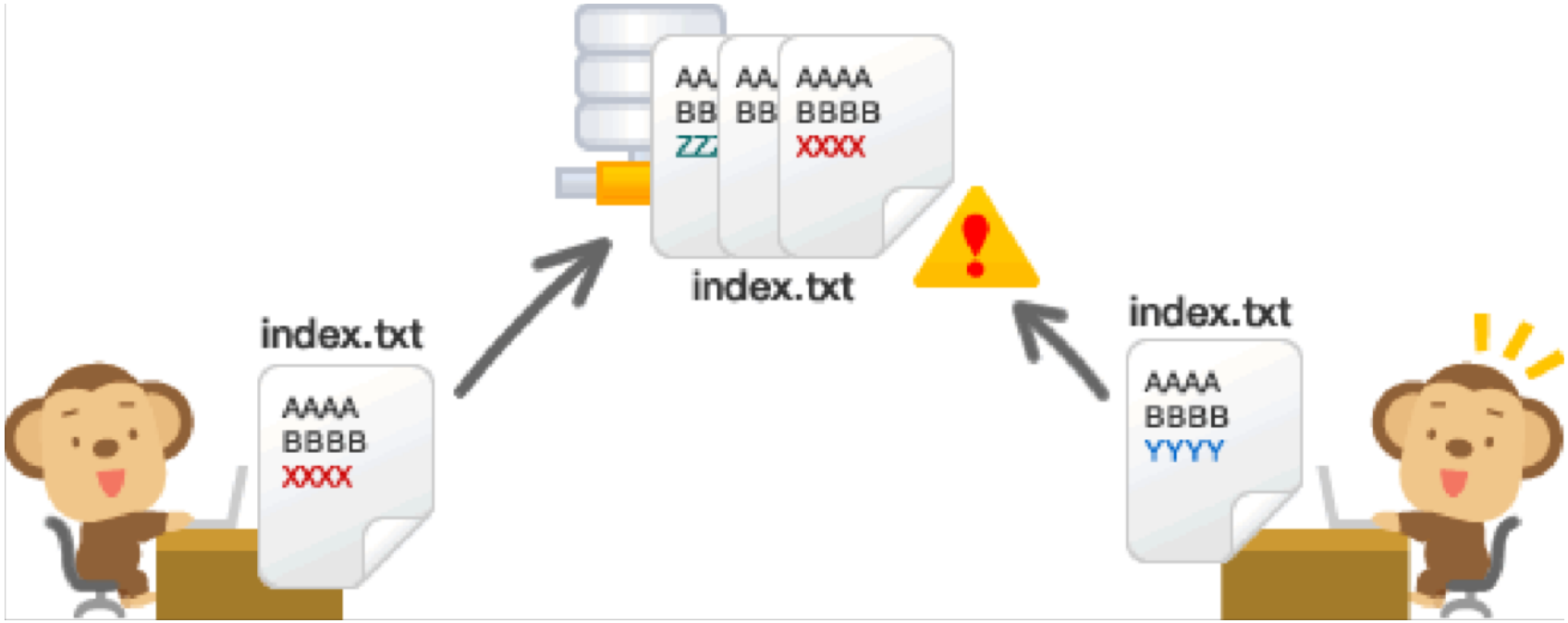
문서집계표(최종).HWP
문서집계표(최종수정).HWP
문서집계표(최종수정컨펌).HWP
문서집계표(컨펌V1).HWP
문서집계표(컨펌V2).HWP
문서집계표(컨펌V3).HWP
문서집계표(진짜최종).HWP
문서집계표(진짜진짜최종).HWP
문서집계표(진짜진짜진짜최종).HWP
문서집계표(회장님).HWP
문서집계표(회장님지시수정).HWP
문서집계표(회장님수정.V1).HWP.....



Conflicts occur when working in teams

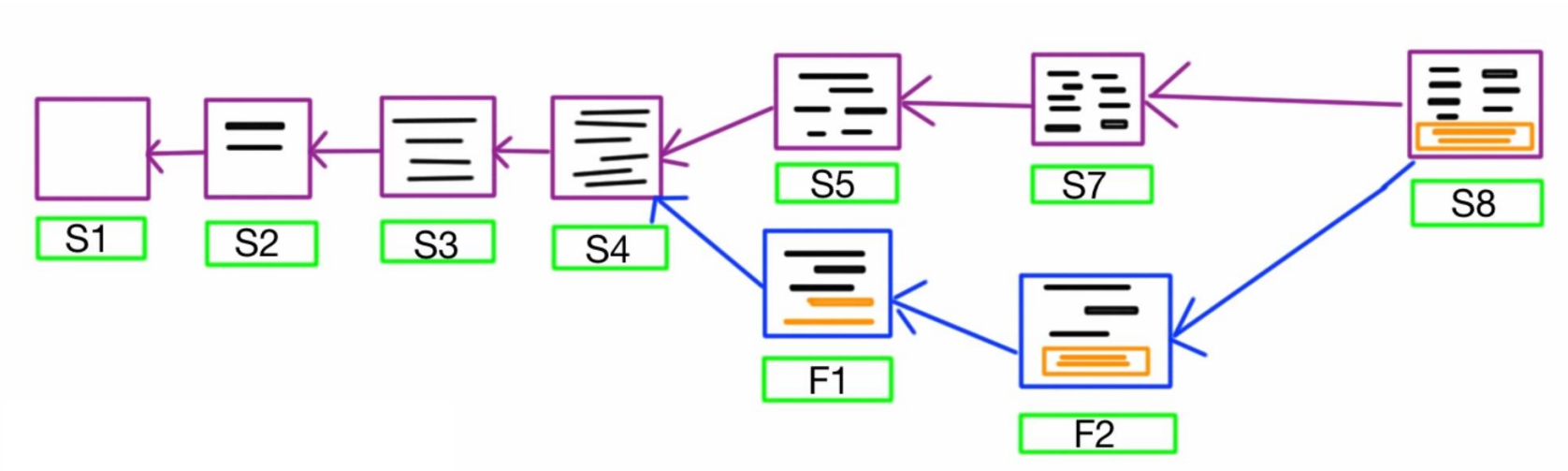


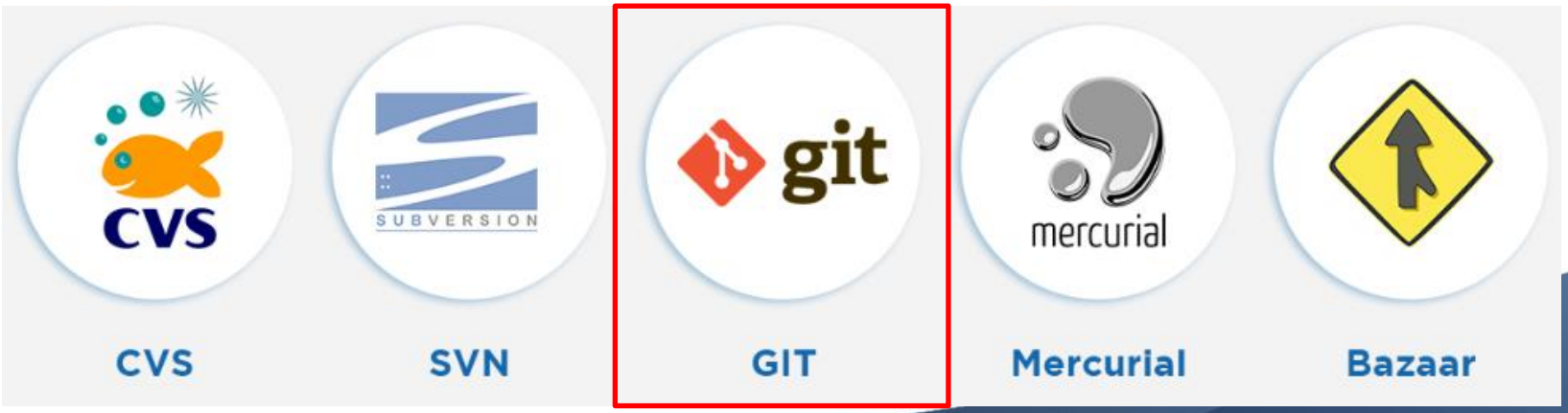
Version Control System (VCS) Supports Teamwork



What is Version Control System?

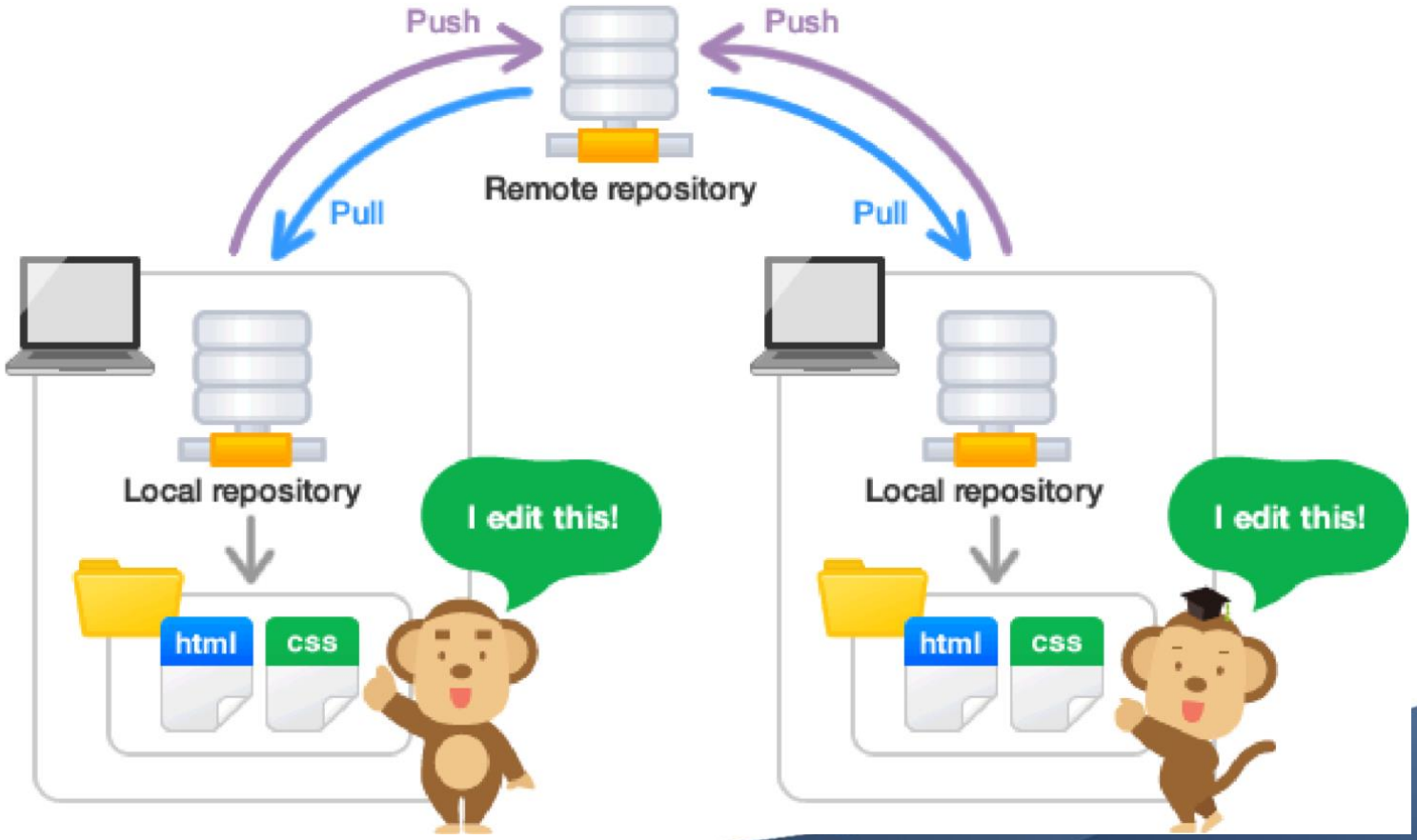
- A system that records changes to the files over time
- User can restore specific versions at any time





What is Git?

- A distributed control system
- Clients fully mirror the repository to the local environment, work in local environments, and updates changes to the server on demand



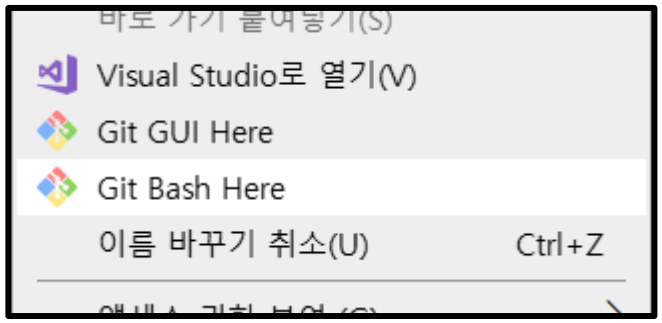
Why Git?

- Developments could be continued even if the server (remote repository) went down
- Each developer can create their own specific revisions and could be merged later
- Many open-source projects use Git and can be easily found in GitHub (https://www.github.com)

Setting up Git

- Download from https://git-scm.com/downloads

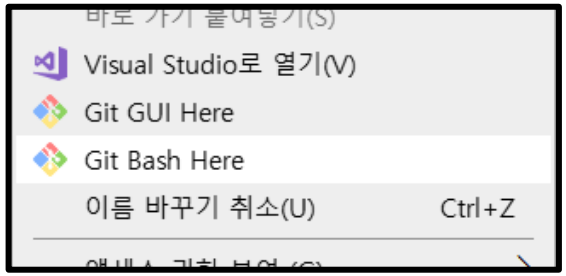
- Invoking Git commands
  - Windows: open a console for Git
    - Right click on the folder ->Click 'Git Bash here'

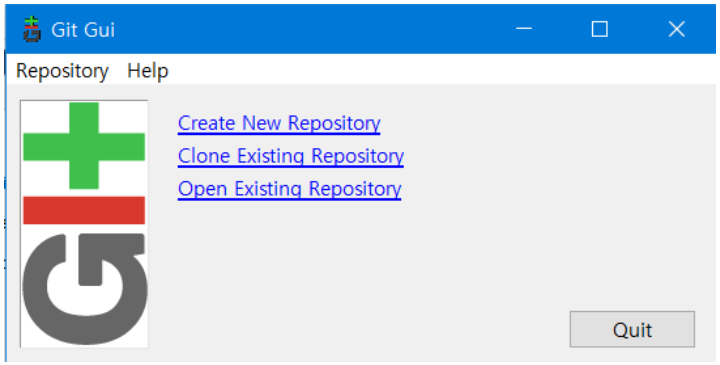


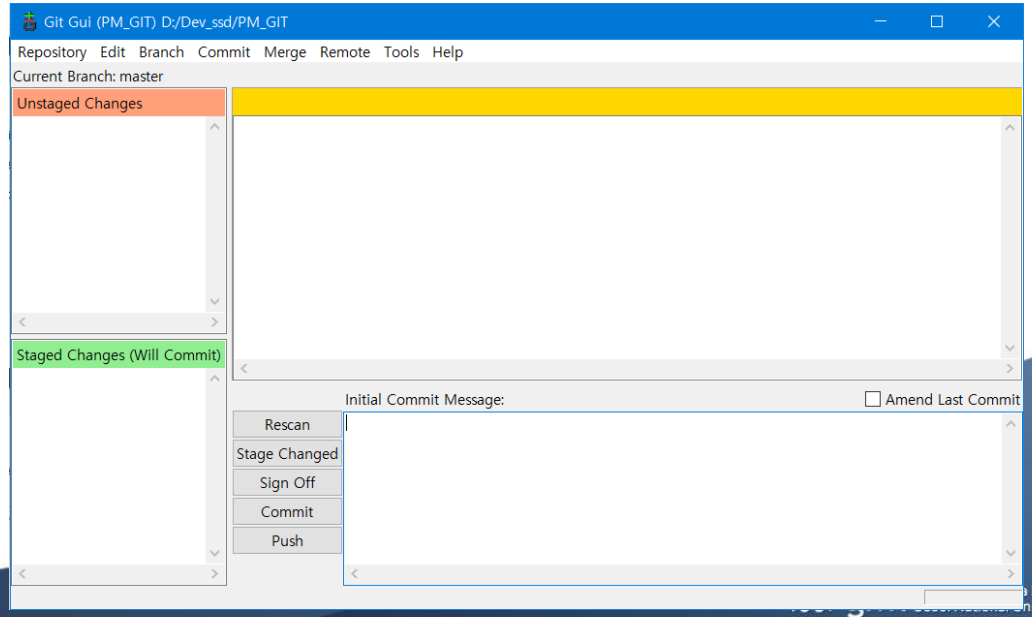
- Basic setup (user name and e-mail)
  - git config --global user.name "Your Name"
  - git config --global user.email yourmail@example.com
- Check setting with
  - git config --global --list

Git Basics: Creating a Git Repository

- To make current working directory as a git repository
  - Using git CLI
    - Open git bash as previous slide where you want to make a git repo.
    - Type git init
  - Using git GUI
    - Open git GUI in the same way as you opened git bash (Git GUI Here)
    - Select "Create New Repository"
    - Choose a directory into which you want to make repo

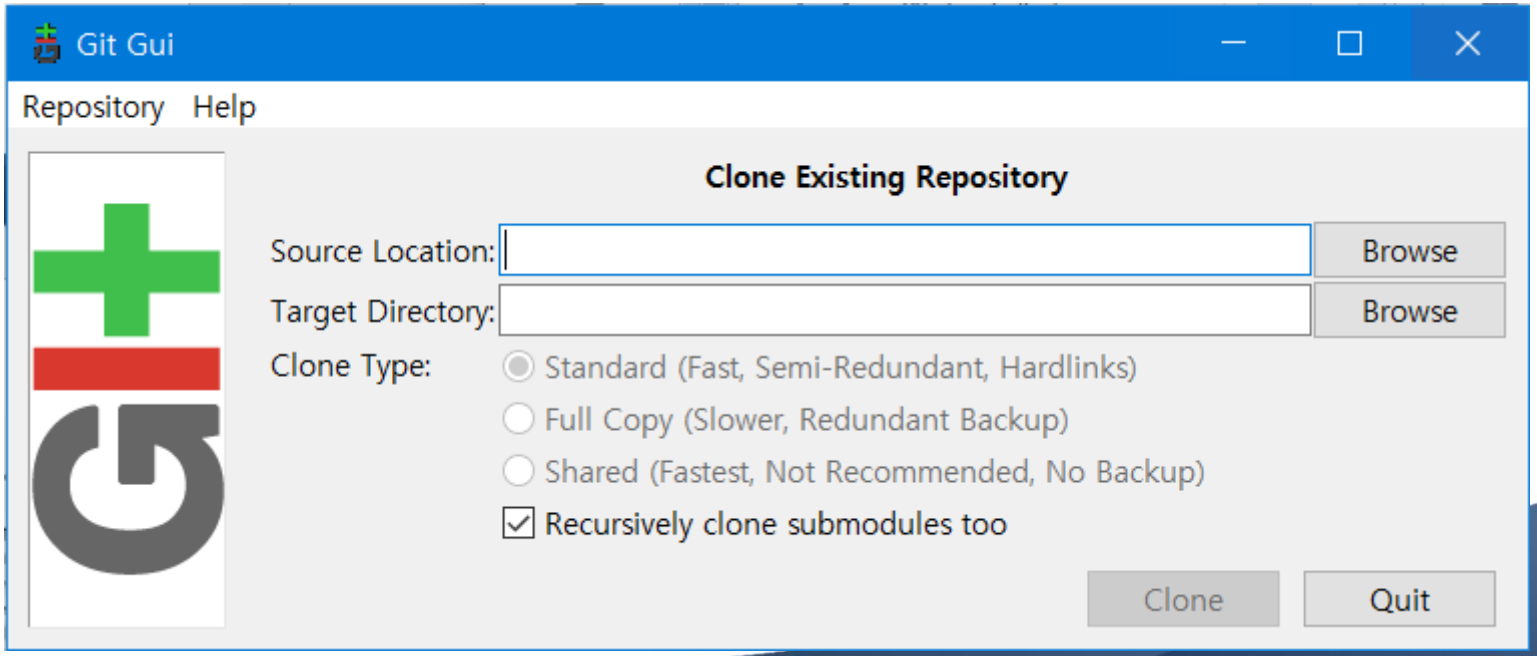






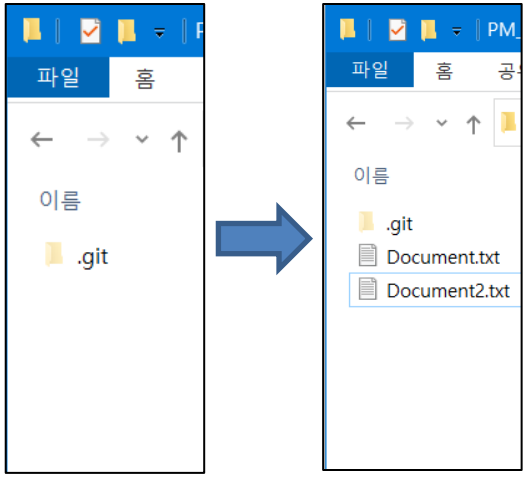
Git Basics: Cloning a Git Repository

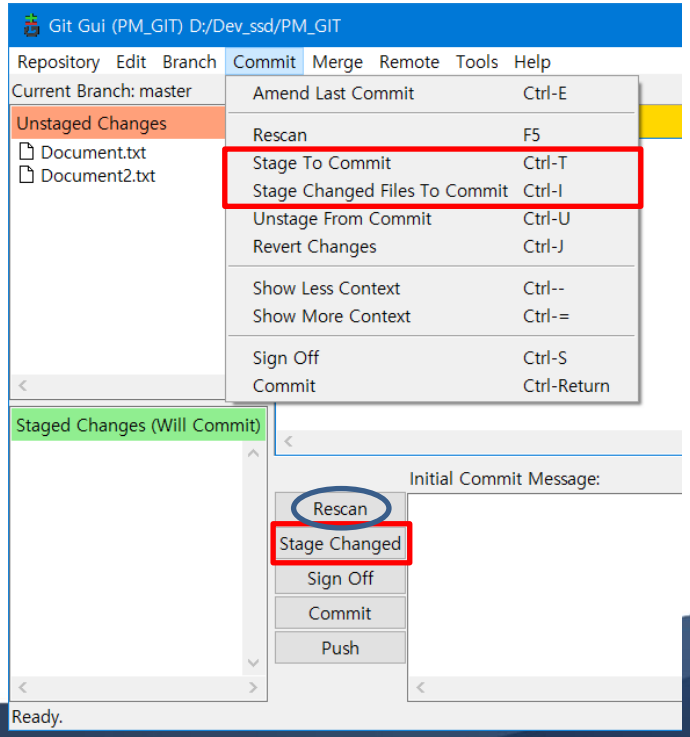
- Or you can clone existing git repository into your workspace
  - Using Git CLI
    - git clone <URL/path to a Git Repository>
    - git clone <URL/path to a Git Repository> <target directory>
  - Using Git GUI
    - Select "Clone Existing Repository"

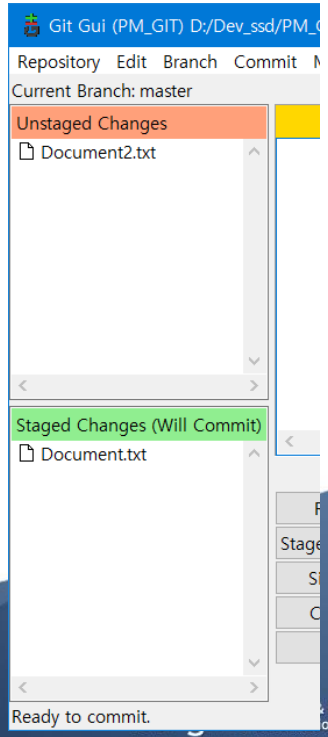


Git Basics: Making Changes in Git Repo Staging

- New or modified files are noticed in GUI, as 'Unstaged Changes'
  - If changes are not shown, click 'Rescan'
- Changes can be added to git by staging
  - Only files that are staged will be committed and versioned in git
  - Stage can be done by clicking Stage Changed Files to Commit (Everything) / Stage To Commit (Selected)

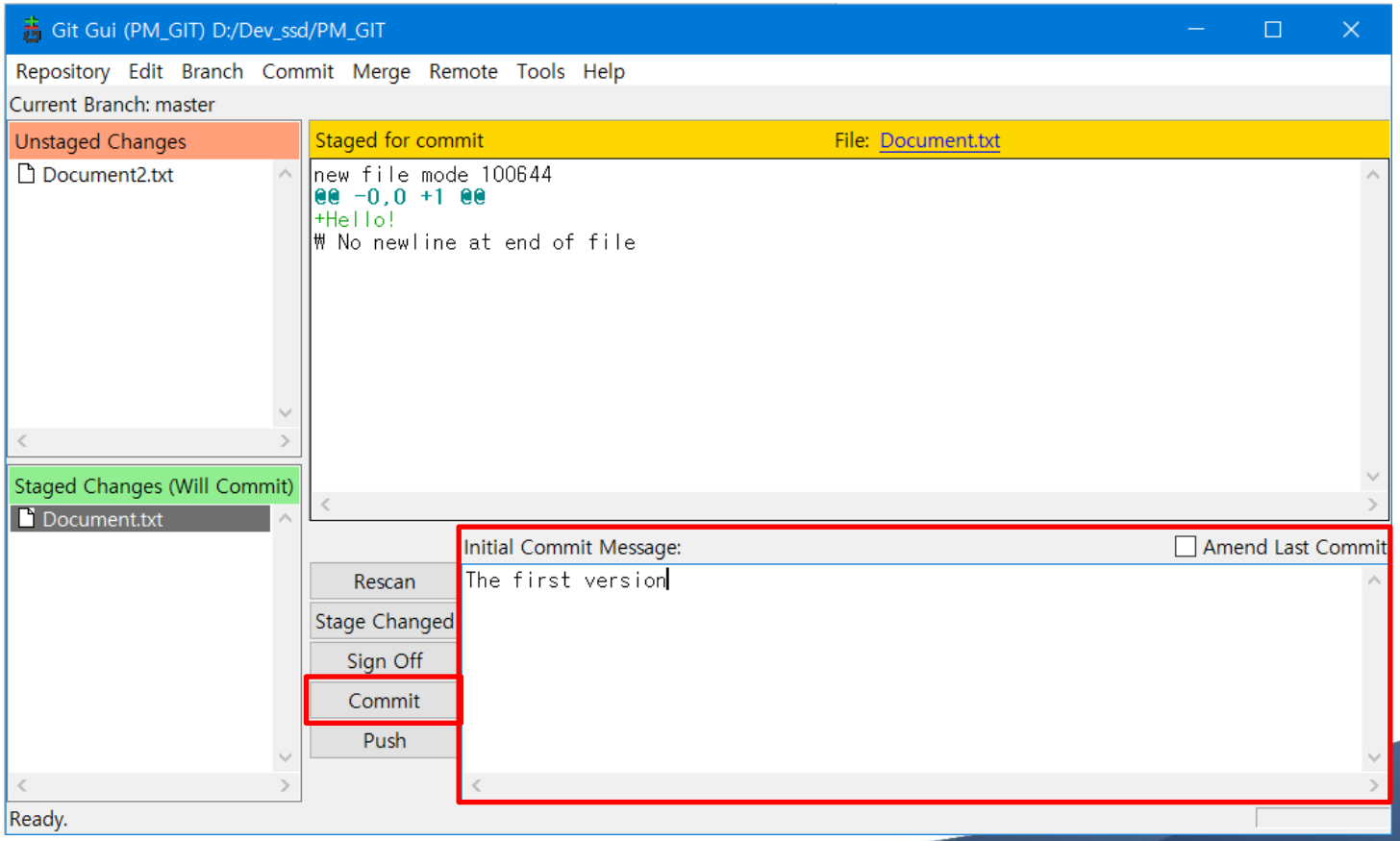






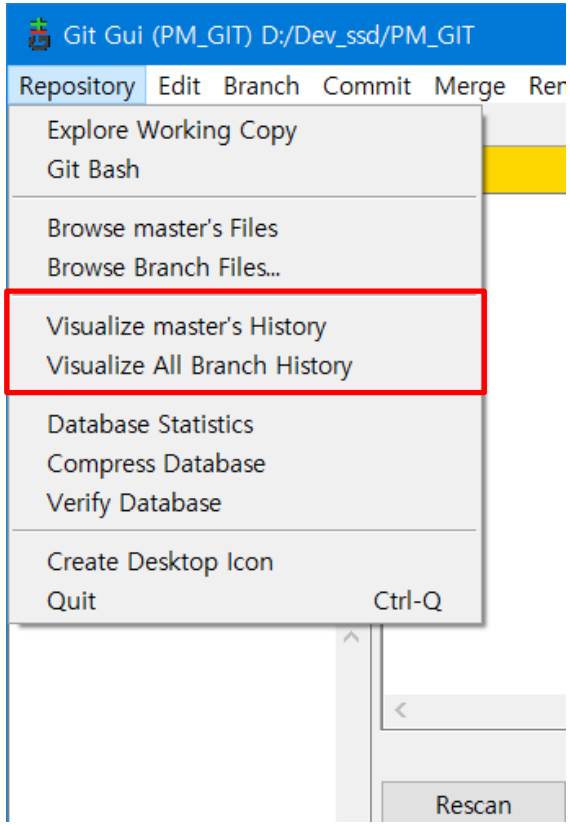
Git Basics: Making Changes in Git Repo Commit

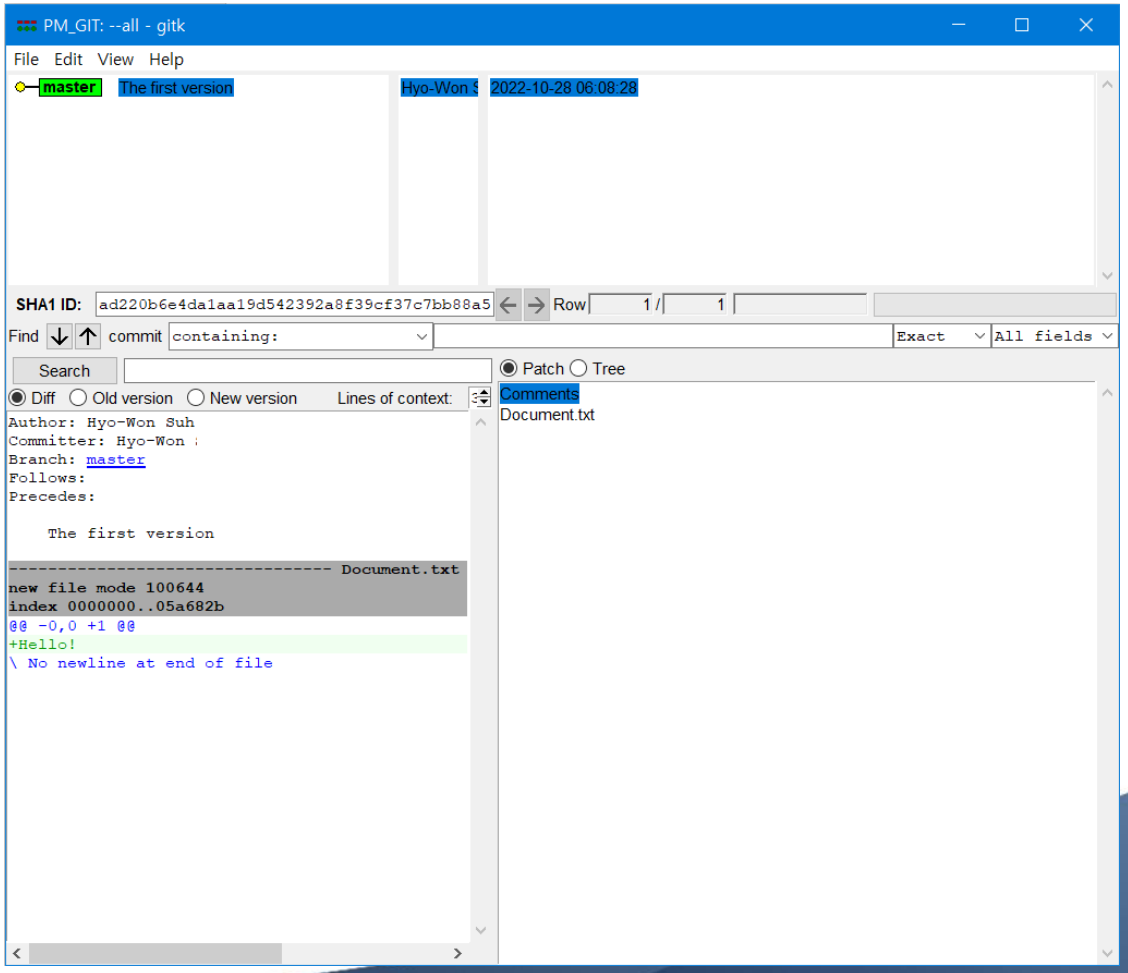
- To make a version from staged changes, type a message and click 'Commit'



Git Basics: Version History

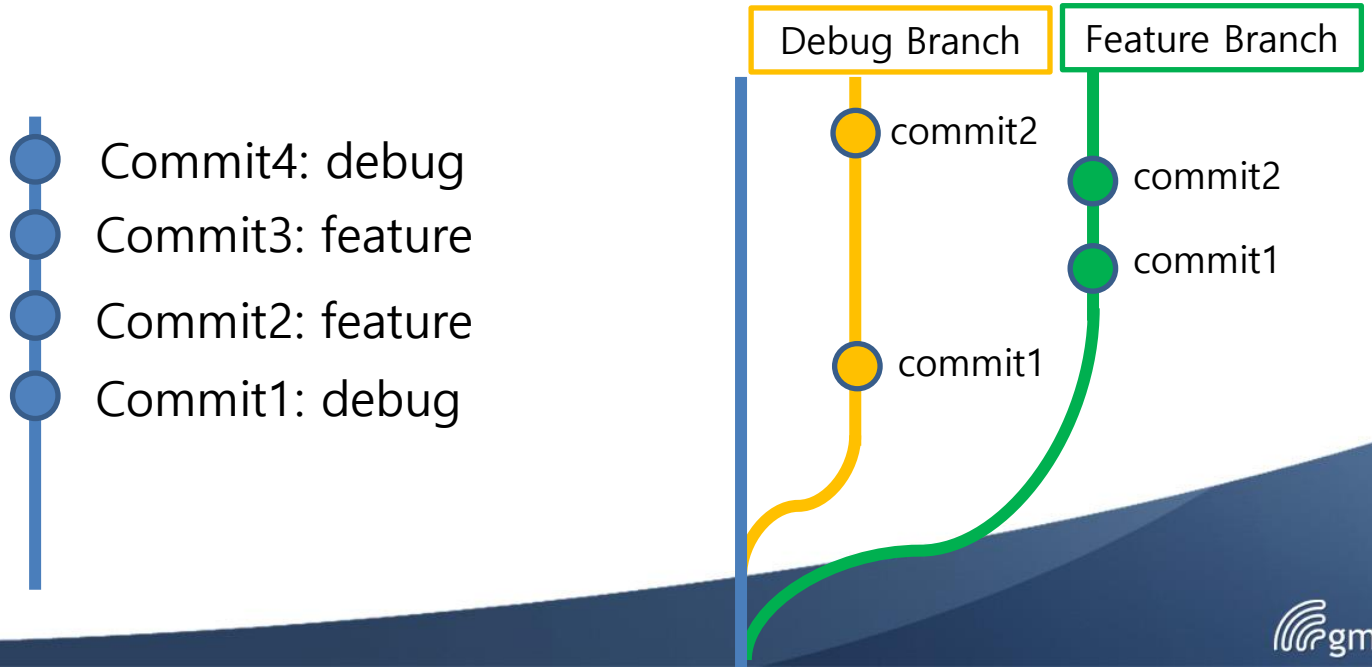
- Version history can be viewed in ‘Visualize All Branch History’





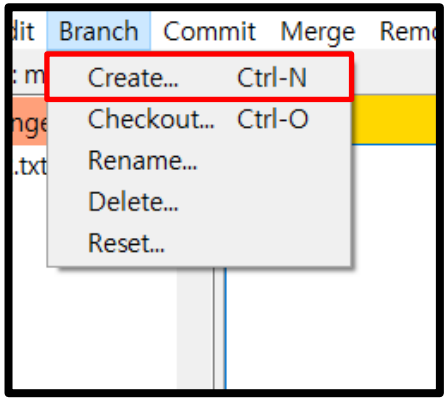
Branching

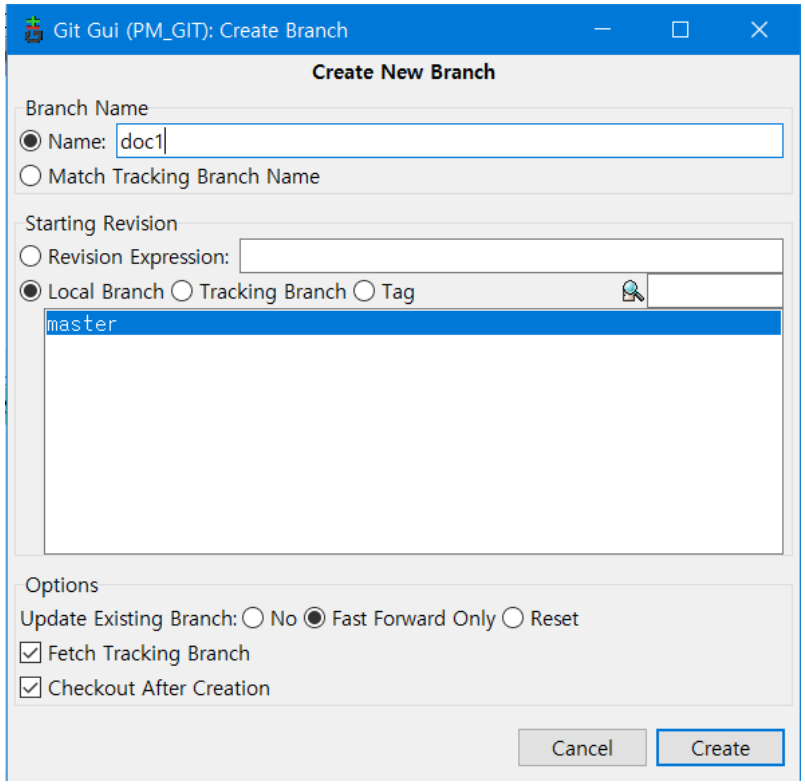
- In a collaborative environment, it is common for several developers to share and work on the same source code
  - For instance, some developers is fixing bugs while others are implementing new features
  - In such case, committed versions is hard to follow because it is hard to know which part of the project the commit is related to
- Make a "branch" with meaningful name which will be an independent line of development (debug, feature, etc.)
- Branches make the history of the version easy to read



Branching in Git

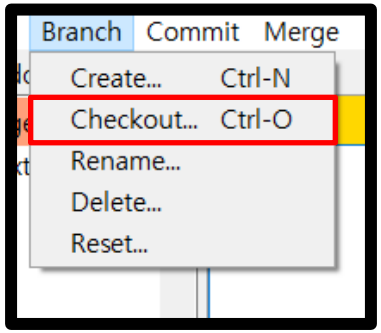
- Every git repository has the ‘master’ branch by default
- Additional branches can be created using ‘Branch>Create’ menu

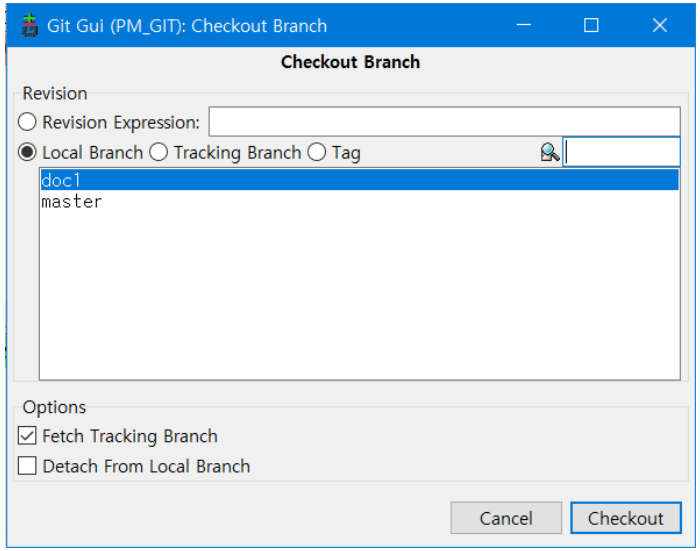


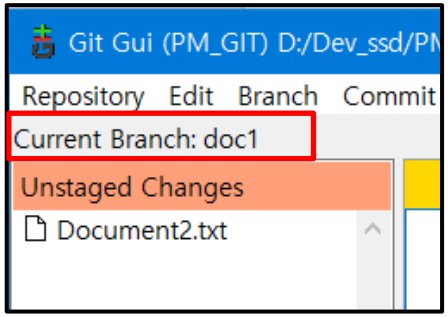


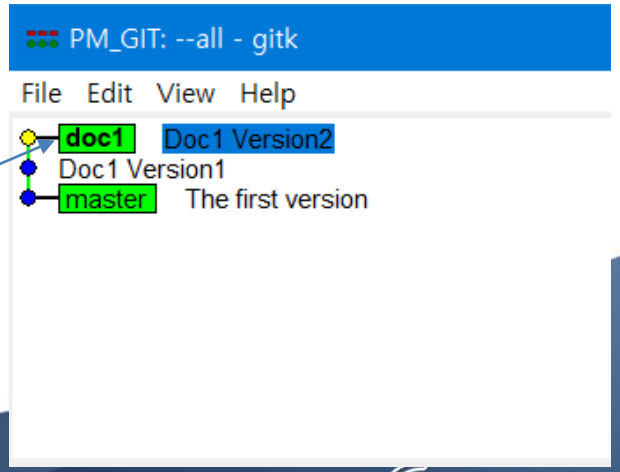
Switching Branch

- "Checkout" the branch you wish to work on
  - Branch>Checkout





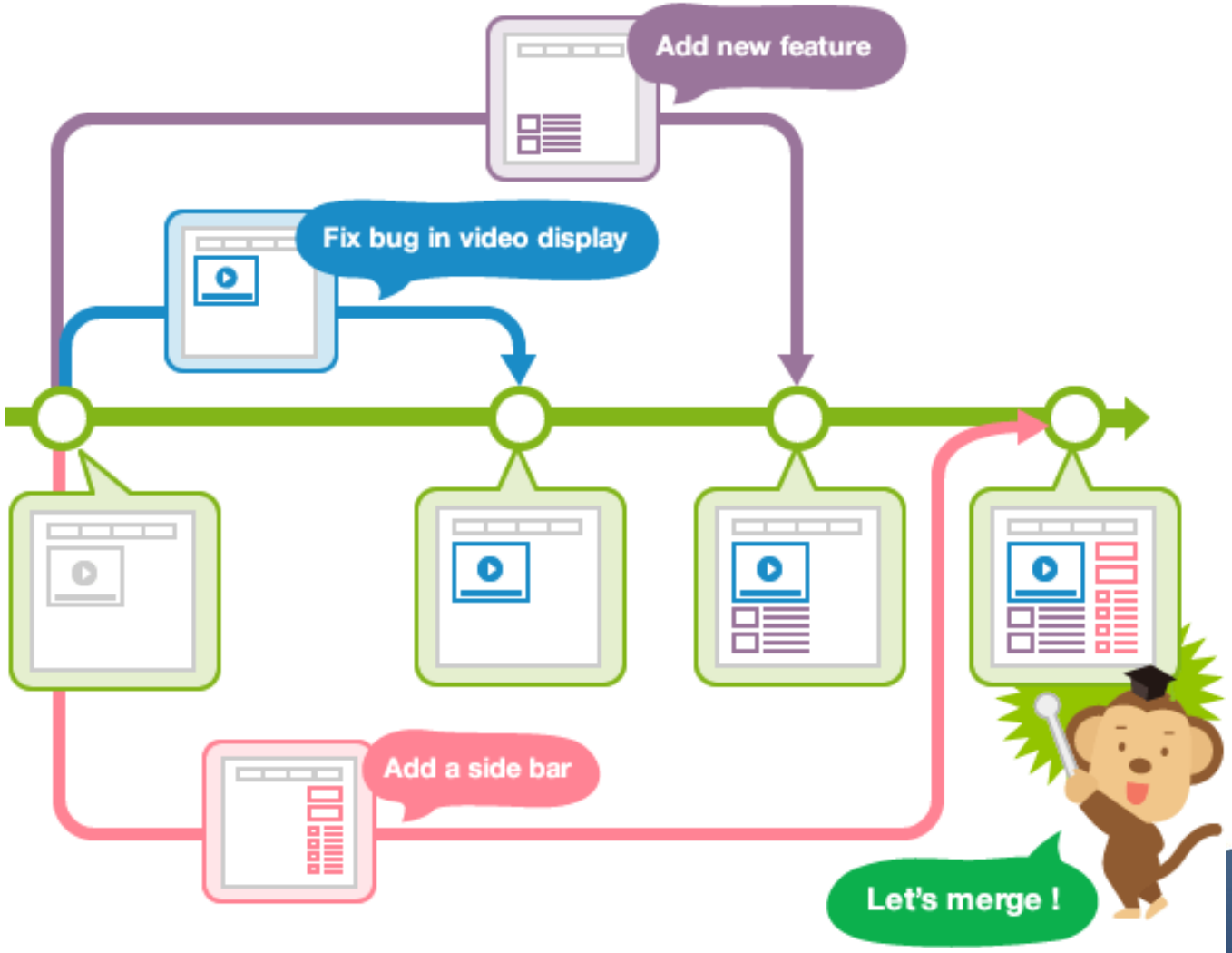




HEAD
last commit
of a branch

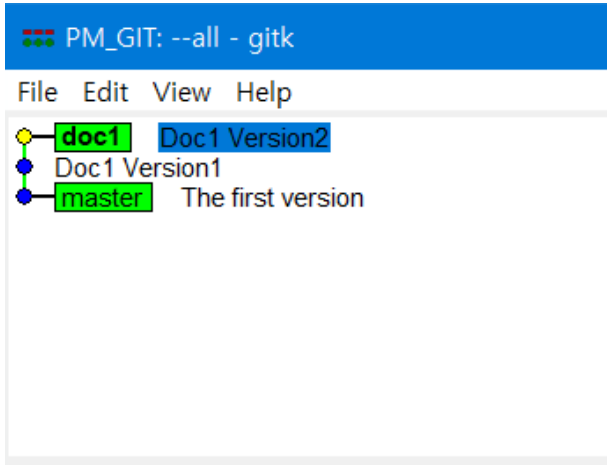
Merging

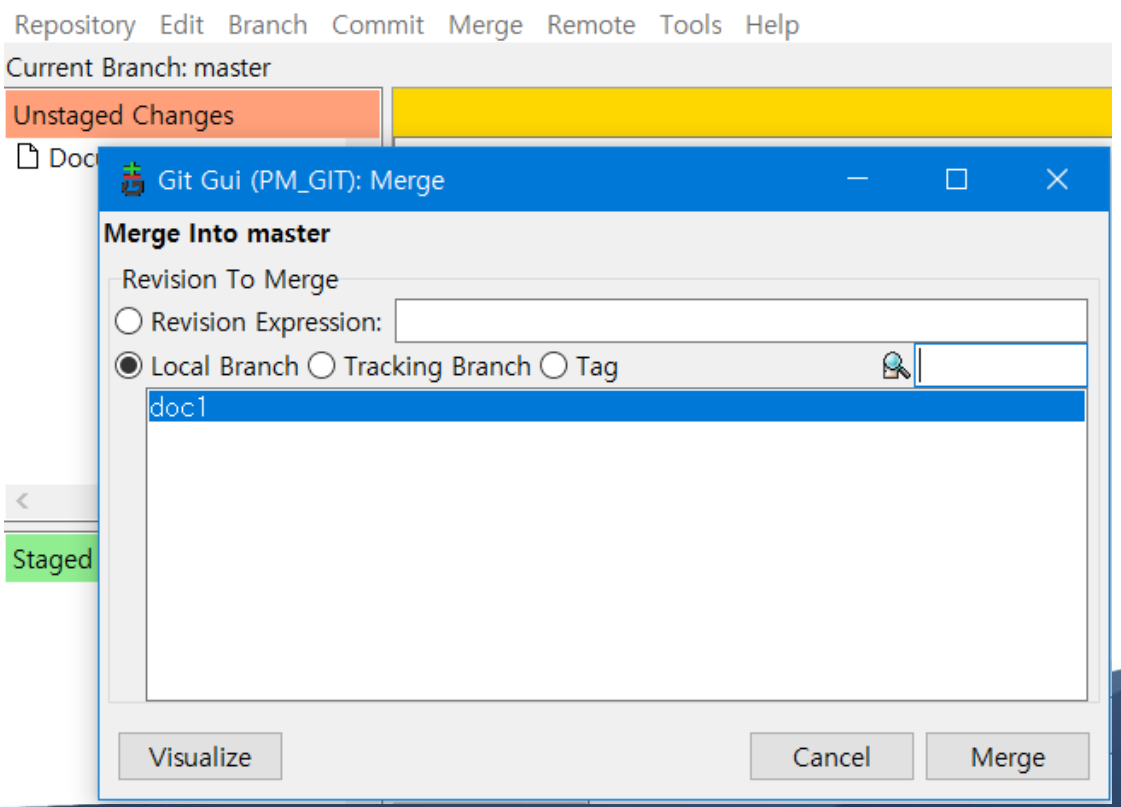
- Different branches can be **merged**



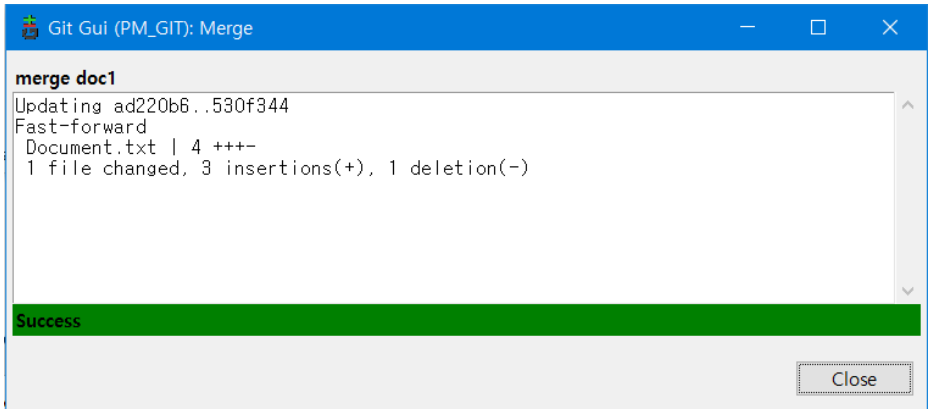
Merging

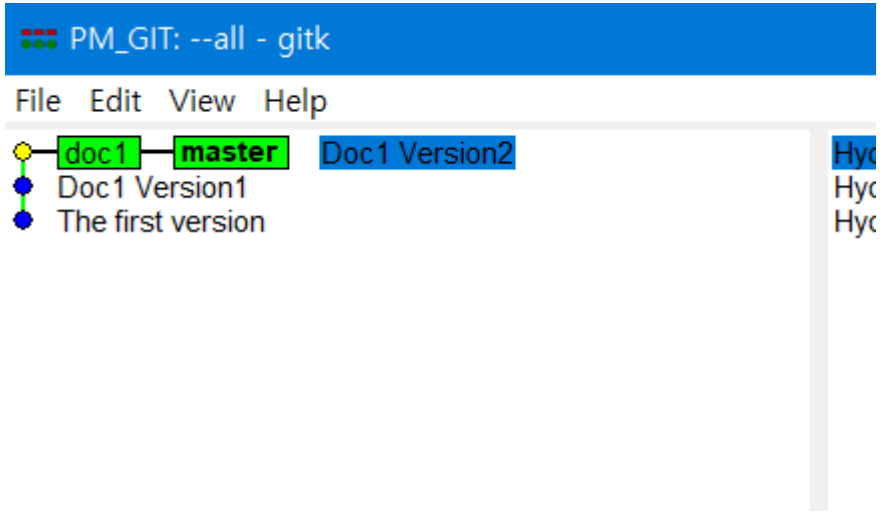
- Combine the revision of Doc1 to master by merge menu
  - Checkout to master branch first
  - Go to Merge>Local Merge and select a branch to merge





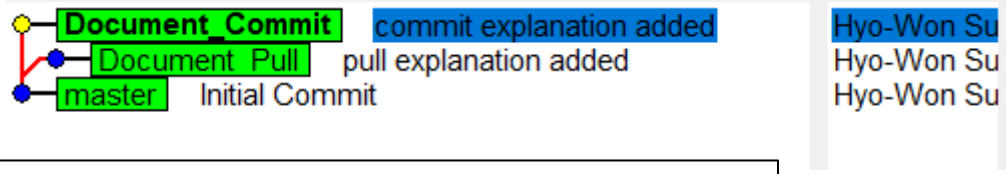
Merge Result





Conflicts

- Automatic merge cannot be performed when there are any 'conflicts' between branches
  - For instance, a file is modified in both branches, so that contents are not same
  - In the example, we are going to merge Document_Commit and Document_Pull branch into master



Git commands even a monkey can understand
add: Register a change in an index

master HEAD

Git commands even a monkey can understand
add: Register a change in an index
pull: Obtain the content of a remote repository

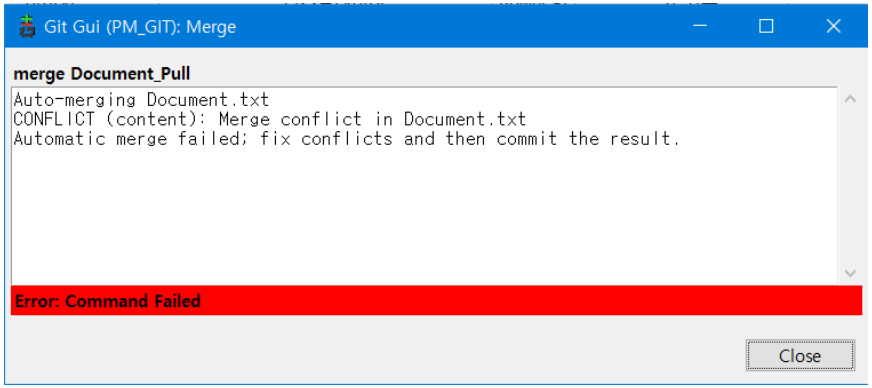
Document_Pull HEAD

Git commands even a monkey can understand
add: Register a change in an index
commit: Save the status of an index

Document_Commit HEAD

Conflicts

1. Merge Document_commit into master
   - Automatic merge should be done without any conflicts
2. Merge Document_pull into master
   1. Conflicts occurs and automatic merge fails



Git commands even a monkey can understand
add: Register a change in an index
commit: Save the status of an index

Document_Commit HEAD
Master HEAD

Git commands even a monkey can understand
add: Register a change in an index
pull: Obtain the content of a remote repository

Document_Pull HEAD

Resolving conflicts

- To resolve conflicts, modify files which caused conflict

```
  Git commands even a monkey can understand
  add: Register a change in an index
- commit: Save the status of an index
 -pull:Obtain the content of a remote repository
  <<<<<<< HEAD
++commit: Save the status of an index
  =======
++pull:Obtain the content of a remote repository
  >>>>>>> fc0dec9d83a4724cde9b6f962f6f02a3a5bdb12a
```

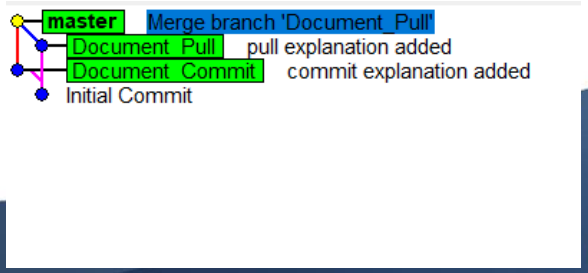
Version in HEAD (currently master)

Version in Incoming (commit ID)

Modify

Git commands even a monkey can understand
add: Register a change in an index
commit: Save the status of an index
pull:Obtain the content of a remote repository

Stage and commit



Appendix: Git CLI commands

- Status command (`git status`)
  - Check the staged (tracked), unstaged (new or modified, but not tracked) files
- commit command (`git commit –m “descriptions”`)
  - Committing last changes and make a version
- Log command (`git log`)
  - Viewing history of commit
  - Difference between files can also be checked using `–p` switch
- `.gitignore` file
  - contains a list of files which should not be tracked
  - log files, any temporary files, etc

Appendix: Git CLI Commands

- Undoing Changes
  - Commit with amend switch (`git commit --amend`)
    - Make additional changes to the latest commit
    - Useful to avoid excessive commits
- Unstaging files (`git reset HEAD FILENAME`)
  - Useful when files are changed, but want to be committed later
- Unmodifying files (`git checkout -- FILENAME`)
  - Reverting files to the latest commit

Appendix: Git CLI Commands

- Branch related
  - Create a branch (branch command)
    - `git branch NAME_OF_BRANCH`
  - Merge with another branch (merge command)
    - `git merge BRANCH_TO_MERGE_WITH`