CSE 403

Software Engineering

Winter 2023

Version control and Git

Today

- Version control: why, who, how?
- Git: concepts and terminology

Why use version control?

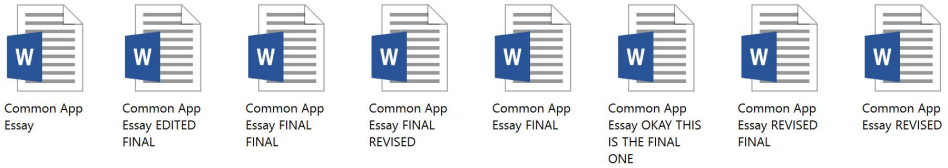
11:51pm

Why use version control?

11:51pm

11:57pm

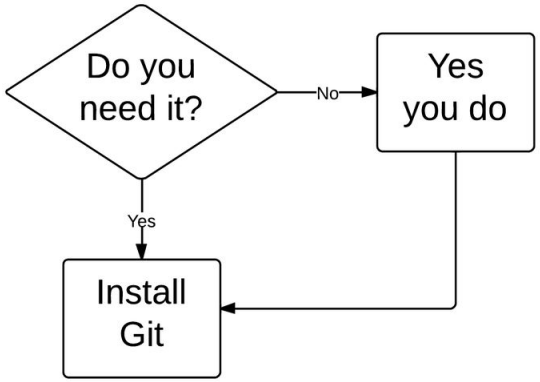
Why use version control?



Just kidding... this is far more realistic.

Version control

Version control records changes to a set of files over time.
This makes it easy to review or obtain a specific version (later).



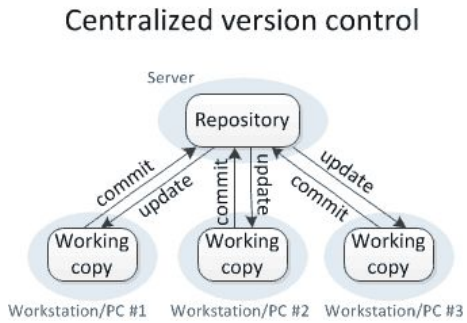
Who uses version control?

Example application domains

- Software development
- Research (infrastructure and data)
- Applications (e.g., (cloud-based) word processors)

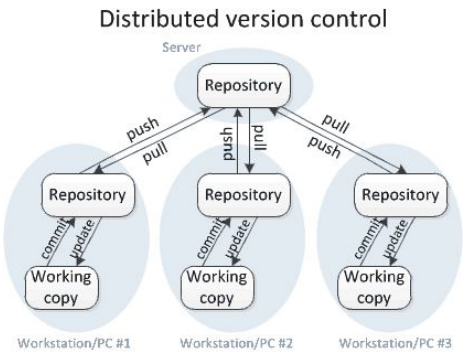
Centralized version control

- **One central repository.**
- All users **commit** their changes to a **central repository**.
- Each user has a working copy. As soon as they commit, the repository gets updated.
- Examples: SVN (Subversion), CVS.



Distributed version control

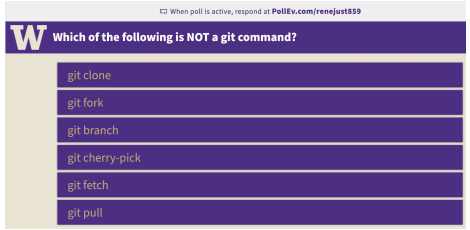
- **Multiple copies of a repository**.
- Each user **commits** to a **local** (private) repository.
- All committed changes remain local unless **pushed** to another repository.
- No external changes are visible unless **pulled** from another repository.
- Examples: Git, Hg (Mercurial).



Version control with Git
(aka the best thing since sliced bread)

- “I see Subversion as being the most pointless project ever started”
- " ‘what would CVS never ever do’-kind of approach”

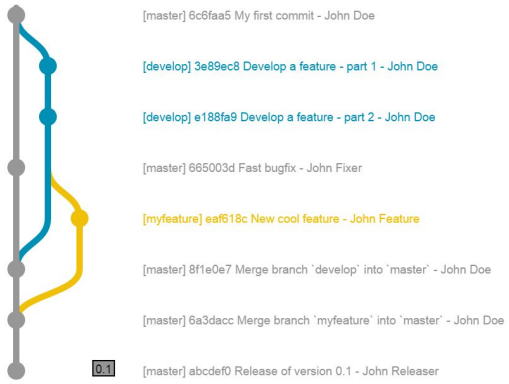
A little quiz



Branch vs. Clone vs. Fork

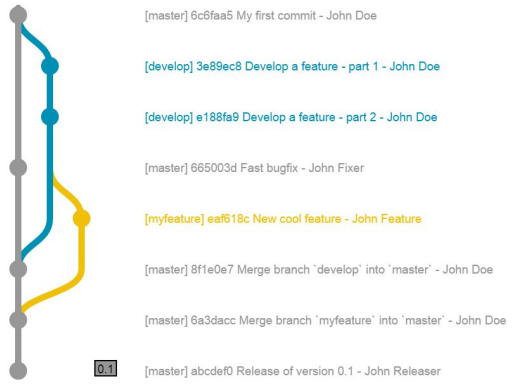
Branches

- One **main** development **branch** (**main**, master, trunk, etc.).
- Adding a new feature, fixing a bug, etc.: create a new **branch** -- a **parallel line of development**.
- **Lightweight** branching (**branch**).
- **Heavyweight** branching (**clone**).
- **Forking** (clone + metadata).

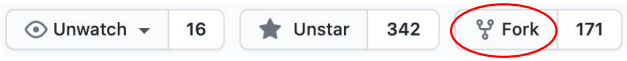


Branches

- One **main** development **branch** (**main**, master, trunk, etc.).
- Adding a new feature, fixing a bug, etc.: create a new **branch** -- a **parallel line of development**.
- **Lightweight** branching (**branch**).
- **Heavyweight** branching (**clone**).
- **Forking** (clone + metadata).

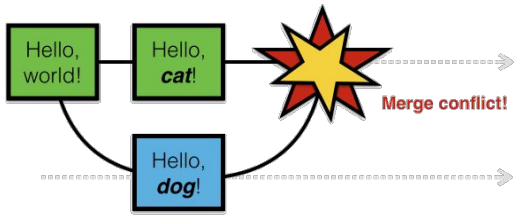


Branch and clone are common version control commands; forking is a concept used by GitHub etc.



Conflicts

Conflicts

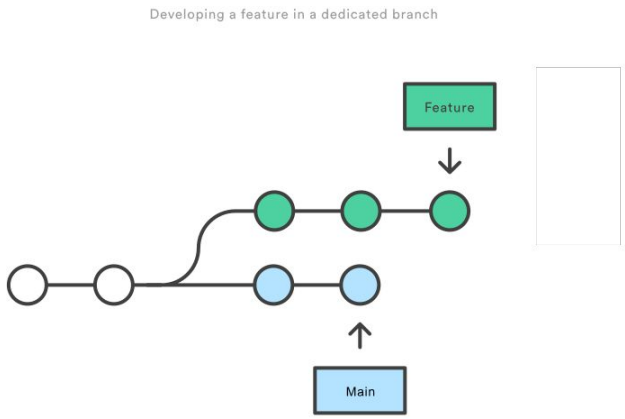


- **Conflicts** arise when two users **change the same line** of a file.
- When a conflict arises, the last committer needs to resolve it.

How to avoid merge conflicts?

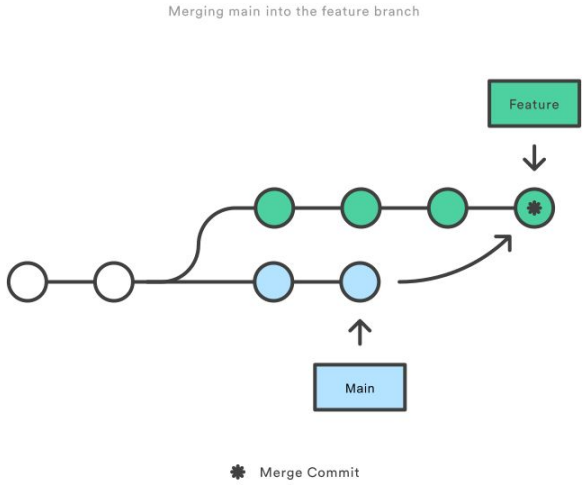
Merge vs. Rebase
(vs. Interactive Rebase)

Merge vs. Rebase



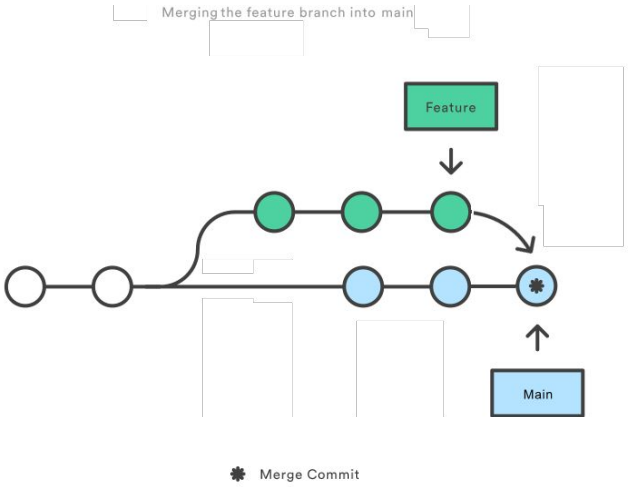
https://www.atlassian.com/git/tutorials/merging-vs-rebasing

Merge (integrating changes **from main**)



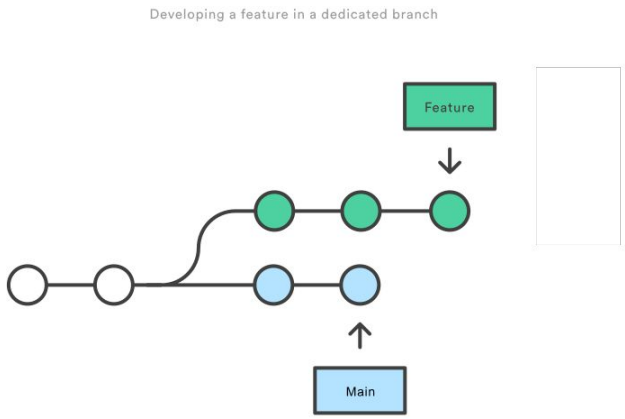
https://www.atlassian.com/git/tutorials/merging-vs-rebasing

Merge (integrating changes **into main**)



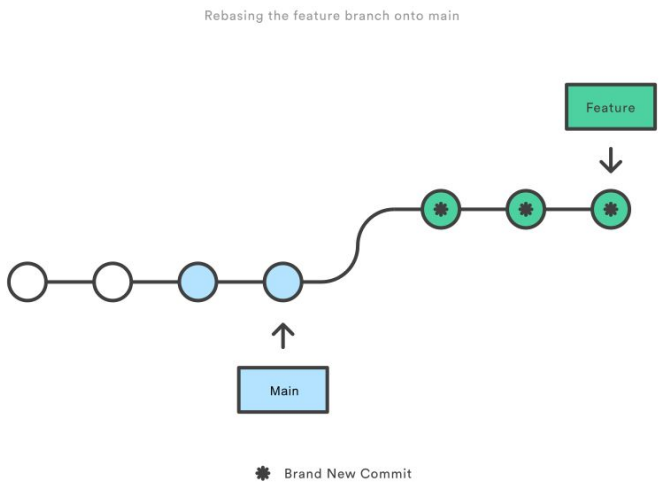
https://www.atlassian.com/git/tutorials/merging-vs-rebasing

Merge vs. Rebase



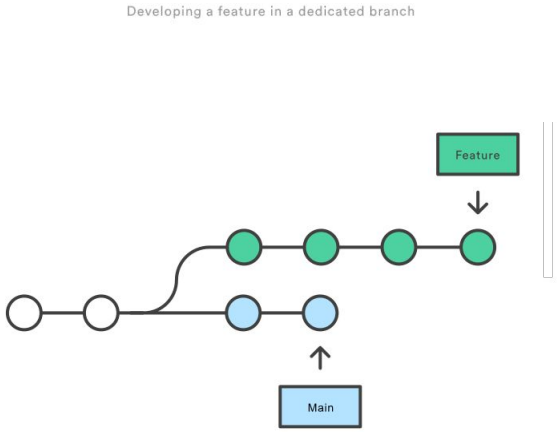
https://www.atlassian.com/git/tutorials/merging-vs-rebasing

Merge vs. **Rebase**



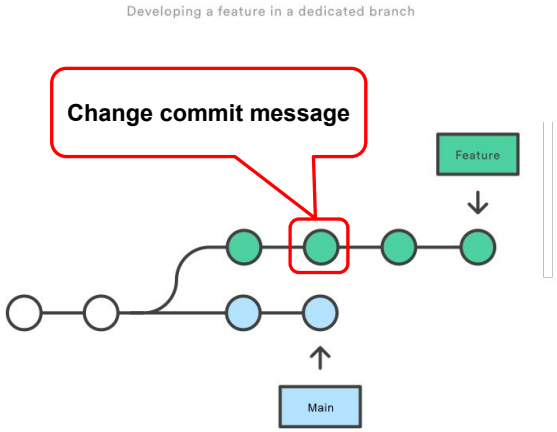
https://www.atlassian.com/git/tutorials/merging-vs-rebasing

Interactive Rebase



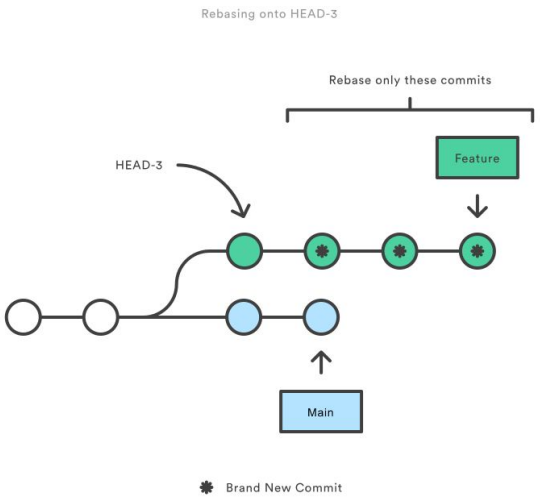
https://www.atlassian.com/git/tutorials/merging-vs-rebasing

Interactive Rebase (reword)



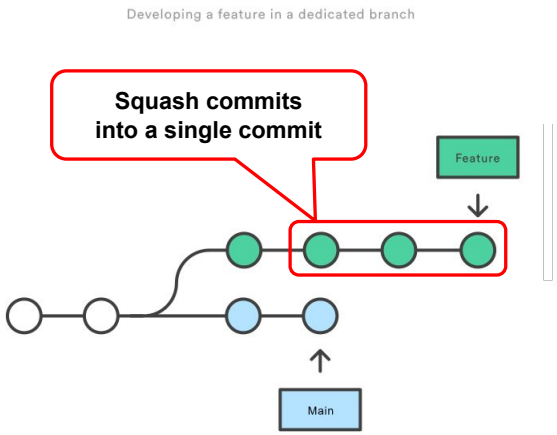
https://www.atlassian.com/git/tutorials/merging-vs-rebasing

Interactive Rebase (reword)



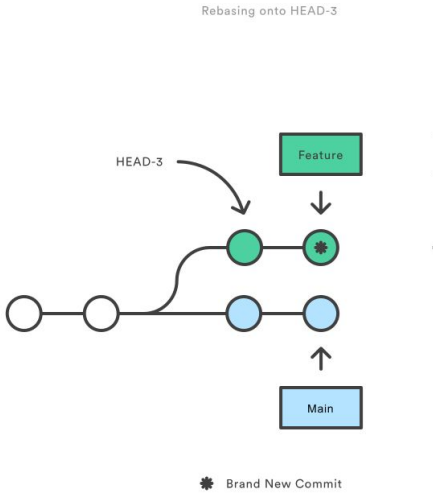
https://www.atlassian.com/git/tutorials/merging-vs-rebasing

Interactive Rebase (squash)



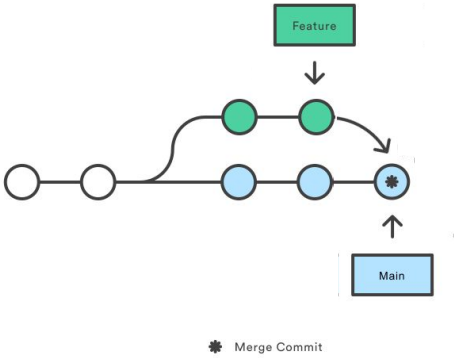
https://www.atlassian.com/git/tutorials/merging-vs-rebasing

Interactive Rebase (squash)



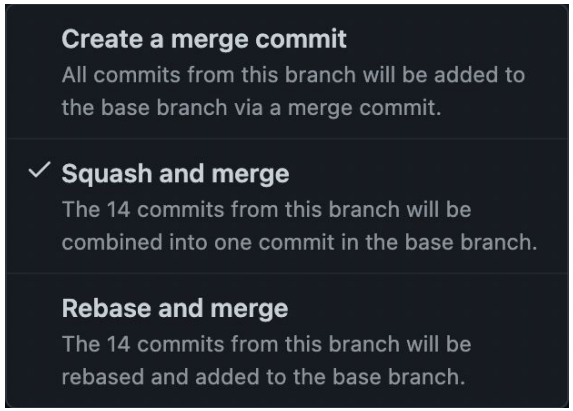
https://www.atlassian.com/git/tutorials/merging-vs-rebasing

Interactive Rebase (squash & merge)

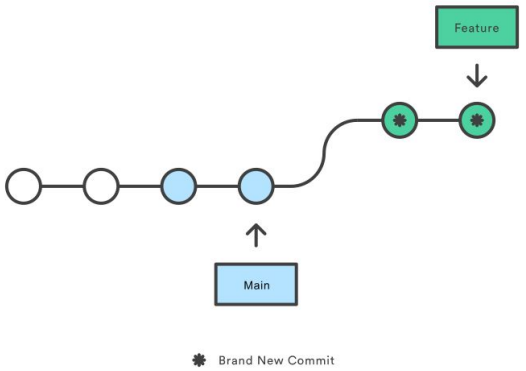


https://www.atlassian.com/git/tutorials/merging-vs-rebasing

Squash & merge on GitHub



Interactive Rebase (squash & rebase)



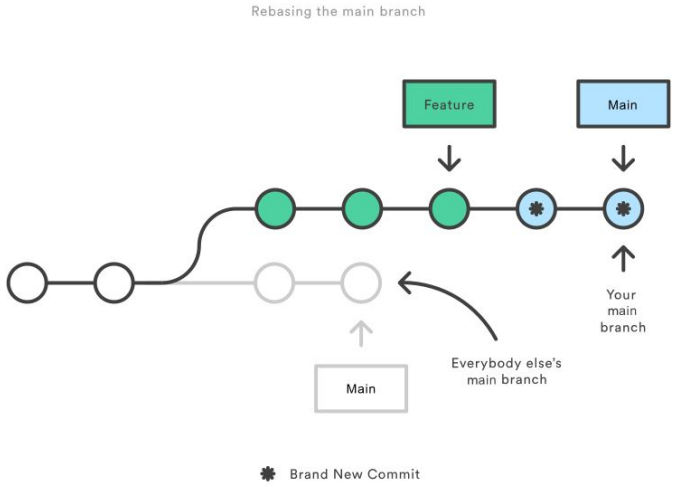
https://www.atlassian.com/git/tutorials/merging-vs-rebasing

Rebase: a powerful tool, but …

- Results in a sequential commit history.
- Interactive rebasing often used to squash commits.
- **Changes the commit history!**

Do not rebase public branches with a force-push!

Rebase: a powerful tool, but …



https://www.atlassian.com/git/tutorials/merging-vs-rebasing

Git concepts and terminology

Motivating Example: What is this Git command?

NAME

git-_____ - _____ file contents to the index

SYNOPSIS

git _____ [--dry-run | -n] [--force | -f] [--interactive | -i] [--patch | -p]

DESCRIPTION

This command updates the index using the current content found in the working tree, to prepare the content staged for the next commit. It typically _____s the current content of existing paths as a whole, but with some options it can also be used to _____ content with only part of the changes made to the working tree files applied, or remove paths that do not exist in the working tree anymore.

Motivating Example: What is this Git command?

NAME

git-add - Adds file contents to the index

SYNOPSIS

git add [--dry-run | -n] [--force | -f] [--interactive | -i] [--patch | -p]

DESCRIPTION

This command updates the index using the current content found in the working tree, to prepare the content staged for the next commit. It typically adds the current content of existing paths as a whole, but with some options it can also be used to add content with only part of the changes made to the working tree files applied, or remove paths that do not exist in the working tree anymore.

Git: concepts and terminology

SYNOPSIS

git-diff-index [-m] [--cached] [<common diff options>] <tree-ish> [<path>…]

DESCRIPTION

git-diff-index compares the content and mode of the blobs found in a tree object with the corresponding tracked files in the working tree, or with the corresponding paths in the index.

Git: concepts and terminology

SYNOPSIS
git-diff-index [-m] [--cached] [<common diff options>] <tree-ish> [<path>…]

DESCRIPTION
git-diff-index compares the content and mode of the blobs found in a tree object with the corresponding tracked files in the working tree, or with the corresponding paths in the index.

SYNOPSIS
git-allocate-remote *[--derive-head | --massage-link-head | --abduct-commit]*

DESCRIPTION
git-allocate-remote allocates various non-branched local remotes outside added logs, and the upstream to be packed can be supplied in several ways.

SYNOPSIS
git-resign-index *[--snap-file] [--direct-change]*

DESCRIPTION
git-resign-index resigns all non-stashed unstaged indices, and the --manipulate-submodule flag can be used to add a branch for the upstream that is counted by a temporary submodule.

Git: concepts and terminology

SYNOPSIS
git-diff-index [-m] [--cached] [<common diff options>] <tree-ish> [<path>…]

DESCRIPTION
git-diff-index compares the content and mode of the blobs found in a tree object with the corresponding tracked files in the working tree, or with the corresponding paths in the index.

SYNOPSIS
git-allocate-remote *[--derive-head | --massage-link-head*

DESCRIPTION
git-allocate-remote allocates
be packed can be

SYNOPSIS
git-resign-index

DESCRIPTION
git-resign-index resig
used to add a branch for the upstream that is counted by a temporary submodule.

Git: vocabulary

- **index**: staging area (located .git/index)
- **content**: git tracks **what is in a file, not the file itself**
- **tree**: git's representation of a file system
- **working tree**: tree representing the local working copy
- **staged**: ready to be committed
- **commit**: a snapshot of the working tree (a database entry)
- **ref**: pointer to a commit object
- **branch**: just a (special) ref; semantically: represents a line of dev
- **HEAD**: a ref pointing to the working tree

Git: concepts and terminology