



Outline

Planned topics for this lesson:

- Version (source) control
WHY WE NEED IT? HOW SHOULD WE USE IT?
- Types of version control
CENTRALISED AND DISTRIBUTED
- Version control software



- Lab: Fun with GitHub
GET YOUR STUDENT ACCOUNT, AND GIT "AWAY"

```

CodeFixService.cs*  X Git Repository - roslyn
Show Conflicts Only  Take Incoming  Take Current  Compare  Accept Merge  Show Word Diffs
24 Conflicts (23 Remaining) | 60 AutoMerged (Incoming: 58, Current: 0, Both: 2)

Incoming: master
56 .....private·readonly·ImmutableDictionary<LanguageKind,·Lazy<ISuppressionFixProvider>>·_suppressionPro
57 .....private·readonly·IEnumerable<Lazy<ILoggerService>>·_errorLoggers;
58
59 .....private·ImmutableDictionary<object,·FixAllProviderInfo>·_fixAllProviderMap;
60
61 .....[ImportingConstructor]
62 ✓ .....[SuppressMessage("RoslynDiagnosticsReliability",·"RS0033:Importing·constructor·should·be·[Obsole
63 .....public·CodeFixService(
64 ✓ .....IThreadingContext·threadingContext,
65 .....IDiagnosticAnalyzerService·service,
66 ✓ .....[ImportMany]·IEnumerable<Lazy<ILoggerService>>·loggers,
67 ✓ .....[ImportMany]·IEnumerable<Lazy<CodeFixProvider>>·fixProviders;

Current: DemoBranch
45 .....private·readonly·ImmutableDictionary<LanguageKind,·Lazy<ISuppressionFixProvider>>·_suppressionPro
46 .....private·readonly·IEnumerable<Lazy<ILoggerService>>·_errorLoggers;
47
48 .....private·ImmutableDictionary<object,·FixAllProviderInfo>·_fixAllProviderMap;
49
50 .....[ImportingConstructor]
51 .....public·CodeFixService(
52 .....IDiagnosticAnalyzerService·service,
53 .....[ImportMany]·IEnumerable<Lazy<ILoggerService>>·loggers,
54 .....[ImportMany]·IEnumerable<Lazy<CodeFixProvider>>·fixProviders;

Result: src/EditorFeatures/Core/Implementation/CodeFixes/CodeFixService.cs
56 .....private·readonly·ImmutableDictionary<LanguageKind,·Lazy<ISuppressionFixProvider>>·_suppressionPro
57 .....private·readonly·IEnumerable<Lazy<ILoggerService>>·_errorLoggers;
58
59 .....private·ImmutableDictionary<object,·FixAllProviderInfo>·_fixAllProviderMap;
60
61 .....[ImportingConstructor]
62 .....[SuppressMessage("RoslynDiagnosticsReliability",·"RS0033:Importing·constructor·should·be·[Obsole
63 .....public·CodeFixService(
64 .....IThreadingContext·threadingContext,
65 .....IDiagnosticAnalyzerService·service,
66 .....[ImportMany]·IEnumerable<Lazy<ILoggerService>>·loggers,
67 .....[ImportMany]·IEnumerable<Lazy<CodeFixProvider>>·fixProviders;
    
```



What is version control?

- A system that records changes to a file or set of files over time, allowing you to recall or access specific versions later.
- Also known as revision control or source control
- Keep track of changes, by whom and when
- Fundamental tools for developing software projects



Version control, also known as source control or revision control, is a system that helps manage changes to files, documents, or any set of information over time. It is widely used in software development but can be applied to other contexts as well. The primary purpose of version control is to track modifications to a project's files, allowing multiple people to collaborate on a project, providing a history of changes, and enabling easy retrieval of previous versions.

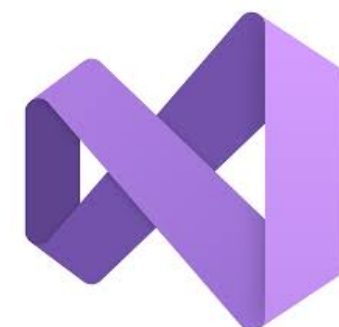




What is version control software?



SOME OF THE POPULAR VERSION CONTROL SOFTWARES [L - R] CVS, SUBVERSION, git (GitHub), Team Foundation Server, Mercurial



BUILT IN VERSION CONTROL AVAILABLE IN A VARIETY OF SOFTWARE PRODUCT



Why do we need it?

- Backup software source
 - Rollback to previous version
- Keeping a record of who did what and when
 - Know who to praise and who to fire
- Collaborating with the team
 - Know who to praise and who to fire
- Troubleshooting
 - Analyse the change history to figure out what cause the problem
- Statistic
 - Find out who is the most productive





What should you check in?

- Everything that influence the build
CONFIGURATION FILES, FILE ENCODINGS, BINARY SETTINGS, ETC.
- Everything to setup the project from a clean checkout / fork
SOURCE CODES, DOCUMENTATIONS, MANUALS, IMAGE FILES, DATASETS, ETC.
- What about IDE “Junk” files?

The screenshot shows the GitHub repository for twbs/bootstrap. The repository is public and has 165k stars, 78.9k forks, and 6.8k watchers. The main branch is selected, and the repository contains 96 branches and 87 tags. The commit history is displayed, showing a list of commits with their titles, commit hashes, and dates. The most recent commit is by dependabot[bot] titled 'Build(deps-dev): Bump h...' with a commit hash of 413894d, committed last week. The repository also has a README, MIT license, Code of conduct, Security policy, and Activity. The latest release is v5.3.1, released on Jul 26.

Commit Hash	Commit Title	Commit Date
413894d	Build(deps-dev): Bump h...	last week
...
...	Build(deps): Bump streetsidesoftware/cspell-acti...	3 weeks ago
...	Improve change-version script (#38983)	3 weeks ago
...	Release v5.3.1 (#38956)	2 months ago
...	Dropdown: reuse variable (#39046)	3 weeks ago
...	Slightly improve PNG files compression (#38438)	5 months ago
...	Use box-shadow CSS vars instead of Sass vars in...	last month
...	Corrected a grammer error in by adding the word...	3 weeks ago
...	Update Babel config (#31011)	3 years ago
...	.browserslistrc: remove Android and make Safari/i...	2 years ago
...	Fix bundlwatch values (#38954)	2 months ago
...	Fix reference to twbs/examples/icons-font + fine-...	6 months ago
...	Trim trailing whitespace from markdown files (#2...	4 years ago
...	Tweak and re-organize ESLint config (#38369)	6 months ago
...	Convert build scripts to ESM (#38984)	last month
...	Update .gitattributes (#30934)	3 years ago



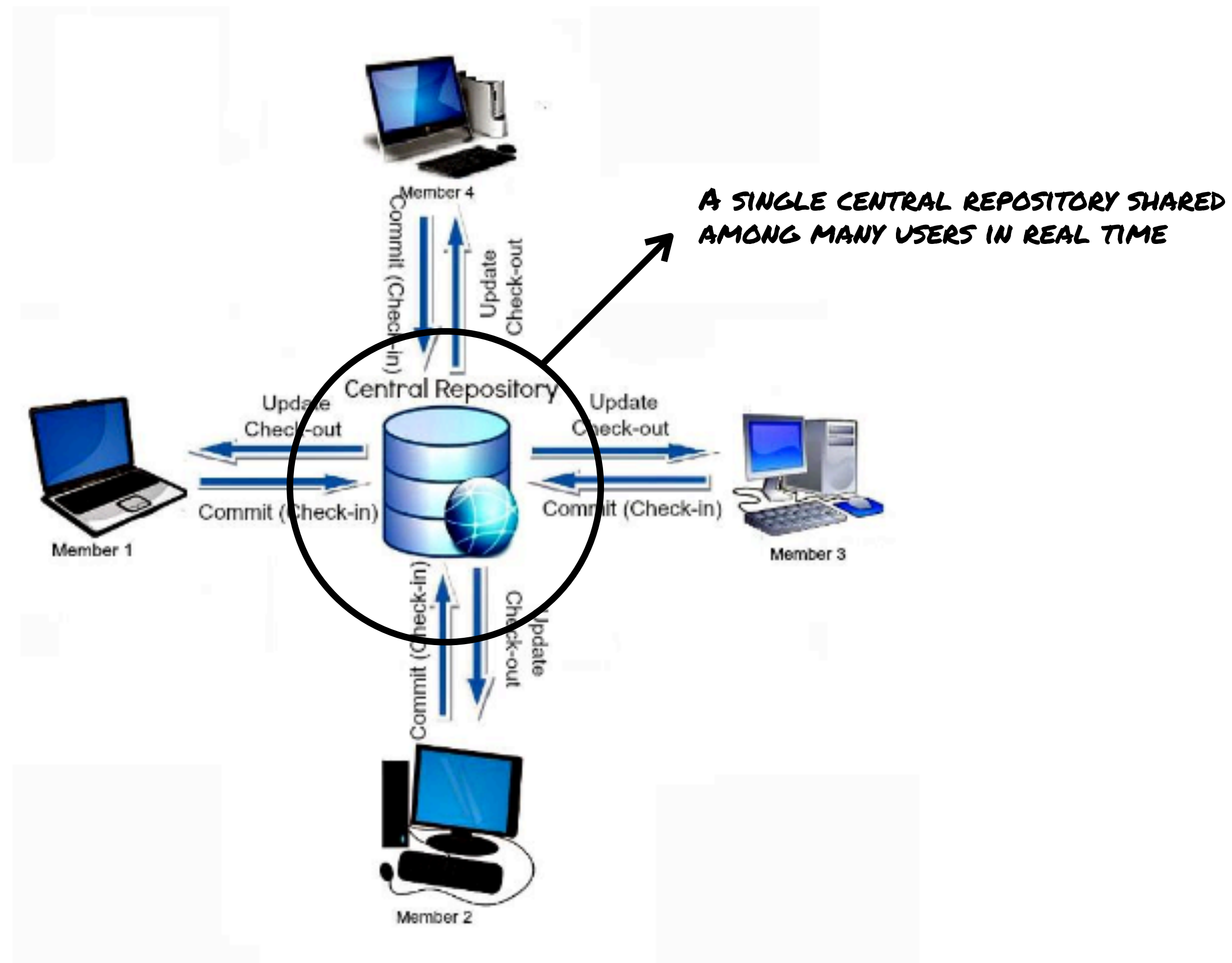
What should you **NOT** check in?

- Binaries
JAR FILES, OR ANY OTHER "BUILD" FILES FROM THE SOURCE
- Intermediate files from build / compilation
.PYC (PYTHON) OR .O (FOR C), ETC
- Files which contain "absolute path"
- Personal Preferences / Settings
SPECIFIC SETTING ONLY FOR YOUR MACHINE



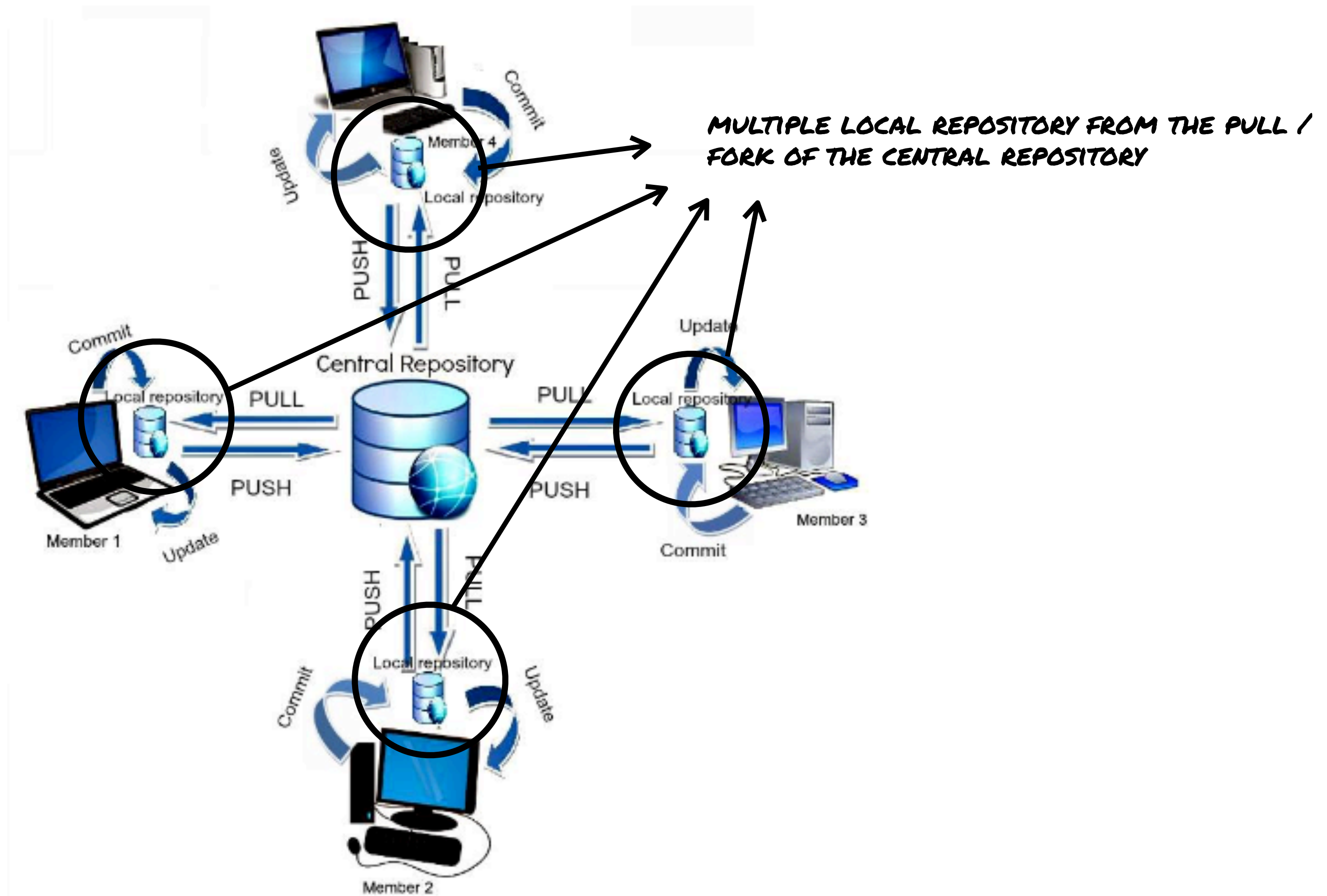


Centralised Version Control System





Distributed Version Control System





Subversion



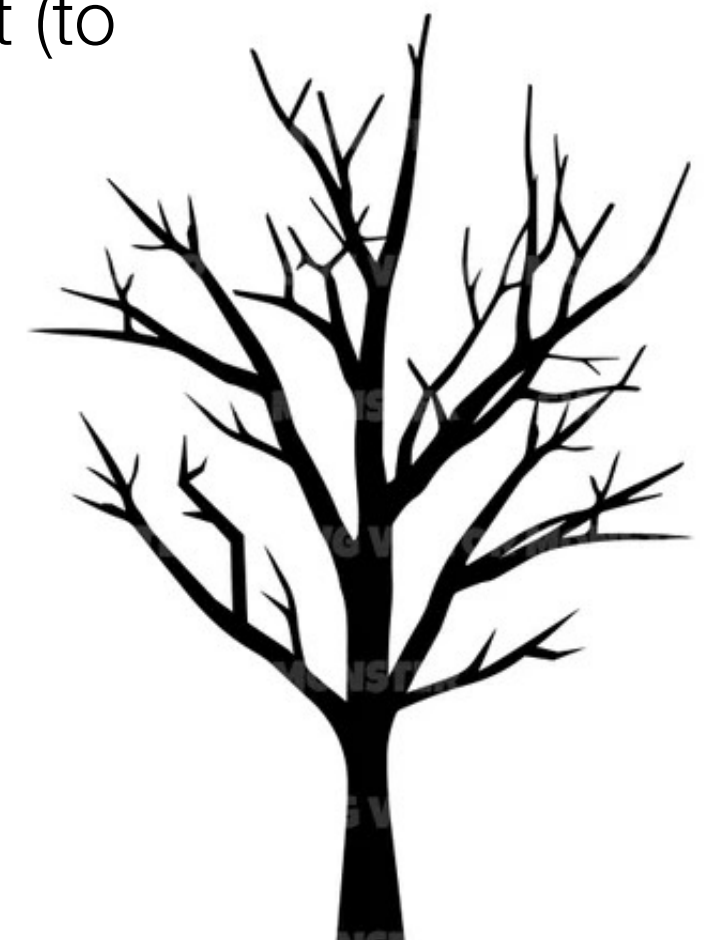
- Code centralised in a repository
- Check out a working copy onto your machine
 - General, you don't have the entire repository checked out in subversion, you only check out a specific branch
- Make changes to it
- Changes committed back to a central repository - “normally” with useful comments
- Maintain a change log (Who, When, What)



Subversion



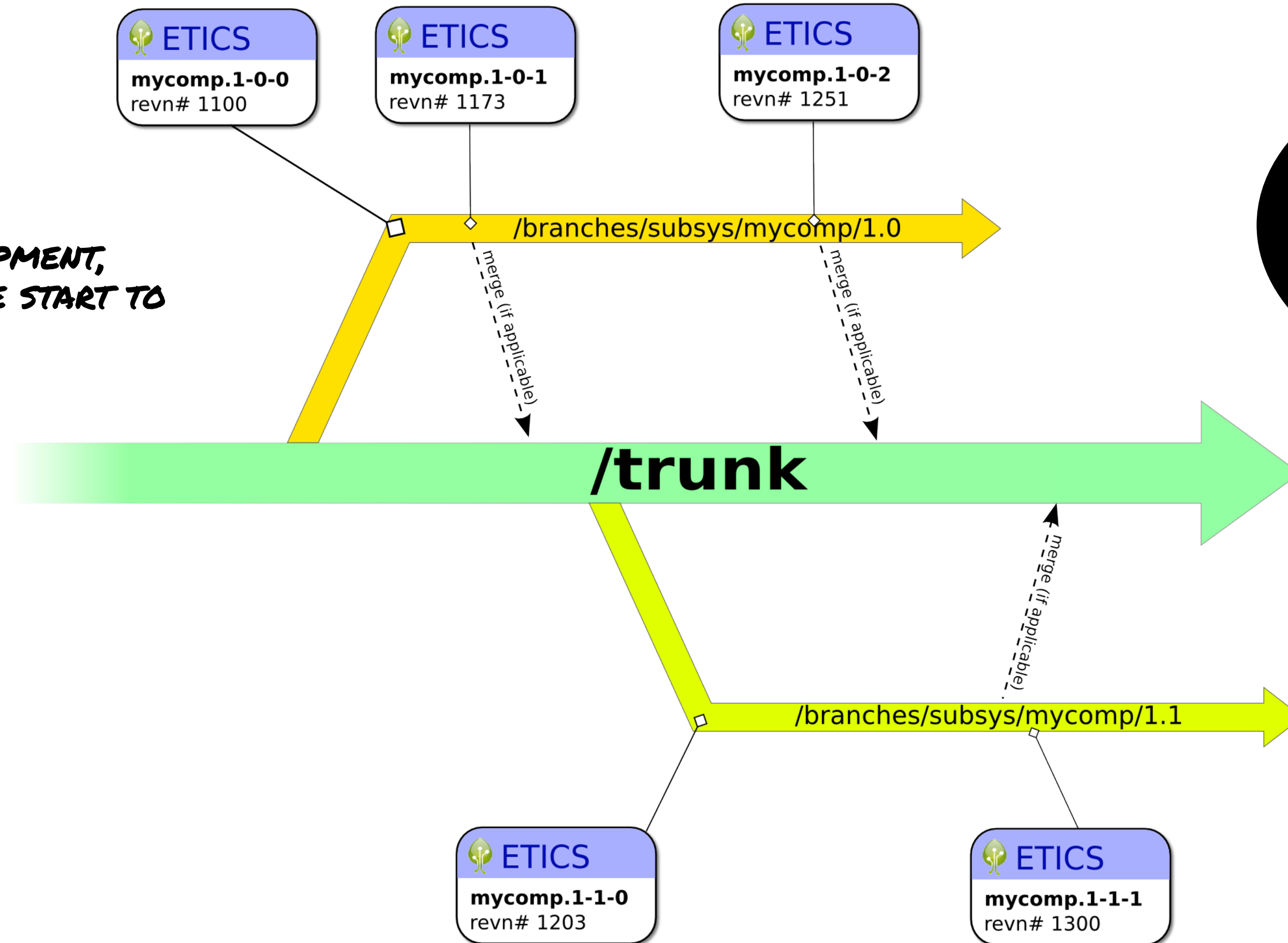
- Subversion is like a tree
- A tree has a trunk and some branches
 - Branches grow from the trunk, and thinner branches grow from thicker branches
- If the trunk is sick, so are the branches, and eventually the whole tree can die
- If a branch is sick, you can cut it, another one may grow
- If a branch grows too much, it may become too heavy for the trunk, and the tree might fall down
- When you feel your tree, your trunk, or a branch is nice looking, you can take a picture of it (to remember how nice it was that day)





Subversion

Trunk:
MAIN BODY OF DEVELOPMENT,
ORIGINATING FROM THE START TO
THE FINISH

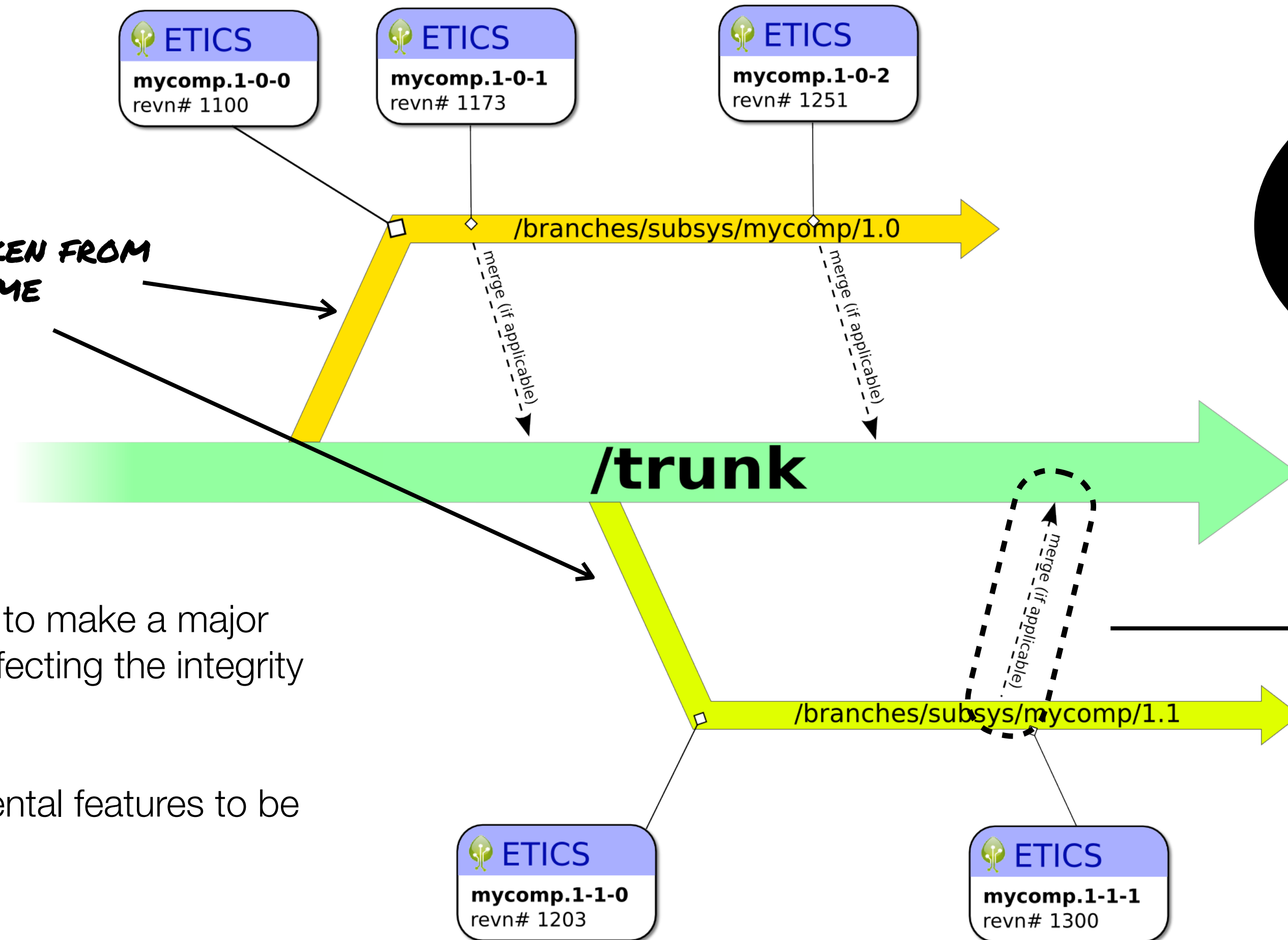


SHOULD I WORK DIRECTLY WITH THE TRUNK?



Subversion

Branch:
COPY OF THE CODE, TAKEN FROM
A SPECIFIC POINT IN TIME



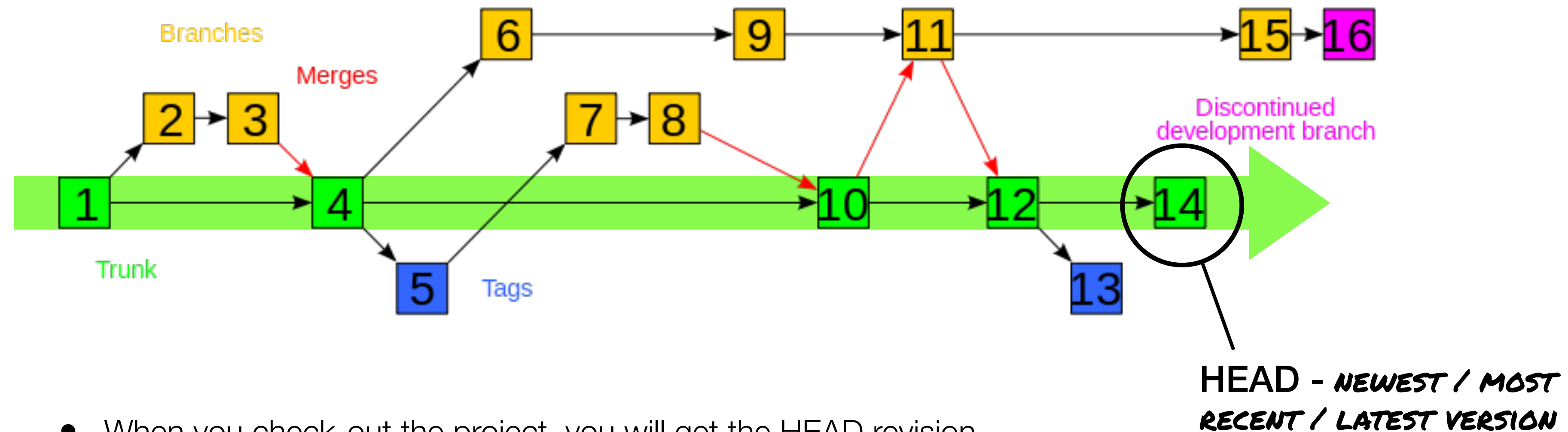
SHOULD I WORK
DIRECTLY WITH THE
TRUNK?
NO

- Allows a developer to make a major changes without affecting the integrity of the trunk
- Allows for experimental features to be tried and tested

If everything works as planned,
then merge the branch back
into the trunk



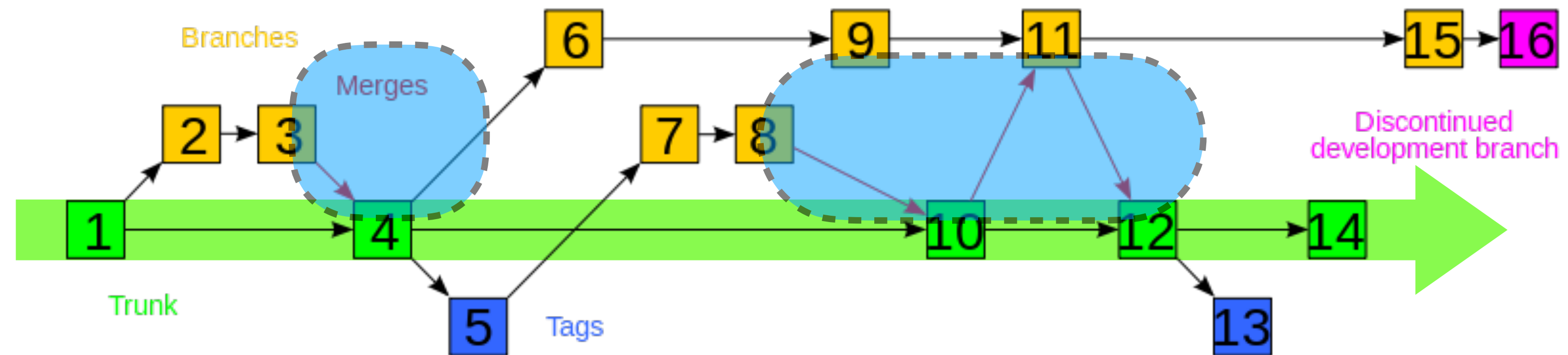
Subversion



- When you check-out the project, you will get the HEAD revision
- When you invoke the command `svn update`, you are updating your local copy to the HEAD version as well.



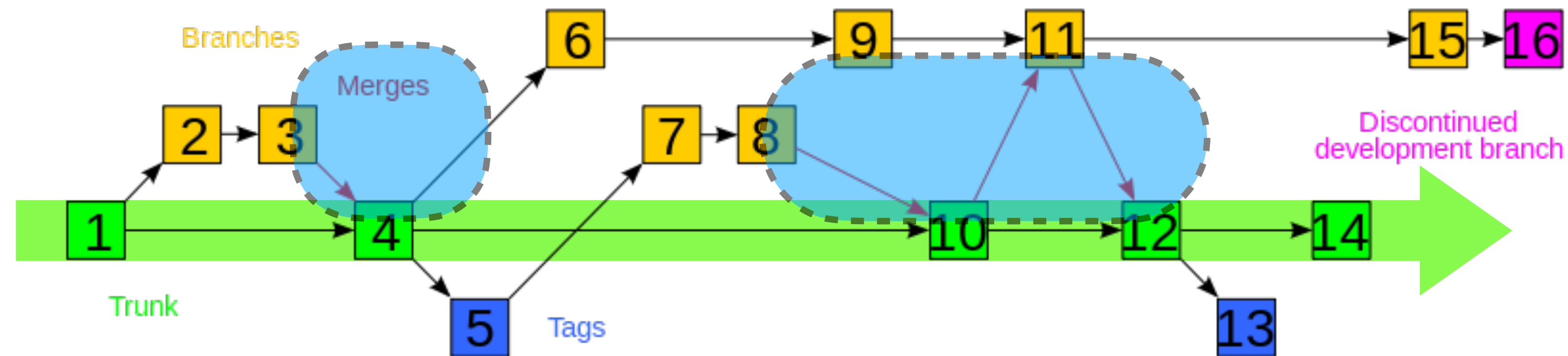
Subversion



- Branches should be eventually merged back into the trunk with `svn commit`
- The trunk must build afterwards



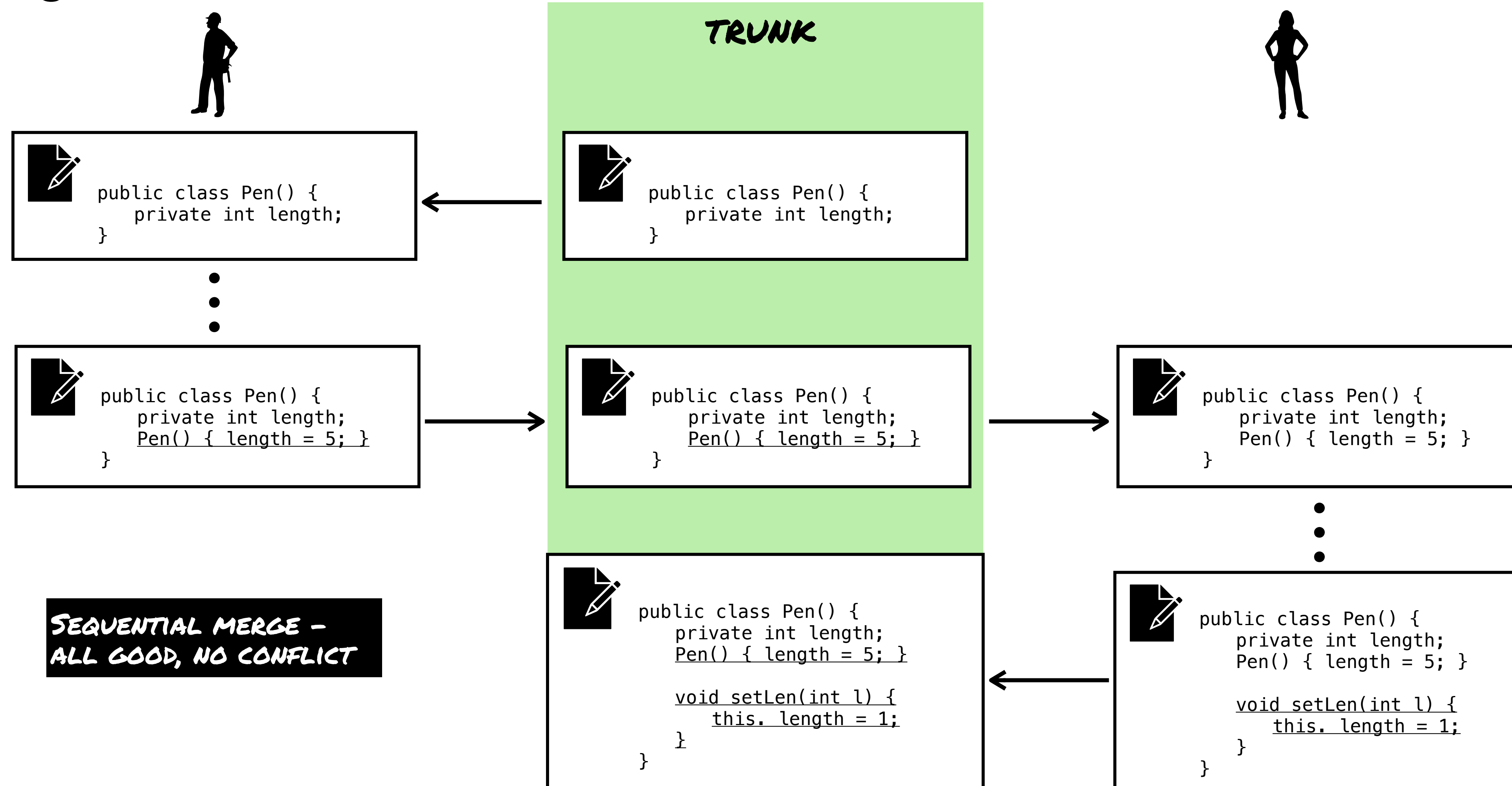
Subversion



- Commit is a process of storing changes from private workplace to central server. After commit, changes are made available to all the team.
- Other developers can retrieve these changes by updating their working copy.
- **Commit is an atomic operation.** Either the whole commit succeeds or is rolled back. Users never see half finished commit.

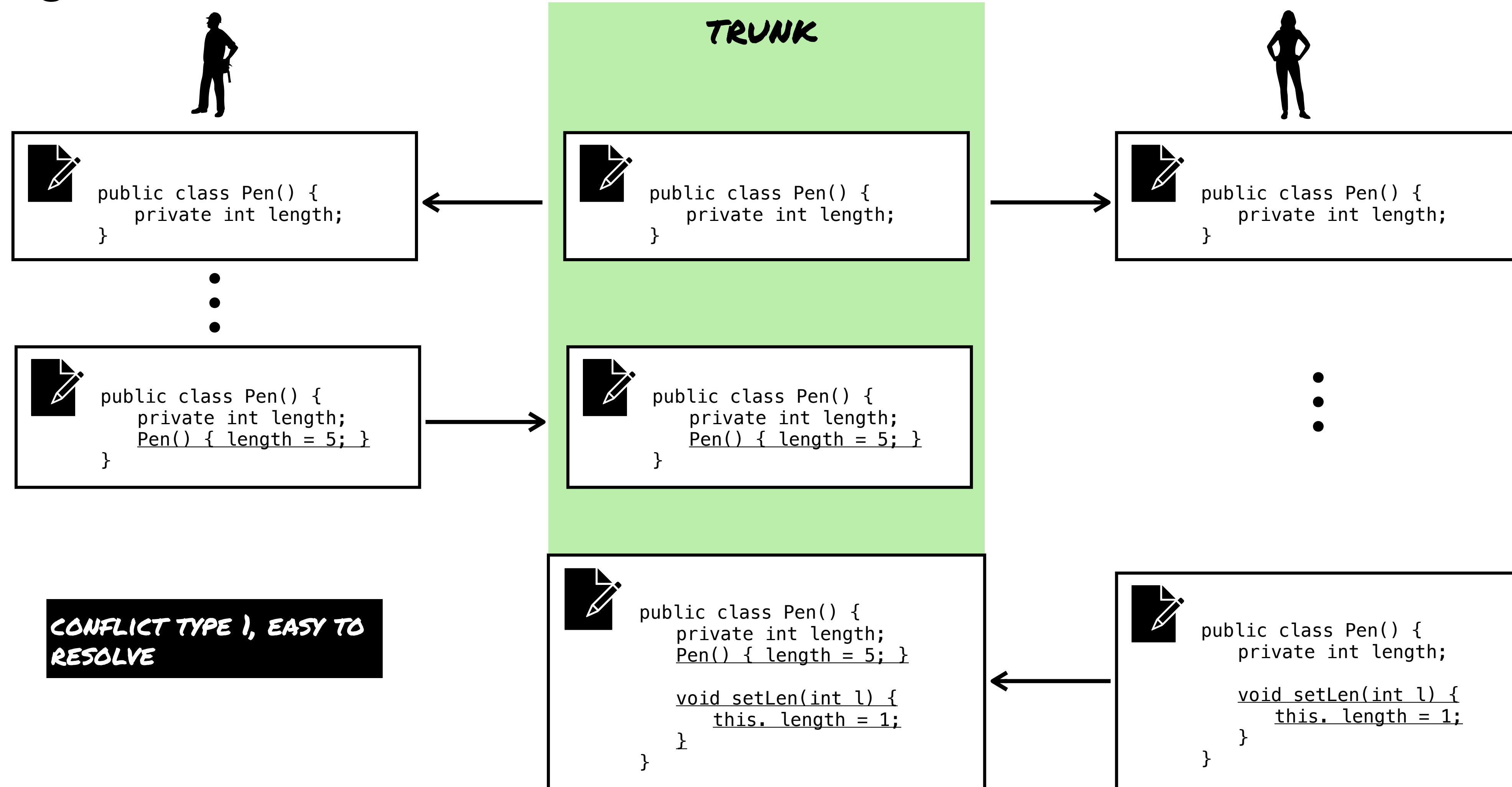


Merge Conflicts



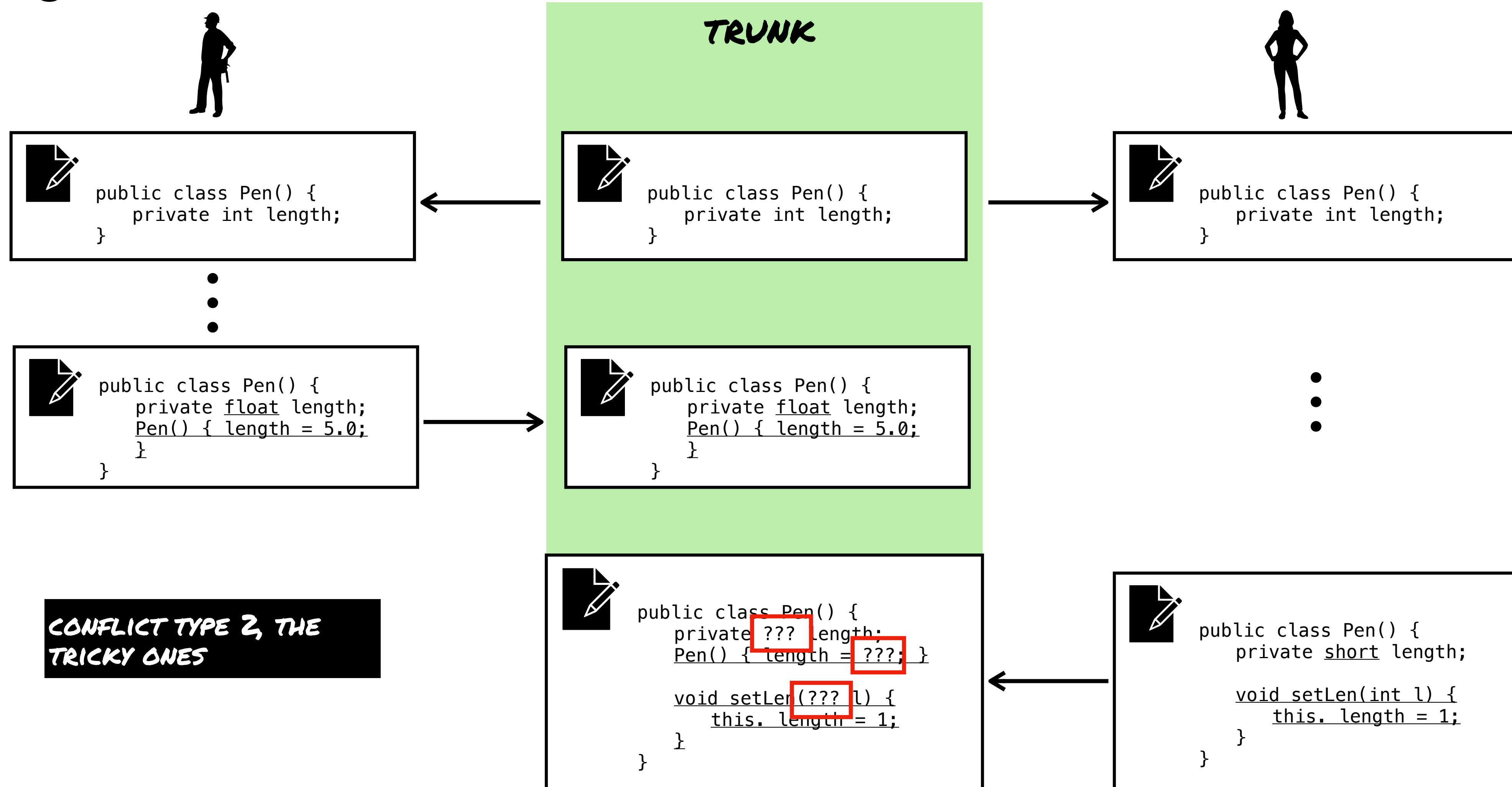


Merge Conflicts





Merge Conflicts



CONFLICT TYPE 2, THE TRICKY ONES



Git



Git is like a Swiss Army knife for version control, empowering developers with its distributed nature and powerful branching capabilities. It's a game-changer in collaborative coding. SVN, on the other hand, offers simplicity and stability, like a reliable old friend. Choose wisely based on your project's needs and team dynamics.

Linus Torvalds



Git



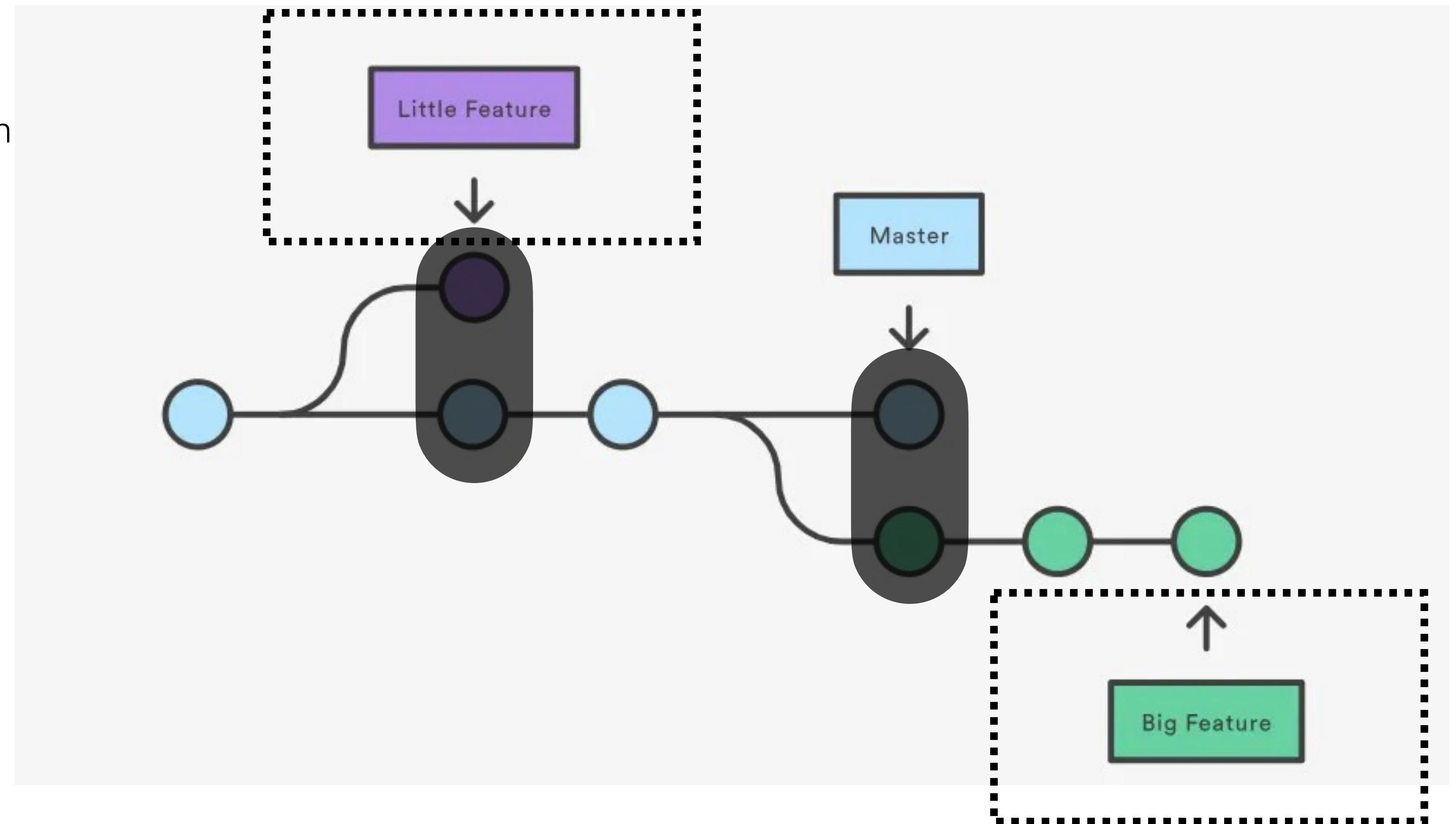
How was it created?

- Torvalds wasn't using any version control for Linux Kernel circa 1991 - 2002
- Changes were passed around as patches and archived files
- In 2002, they began using BitKeeper for managing the source for the Linux Kernel
- In 2005 the relationship broke down and BitKeeper revoked their license
- In no time April 2005, we were blessed with Git



Feature Branch Workflow

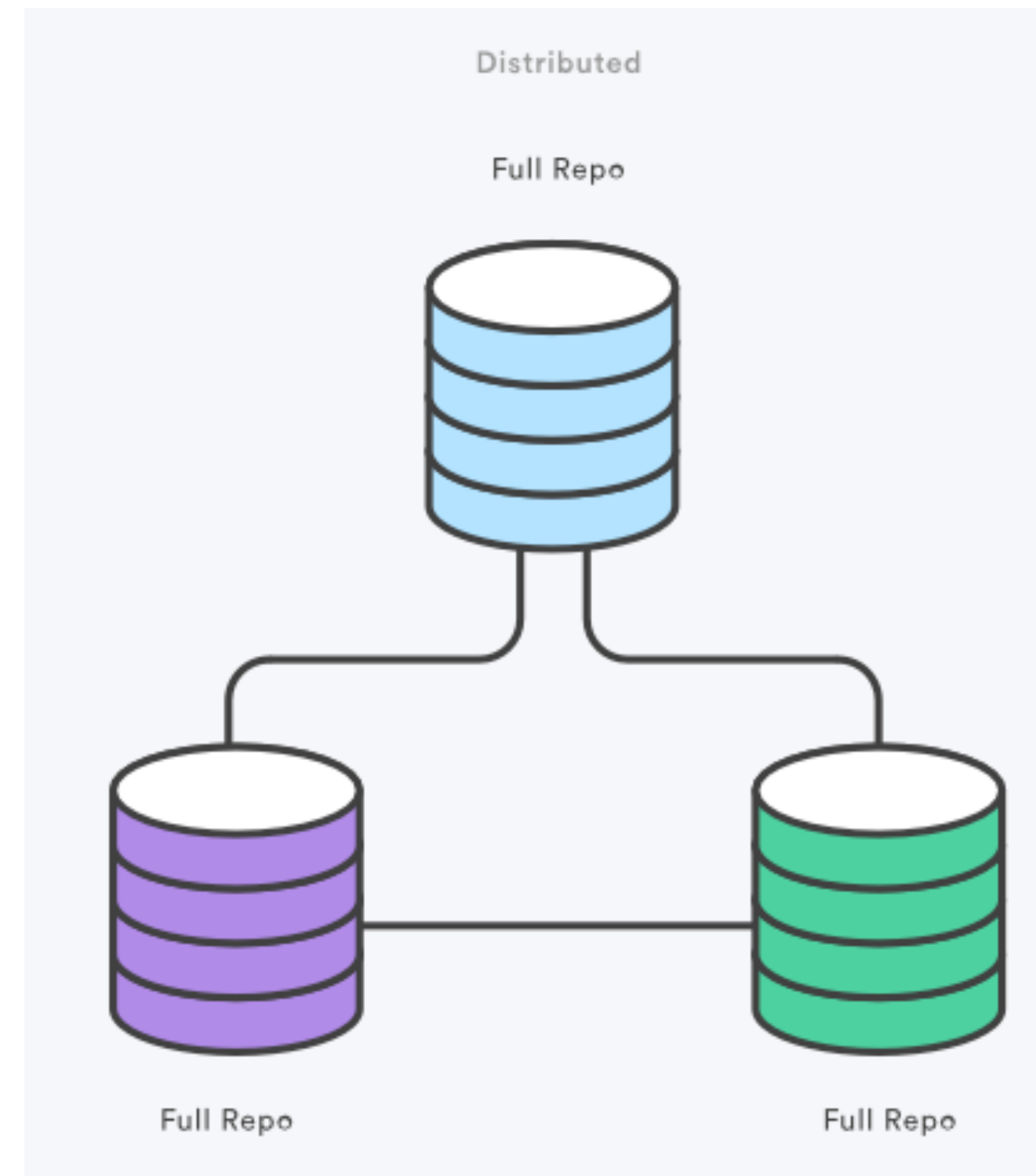
- Git encourages branching for every feature - regardless of the size, a branch can easily be created
- After successful completion of the new feature, branch is merged into trunk





Distributed Development

- Each developer gets their local repository
- Git is extremely fast, you don't need a network connection to commit changes (push), inspect previous version and perform diffs
- If someone breaks the production trunk / branch, you can continue working



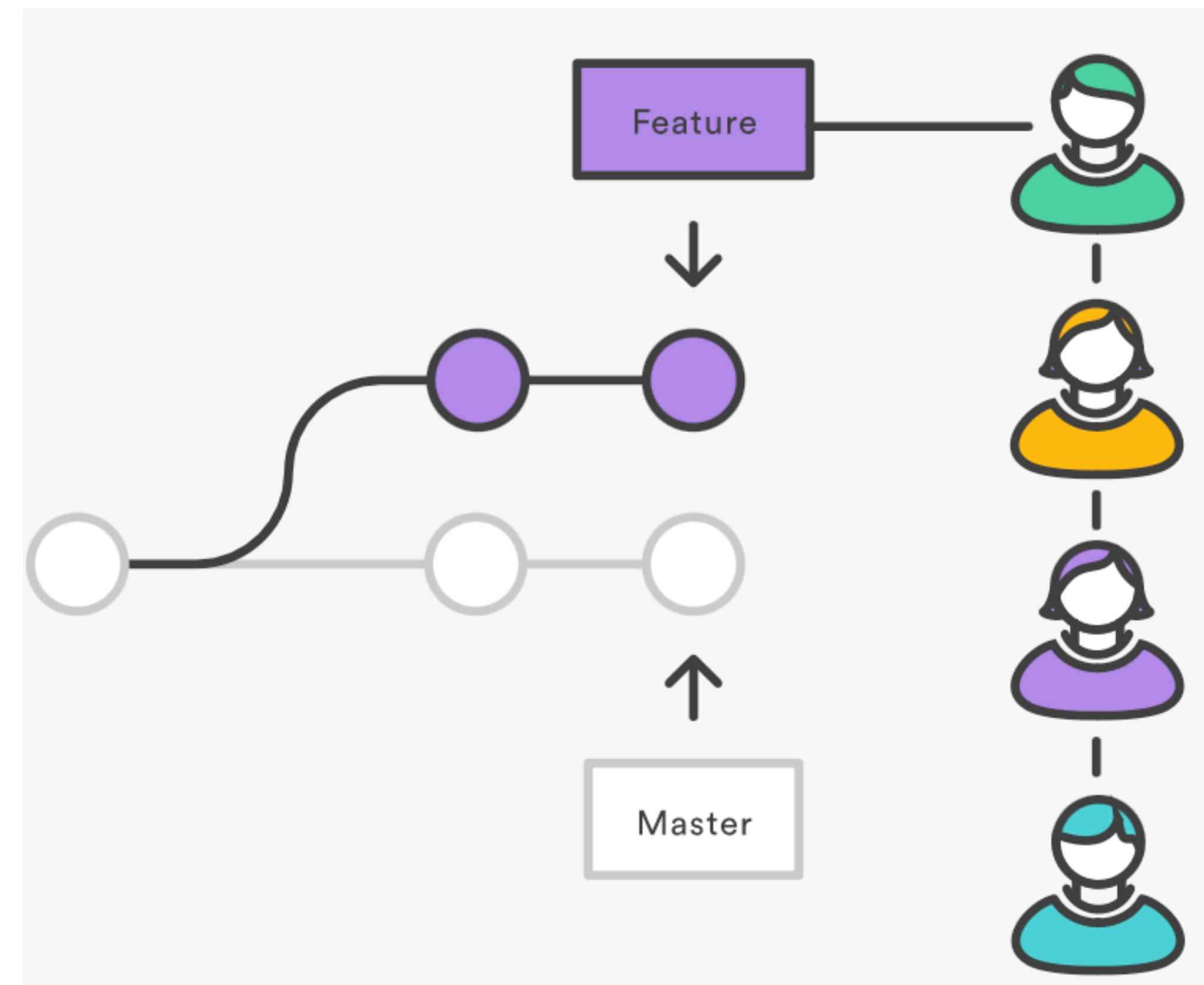


Pull Requests

- A pull request is where you ask another developer to merge your feature into their repository



- Project leads can keep track of changes
- Project leads can merge it with their repository





Git - Demo

The screenshot shows the Git website's 'Downloads' page. The header includes the Git logo and the tagline '--distributed-is-the-new-centralized'. A search bar is located in the top right. The left sidebar contains navigation links for 'About', 'Documentation', 'Downloads', 'GUI Clients', 'Logos', and 'Community'. The main content area features a 'Downloads' section with buttons for 'macOS', 'Windows', and 'Linux/Unix'. A large image of a Mac monitor displays the 'Latest source Release 2.42.0' with a 'Download for Mac' button. Below this, there are sections for 'GUI Clients' and 'Logos'. At the bottom, the 'Git via Git' section provides a terminal command to clone the repository.

Downloads

Latest source Release
2.42.0
Release Notes (2023-08-21)
Download for Mac

Older releases are available and the Git source repository is on GitHub.

GUI Clients
Git comes with built-in GUI tools (**git-gui**, **gitk**), but there are several third-party tools for users looking for a platform-specific experience.
[View GUI Clients →](#)

Logos
Various Git logos in PNG (bitmap) and EPS (vector) formats are available for use in online and print projects.
[View Logos →](#)

Git via Git
If you already have Git installed, you can get the latest development version via Git itself:

```
git clone https://github.com/git/git
```


You can also always browse the current contents of the git repository using the [web interface](#).

1 GET GIT ON YOUR MACHINE - GO HERE:

<https://git-scm.com/downloads>

<https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>

2 USE DEFAULT SETTING FOR NOW - YOU CAN RECONFIGURE LATER

**3 ON YOUR TERMINAL / PROMPT, RUN
git help git**

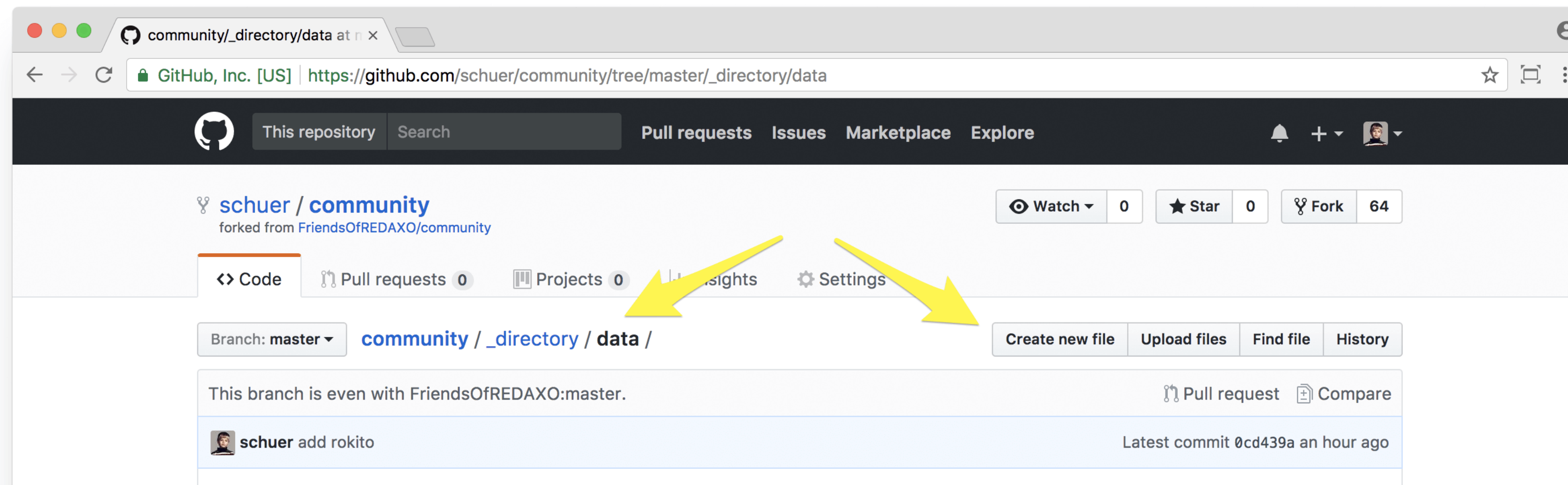
4 CREATE REPOSITORY, COMMIT, AND DO SOME STUFFS



GitHub



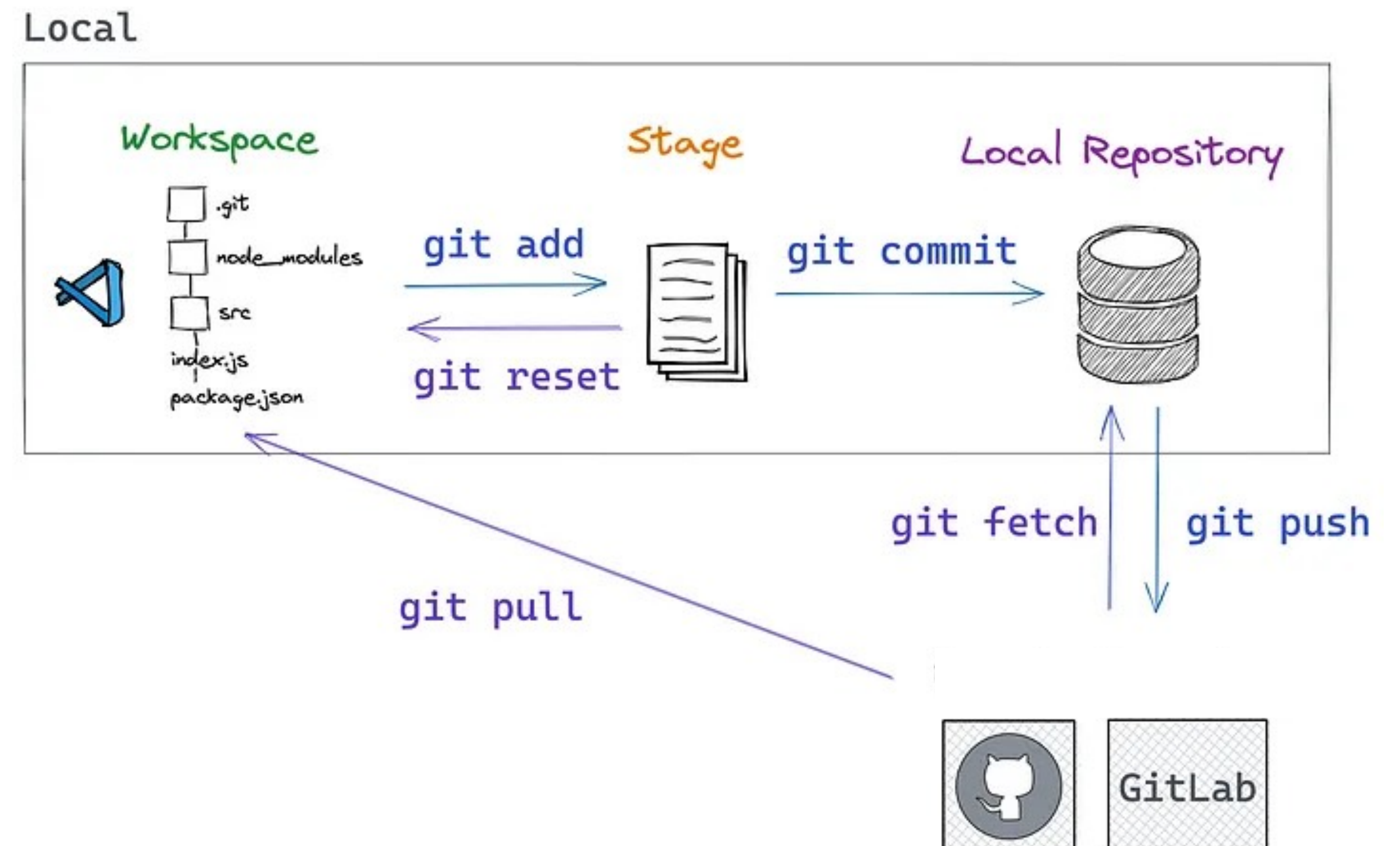
- GitHub is a web based hosted service for Git repositories.
- Git allows you to host remote Git repositories and has a wealth of community based service that makes it ideal for open source projects
- It is a publishing tool, a version control system and a collaboration platform





GitHub - Commands

- `git pull`
 - pull the latest version from the repository you clones
 - synchronise will all commits in the repository
- `git fetch` and `git merge`
 - pull is actually a combination of the two, and you can run these individually
- `git push`
 - once you have committed changes locally, you must push them to a remote repository
- `git clone`
 - many project don't require you to create your own repository, instead you clone it from a remote location





Pull requests

- If you make change to the repository, you can create a pull request
- Everyone can review the code and decide whether or not it should be included in the master branch
- It's a forum for discussing the changes.

The screenshot shows a GitHub pull request interface. At the top, it says "Sending a pull request #248" and "cameronmcefee wants to merge 1 commit into octocat:master from cameronmcefee:master". Below this, there are statistics: "Conversation 5", "Commits 1", and "Files Changed 1". The main content area shows a conversation:

- cameronmcefee** commented 2 years ago: "I made some changes. Please review."
- cameronmcefee** added a commit 2 years ago: "Made some changes for a pull request" (commit hash: a4610fa)
- octocat** commented 2 years ago: "Awesome, thanks!"
- cameronmcefee** commented 2 years ago: "Why yes, of course."

At the bottom, it says "cameronmcefee closed the pull request 2 years ago". On the right side, there are options for "Edit", "New Issue", "Labels", "Notifications", and "Subscribe".