CE/CS/SE 3354
Software Engineering

Software Repositories and Version Control Systems

Courtesy of Andrian Marcus and Juan Manuel Florez

Software repository hosting services

- Sometimes known just as *software repositories*
- Can be used to host most software development artifacts:
  - Source code
  - Bug reports
  - Documentation
- Plenty of free and paid alternatives

Alternatives

- GitHub
- Bitbucket
- SourceForge
- CodePlex
- GNU Savannah
- Launchpad

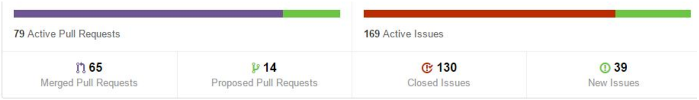
More at http://en.wikipedia.org/wiki/Comparison_of_open-source_software_hosting_facilities

GitHub

- A very solid hosting alternative for open-source projects
- Unlimited free repositories per user
- Ability to *watch* repositories - your dashboard page is like a Facebook wall with recent updates
- Plenty of graphs to keep track of a project's progress
- A focus on collaboration

GitHub Graphs

- Give an overview of the project status
- Pulse
- Contributors
- Commits
- Code frequency
- Punch card
- Network



79 Active Pull Requests

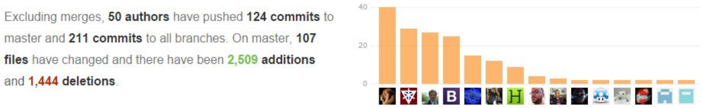
169 Active Issues

65 Merged Pull Requests

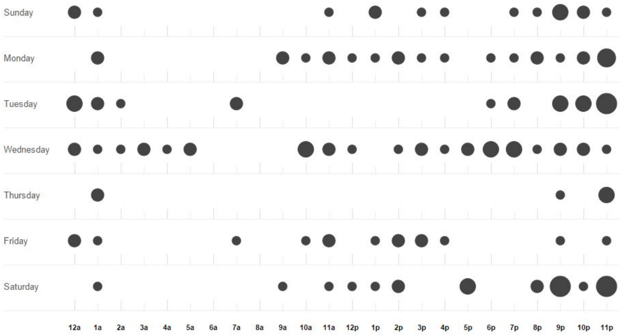
14 Proposed Pull Requests

130 Closed Issues

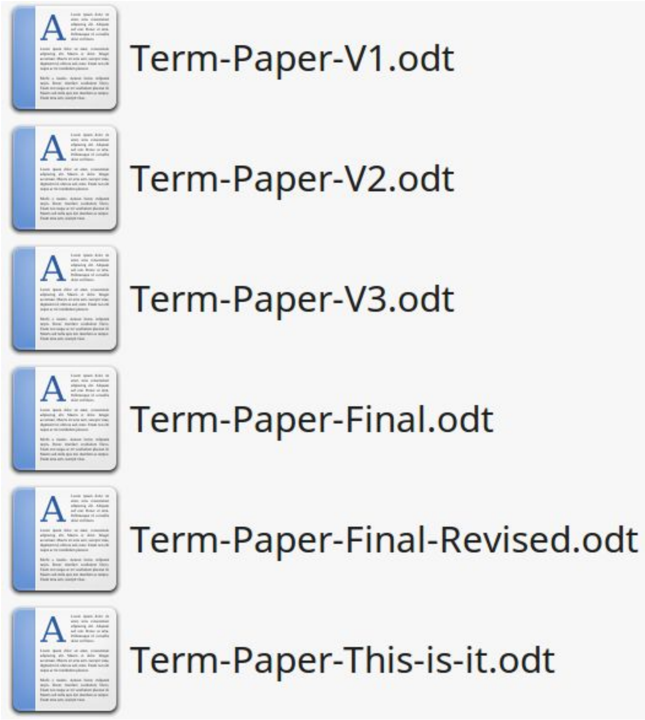
39 New Issues



Excluding merges, **50 authors** have pushed **124 commits** to master and **211 commits** to all branches. On master, **107 files** have changed and there have been **2,509 additions** and **1,444 deletions**.



Has this ever happened to you?



Term-Paper-V1.odt

Term-Paper-V2.odt

Term-Paper-V3.odt

Term-Paper-Final.odt

Term-Paper-Final-Revised.odt

Term-Paper-This-is-it.odt

- Which one is the final version?
- What are the differences between versions?
- When were the changes made and by whom?

We have the technology!

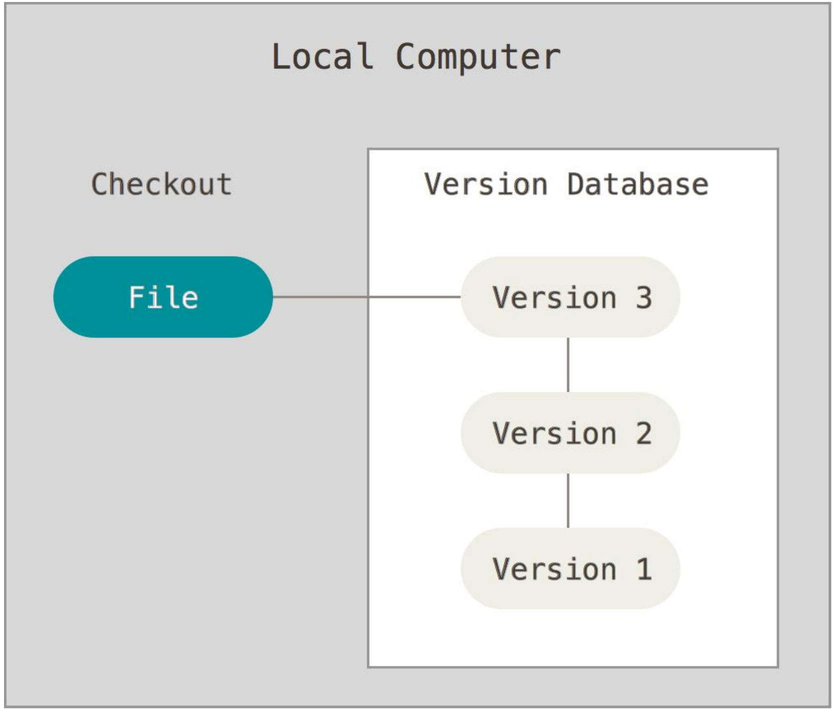
- Version control systems allow to easily manage changes to a set of files (usually source code)
- Generic term!
- Usually refers to *software* version control

How do version control systems work?

They keep a *version history* for a group of files.

They also store:

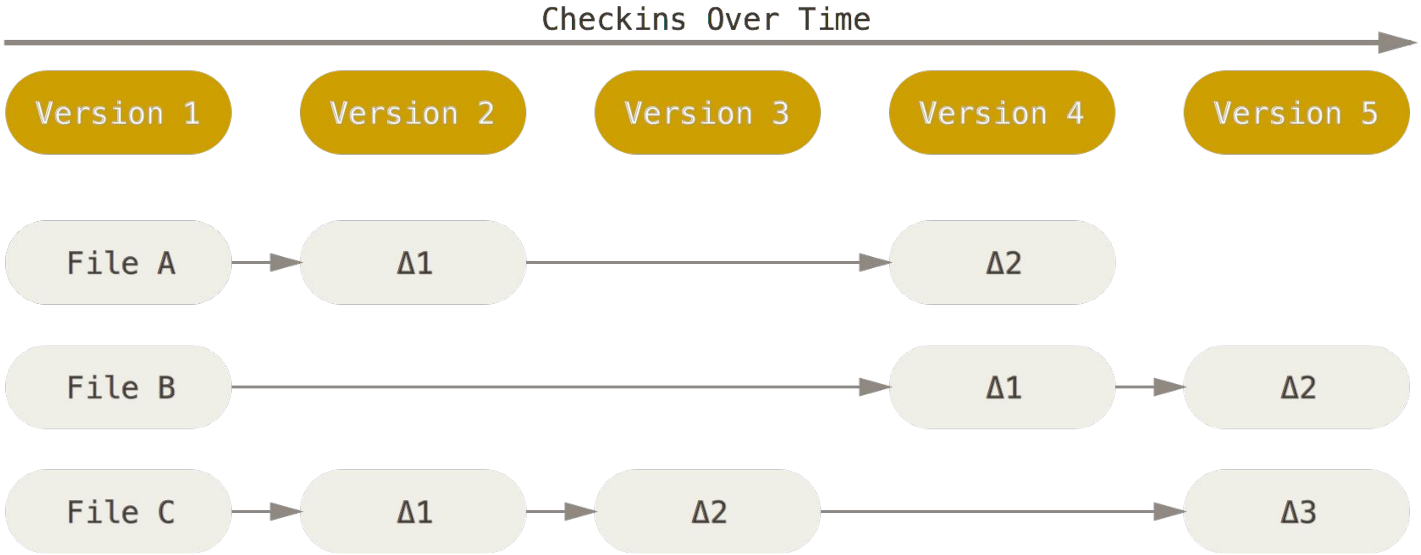
- Date information
- Authorship information
- Change descriptions



https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control

Version history

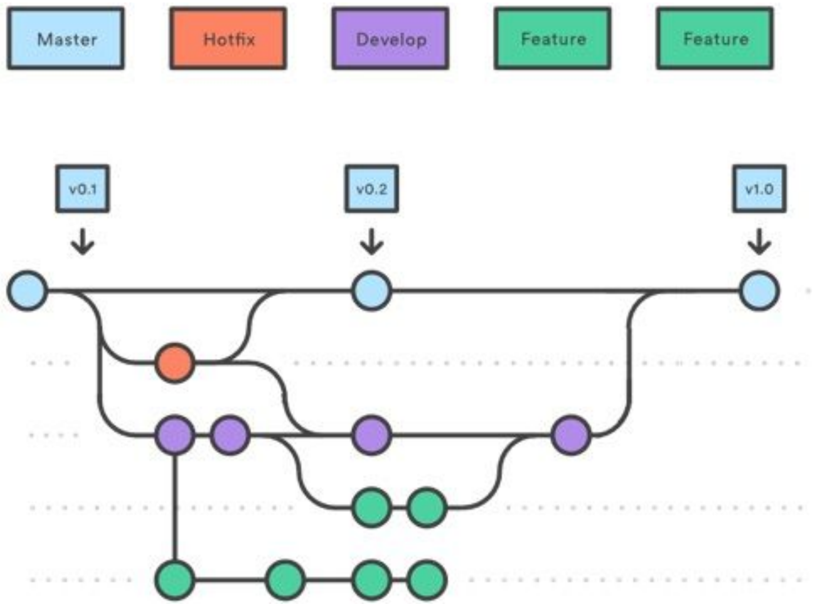
Each version is usually a snapshot of the files at a particular point in time



https://git-scm.com/book/en/v2/Getting-Started-Git-Basics

What does the version history look like?

- Directed acyclic graph
- Each node represents a *commit*
- Commits are organized in *branches*
- Each *commit* can have more than one parent (usually two at most)



http://blog.goprimetime.tv/primetime-process/

Some key terms

Generic terms used to talk about version control, regardless of system

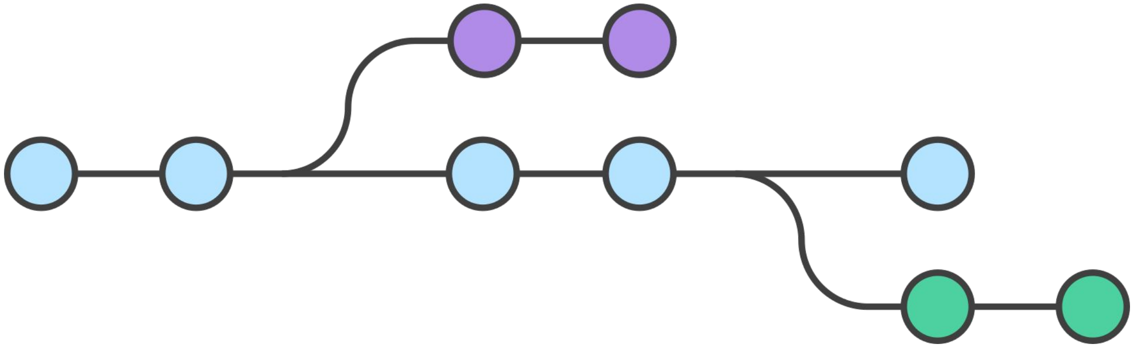
- Commit
- Branch
- Merge
- Repository

Commit or revision

- A snapshot of the state of the project at a particular time
- Can be visualized as a node in the version history
- Usually a commit message is provided explaining the changes

Branch

- Represents an independent line of development
- Has its own version history independent of other branches
- Can be merged with another branch

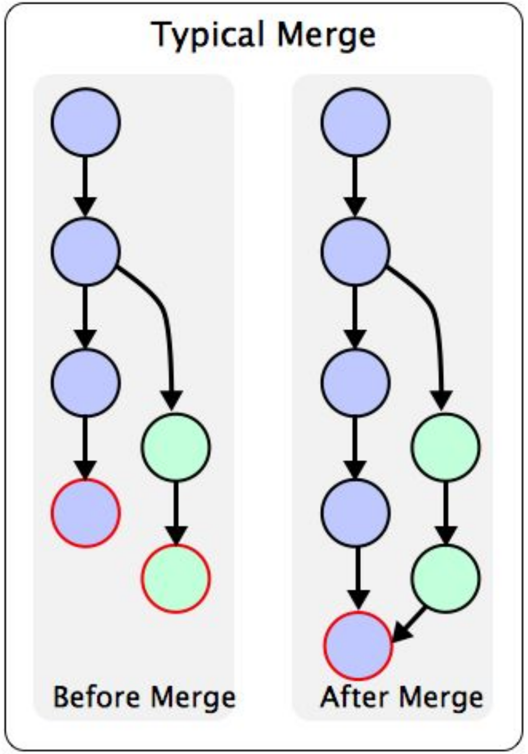


https://www.atlassian.com/git/tutorials/git-merge

Merge

- Joining two or more branches

- *Conflict resolution* is usually required



http://www.bogotobogo.com/cplusplus/Git/Git_GitHub_Fast-Forward_Merge.php

Repository

- A place where the working revision and the version history are stored

- Usually a directory

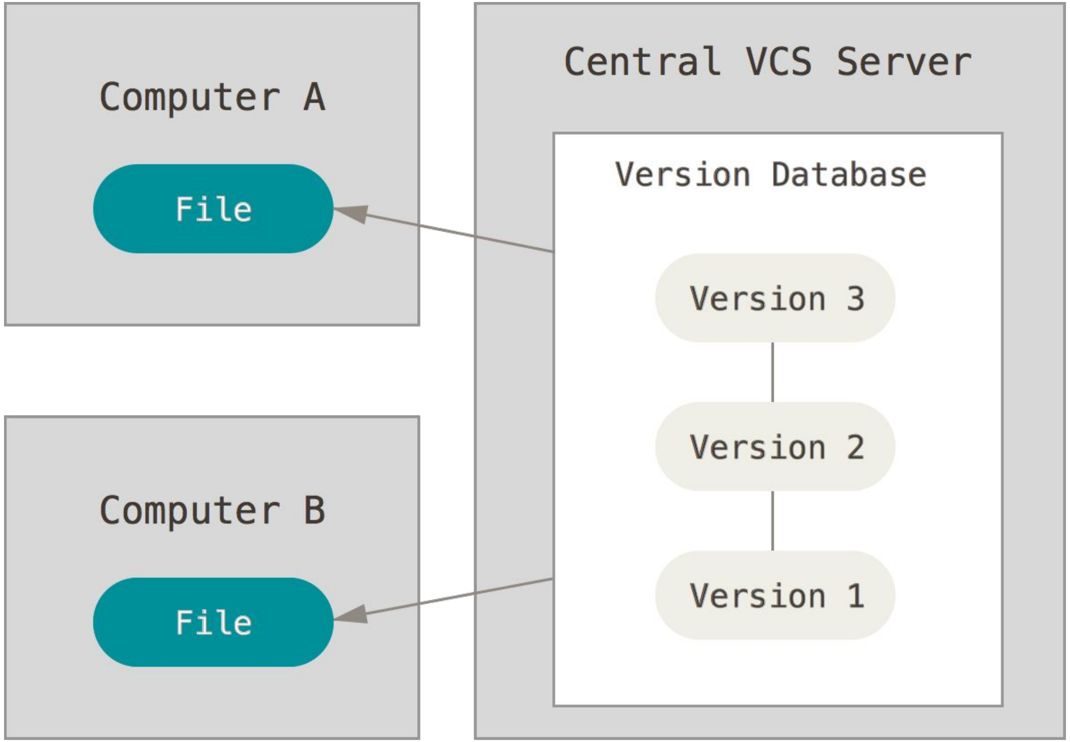
Main approaches to version control

Centralized

Distributed

Centralized version control

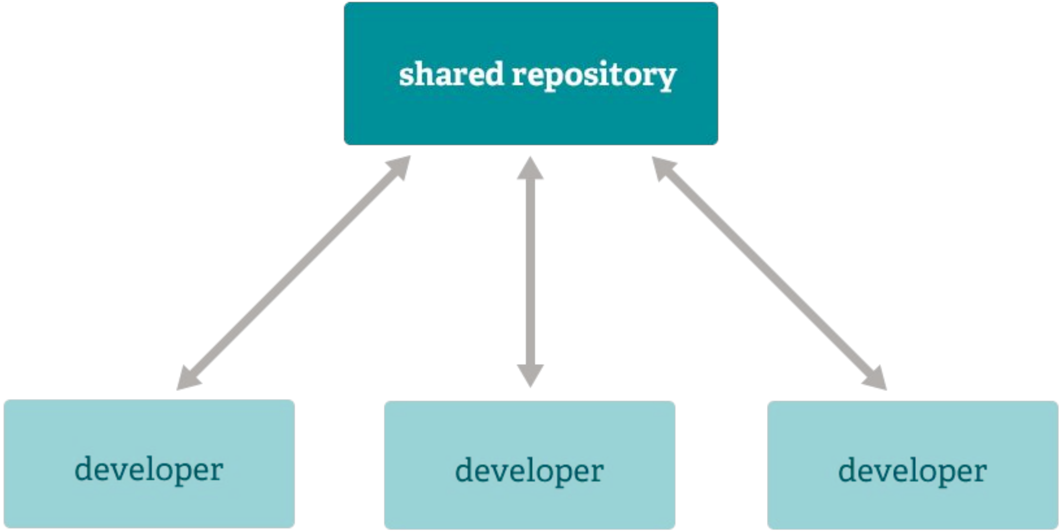
- A central repository hosts all the version history
- Files are modified locally
- A connection is required to alter the history



https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control

Centralized version control workflow

Either file locking is used,
or developers must resolve
conflicts before committing



https://git-scm.com/about/distributed

Advantages of centralized version control

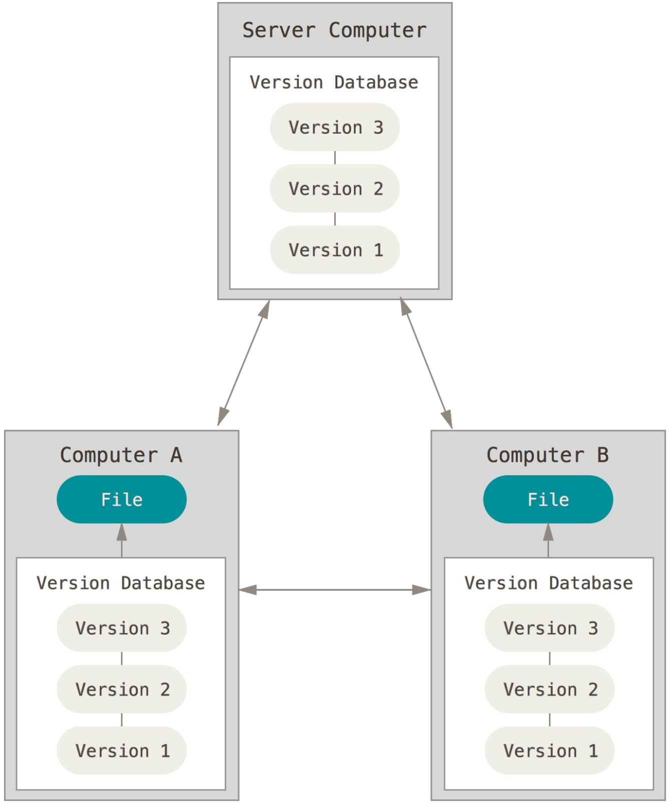
- A lot of administrative control
- It is easier for a developer to see what everyone else is doing
- Straightforward workflow

Disadvantages of centralized version control

- Single point of failure, if server is down no work gets done
- History update operations are as slow as the network
- If backups are not handled correctly, entire history could be lost

Distributed version control

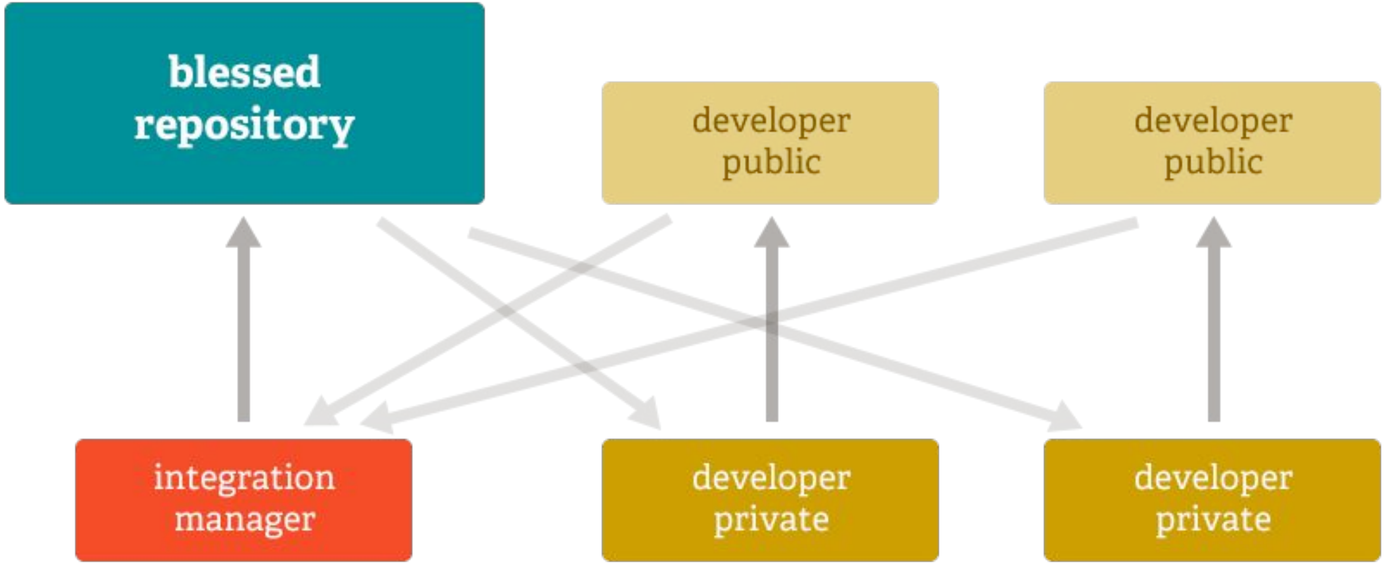
- Every contributor has their own repository
- Files are modified *and committed* locally
- A connection is required to collaborate
- Many collaborative workflows are possible



https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control

Integration manager workflow

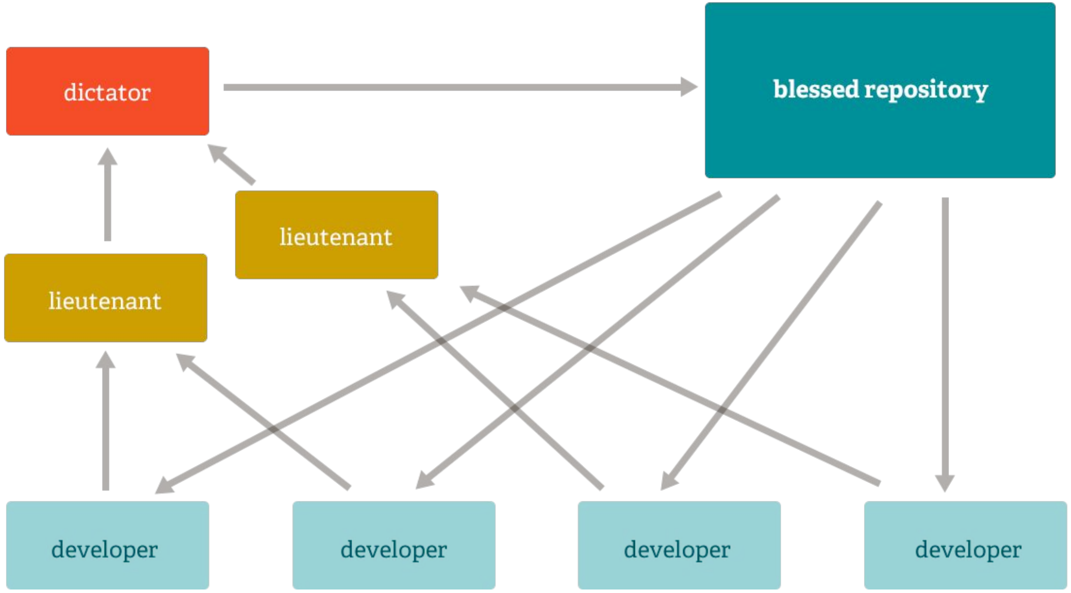
- Only one person commits. Many users contribute through *forks*
- Used by many open-source projects on GitHub



https://git-scm.com/about/distributed

Dictator and lieutenants workflow

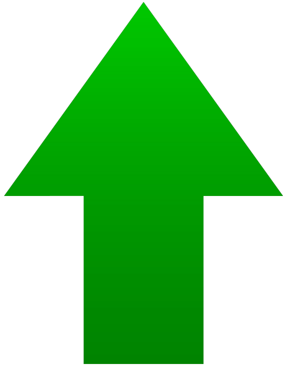
Only dictator can commit. Pulls changes from lieutenants, who pull changes from contributors. Used by the linux kernel.



https://git-scm.com/about/distributed

Advantages of distributed version control

- Each contributor has access to the full change history
- Developers can work offline
- Visualization and modification of history are fast, performed locally
- Multiple working copies protect against data loss
- Enables many different collaboration models



Disadvantages of distributed version control

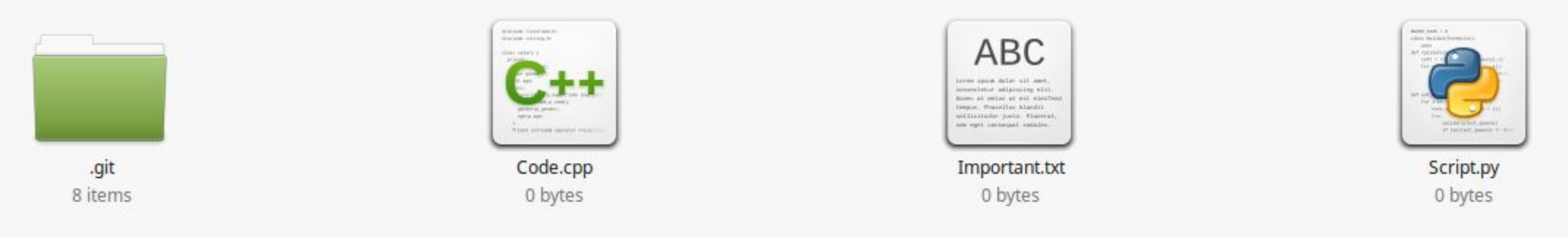
- More complex workflow
- Longer time for initial setup, whole history must be copied
- Steeper learning curve

Git

- Distributed version control system developed by Linus Torvalds starting in 2005
- Focuses on speed and simplicity of design
- It is free and open-source software
- Versioned in Git!

What is a git repository?

Working directory + store



.git
8 items

Code.cpp
0 bytes

Important.txt
0 bytes

Script.py
0 bytes

Working directory

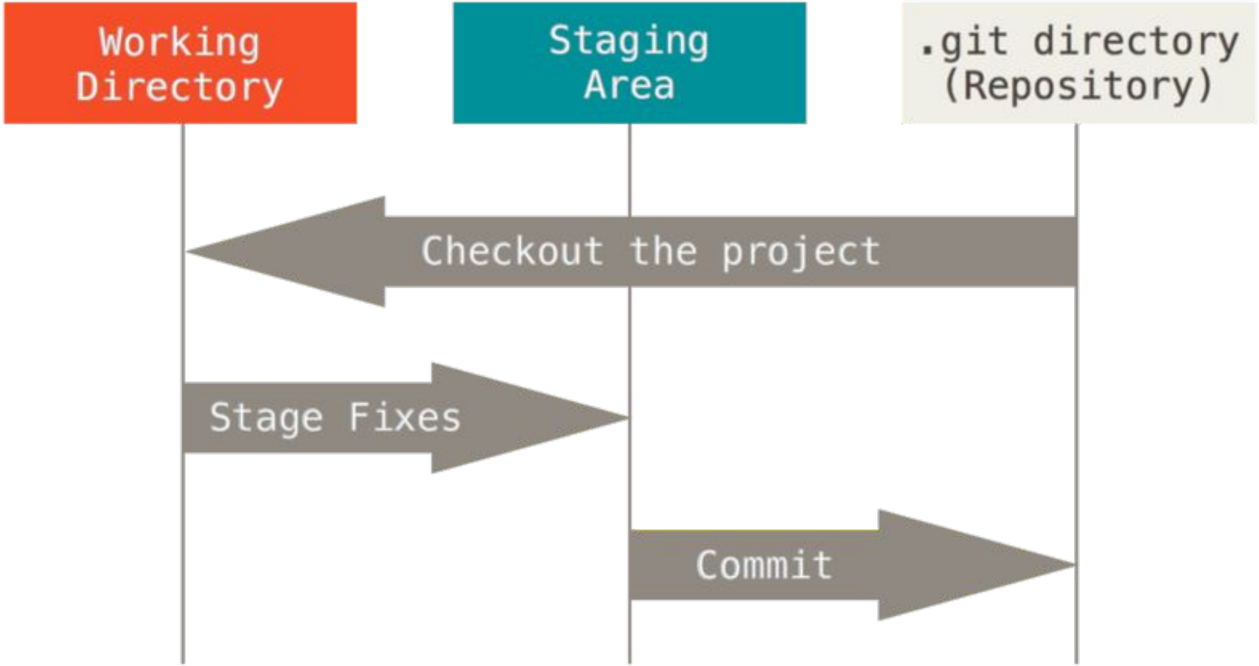
- Contains the project files in a state corresponding to a particular version

- They are ready to be edited

Store

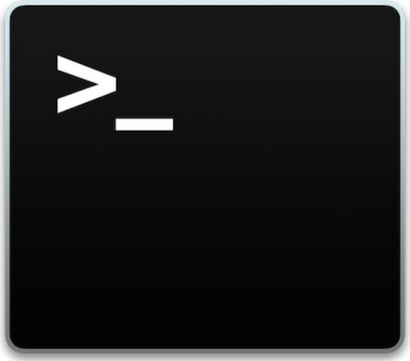
- Contains the complete history of the project
- .git folder
- Do not edit directly!

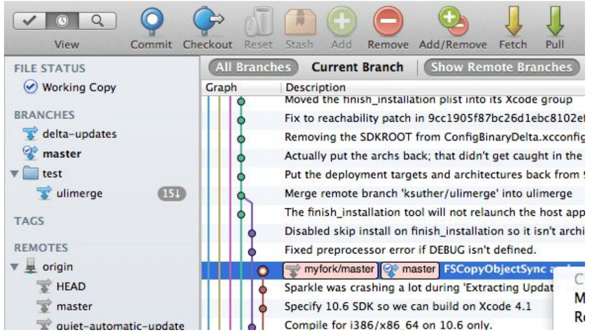
Git workflow



Git basic actions

- Initializing repository
- Ignoring files
- Staging changes
- Committing
- Pushing/Pulling
- Checking out
- Reverting changes
- Branching
- Switching branches
- Merging





```
Moved the finish_installation plist into its Xcode group
Fix to reachability patch in 9cc1905f87bc26d1ebc8102e
Removing the SDKROOT from ConfigBinaryDelta.xcconfig
Actually put the archs back; that didn't get caught in the
Put the deployment targets and architectures back from
Merge remote branch 'ksuther/ulimerge' into ulimerge
The finish_installation tool will not relaunch the host app
Disabled skip install on finish_installation so it isn't archi
Fixed preprocessor error if DEBUG isn't defined.
myfork/master  master  FSCopyObjectSync a
Sparkle was crashing a lot during 'Extracting Updat
Specify 10.6 SDK so we can build on Xcode 4.1
Compile for i386/x86_64 on 10.6 only.
```

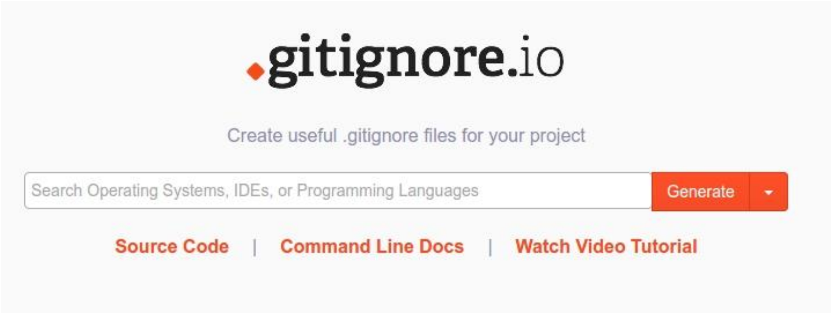
Initializing a repository

- Empty repository
  - `git init`
  - Creates a new blank repository

- Clone
  - `git clone <remote URL>`
  - Copies the whole project history from remote repository
  - Automatically sets *remote* location

Ignoring files

- Done via the .gitignore file
- Simply create, download or generate this file
- Regular expressions tell Git what to ignore



Create useful .gitignore files for your project

Search Operating Systems, IDEs, or Programming Languages
Generate
Source Code | Command Line Docs | Watch Video Tutorial

Types of files in the working directory

- Ignored: Not considered by Git at all
- Untracked: Seen by Git but their changes are not tracked
- Tracked: Git is keeping track of these changes
    - Unstaged: Changes are tracked but will not be committed
    - Staged: Ready to be committed

Checking the status

- `git status`
- This command will inform you about
  - Changed files
  - Untracked files
  - Staged files
- Ignored files don’t show up here

Staging changes

- After changes are made, Git will be aware of them but they will not be *staged*
- Staging means changes are marked to be committed
- `git add -A`
- Stages every non-ignored file
- Specify a list of files instead of `-A` for more control
- Normally *not required* when using *GUI*

Committing

- Adds the staged changes to the version history
- A commit message must be provided
- Use the `-a` option to commit every modified file (untracked files will not be included)
- `git commit -m <commit message>`

A note on commit messages

- Be descriptive!
- Write a quick description and a more detailed summary, like an email
- Focus on *what* and *why* instead of how
- Be concise!
- Good commit messages can facilitate code review

Actions that modify the working directory

- These actions modify the working directory
  - Pull
  - Merge
- Changes must be reverted, *stashed* or committed before they can be carried out
- Git will display an error message otherwise

Pushing

- When you have contributor access to the remote repository and want to upload recent changes
- Git must be configured to track remote repository first (done automatically if cloning)
- `git push -u origin master`
- If remote repository has changed since last pull, push will be rejected
- In this case must pull, merge and push again

Pulling

- When you want to download recent changes from the remote repository
- Remote repository must be configured
- `git pull`
- If someone else has pushed, a merge will be necessary

Checking out

- This will return the working directory to the state of a commit
- `git checkout <commit checksum>`
- Checksum can be obtained by doing `git log`

Reverting changes

- Usually not necessary
- `git revert <commit checksum>`
  - Creates a commit that is the opposite of the one provided
- `git reset --soft <commit checksum>`
  - Resets the index to the state of the commit without modifying working directory
- `git reset --hard <commit checksum>`
  - Resets the index and modifies working directory
  - Dangerous!

Branching

- `git checkout -b <branch name>`
- Creates a new branch
- Does not modify the working directory
- A commit after this operation will be put in the new branch

Switching branches

- Switches to the last commit of the specified branch
- `git checkout <branch name>`
- Does not modify the working directory

Merging

- Switch to the branch into which you want to merge
- `git merge <other branch name>`
- Solve conflicts if any
  - Must use external tool or perform manually
- Provide commit message

This is only the beginning!

- Git offers many more options!
- Many resources exist
  - https://git-scm.com/book/en/v2
  - https://guides.github.com/activities/hello-world/
  - https://www.youtube.com/watch?v=Yq32Ifx0bXw