Git-ting Started With GitHub

Acknowledgements

This training and the images used therein are from https://git-scm.com/book/en/v2/ under the Creative Commons License

The original version of this presentation was created by Taylor Nation

Overview

- VCS & Git
  - VCS overview
  - What is Git?
- Setting up GitHub & GitHub Desktop
- GitHub Pages
- Git Concepts
  - Structure of a Git project
  - Making Changes to a File
  - Branching/Forking
- GitHub Collaboration tools
  - Issues
  - Milestones
  - Pull Requests
  - Git Flow

Commonly Used Git Terms

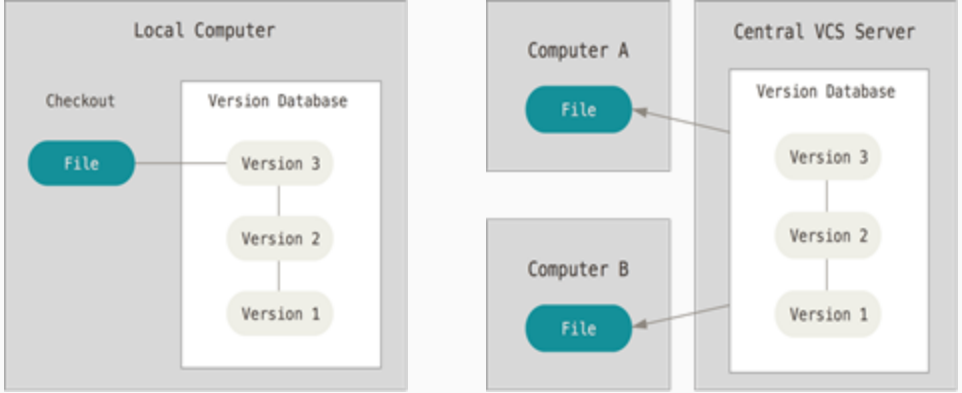
- This Presentation is light on actual git commands, but there is a comprehensive cheatsheet here, a reference manual here, and a very cool visual cheatsheet.
- **Repository**: Data structure containing a history of file changes. A project ***contains*** a git repository, but project and repository are often used interchangeably.
- **Pull**: Retrieve changes from a remote repository, and merge them into the local one.
- **Push**: Update the remote server's version history with your local changes
- **Commit**: Record changes to a repository

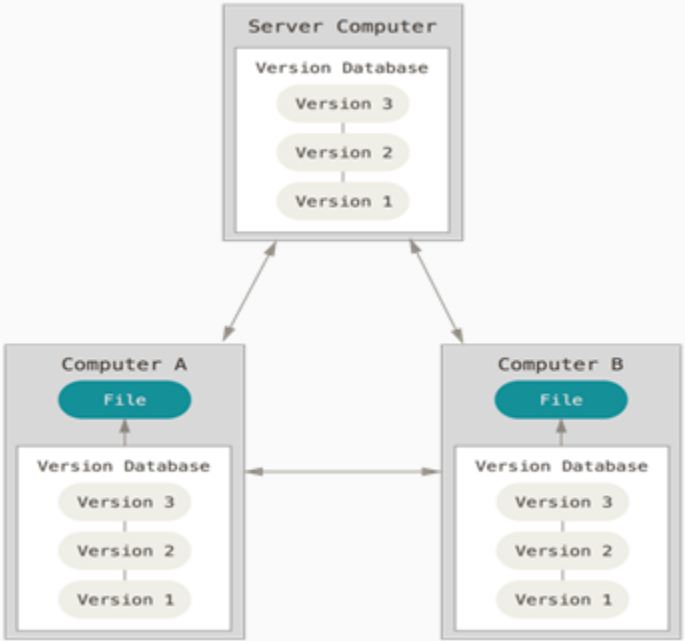
Commonly Used Git Terms (cont'd)

- **Branch**: A copy of the revision history, stored in the repository. Generally used for short term feature development, and merged back into the main tree.
- **Fork**: A copy of revision history, stored in another repository. Generally used to track changes that will not be merged back into the main repository.
- **Pull Request**: the nomenclature varies by organization, but these are requests to merge changes into the main repository.

Version Control Systems (VCS)

- “Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later” (git-scm.com)
- Particularly useful to engineers because it allows us to do quick rollouts/rollbacks, and makes it easy to pinpoint the source of software bugs.
- VCS types: Localized, Centralized, and Distributed
- Git is a distributed VCS.





What is Git?

- From https://git-scm.com : *“Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.”*
- Created in 2005, after the VCS used by Linux Kernel developers revoked their free-use license.
- Used by Facebook, Twitter, Google

What is GitHub?

- GitHub is a project collaboration service that hosts git repositories and handles other aspects of project management.
  - Works with the Git Command Line (CLI); all you need is an account.
- The GitHub API is a useful tool for organizing projects and team collaboration.
  - GitHub issues allow you to keep track of tasks across multiple projects.

GitHub Desktop (and other Git GUI tools)

- Graphical User Interfaces (GUIs) make it easy to interact with Git without remembering any commands.
- Visualizing branches makes it easier to do git operations.
- GitHub desktop is developed by GitHub Engineering, and enables users to do the basic Git commands: branching, pulling, pushing, and committing.
    - This will be enough for most Git/GitHub users, but it appears that some of the more complex commands are missing
- A similar tool, SourceTree by Atlassian, has a more comprehensive set of options available (including Git Flow integration) for advanced users.

GitHub Pages

- Host your website on a GitHub Repository
- Great for blogs, and marketing sites
- More info at https://pages.github.com/

Exercise: Setting Up GitHub Desktop

- Create a GitHub account at https://github.com/join
- Download Git Desktop from https://git-scm.com/download/guis/
  - Sign in and configure your Name and Email address.

Structure of a Git Project

- **Git directory (repository)**: contains the revision database and metadata.
  - Previous commits to files are stored here.
- **Working tree**: A snapshot of one version of the project, stored on disk. Usually, but not necessarily, the latest revision.
  - Created from the changes stored in the Git version database.
  - Ephemeral: changes made in the working tree are not tracked.
- **Staging area**: a section of the git repo that contains information about files that have been modified, but not committed.

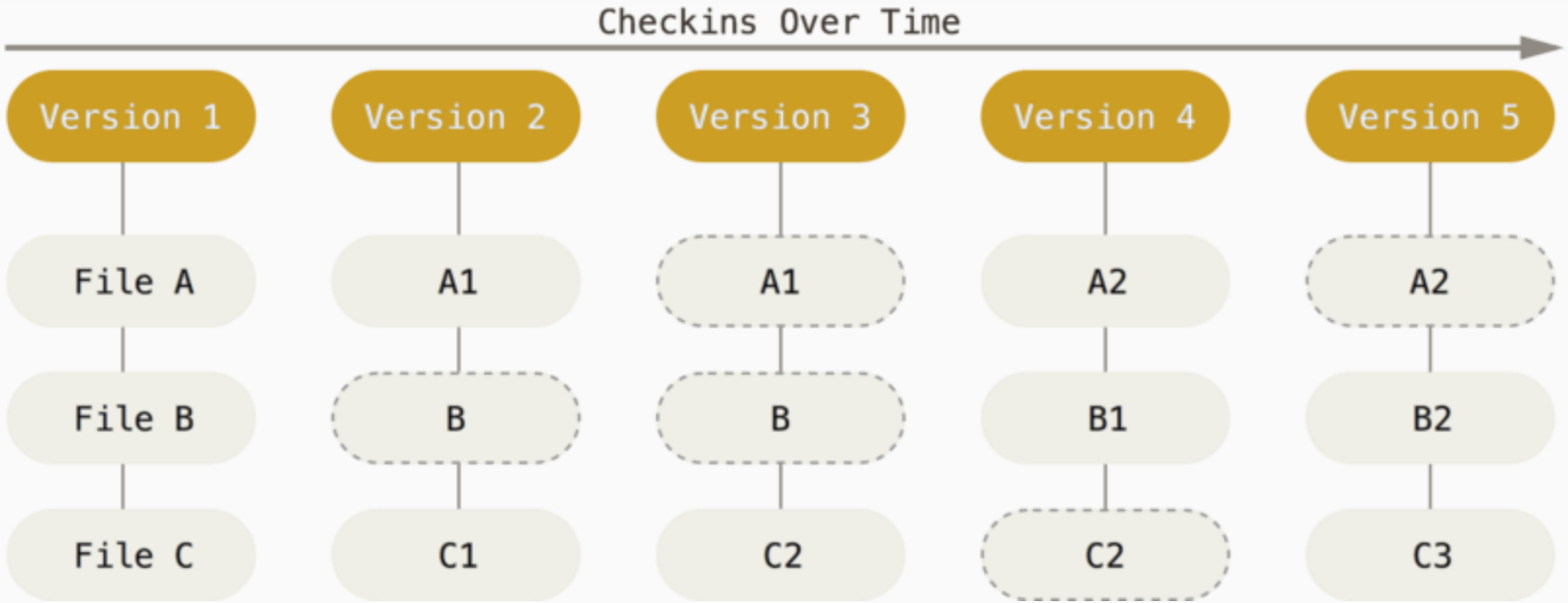
Making Changes to a File

- **Clone** the project you'd like to make edits to
  - **git clone <repository-uri>**
- **Checkout** a version of the project into the working tree
  - **git checkout -b test-branch**
- Make the necessary edits, then **add** them to the Staging area
  - This is ***VERY important***: if you don't add your changes to the staging area, they won't be saved to the version database.
  - **git add edited-file.txt**

Making Changes to a File (cont’d)

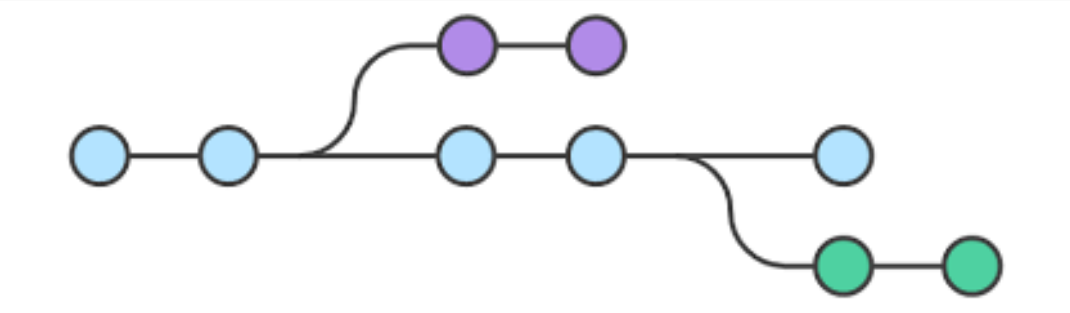
- Review your changes; once you’re satisfied, **commit** them to save your revisions to the database.
  - Inspect changes with **git diff**
  - Save your revisions with **git commit**
- **Push** your changes to the remote
  - **git push origin test-branch**

How Git Views Version History



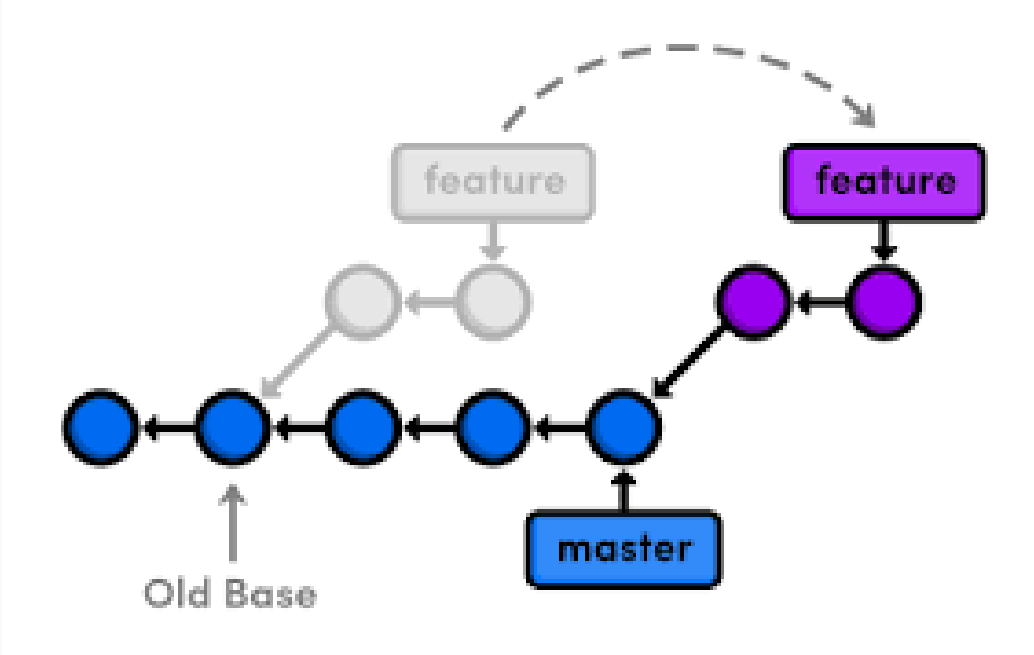
Making Major Changes to a Project

- **Branching** creates a copy of the main revision history that runs parallel to the main history. You can make changes to this copy of the repo without impacting the main line of development.
- **Forking** a repository does the same thing as branching, but it grants full ownership to the user that creates the fork. Forking is useful for creating an idea based on another project, or contributing to a project where you don't have Push access.
- Allows people to experiment with different versions of the source code at the same time.



(Not) N'Sync: Branching/Forking Caveats

- Merge conflicts happens when you try to merge your branch into the main branch, but they're too far out of sync.
  - **Your branch contains changes to a file that's deleted**
  - **Different changes to the same file**
- To limit the amount of conflicts, **git pull** changes into the main branch, then use **git rebase** to add your commits to the top of the newly-updated main branch.
- Sometimes, merge conflicts have to be fixed manually.



Git Workflow Exercise: Creating a GitHub Page

- Download a zip file of https://github.com/tnation14/sample-github-pages-template
- Create a repository in GitHub Desktop: https://github.com/<your username>/<your username>.github.io (i.e. **https://github.com/tnation14/tnation14.github.io**)
  - Take note of the local folder where the repo will live.
- Create a new branch:
  - Click **Current Branch** > **New**
  - Name the branch **master**
- **Copy** the contents of the zip file into your repository.
- **Edit** index.html to say “Hello, <your name>”.
- Open index.html in a web browser
- Add the new files to the staging area, commit changes, then push them to a remote branch
- Merge your changes to the master branch.

GitHub Collaboration Tools

- **Issues**: GitHub's way of tracking bugs
- **Milestones**: Grouping issues by deadline. Also helps break issues down into more manageable chunks
- **Pull requests/Merge Requests (PRs/MRs)**: a request to merge changes from a branch or fork into the main branch.

GitHub Issues: Because There's No Such Thing as Perfect Software

- Useful tool for iterative development
  - Plan, Build, Test, Deploy, Create Issues
- Issue comments are great places for planning between multiple team-members
- Break down complex tasks into more manageable steps
- Good to reference as context for a Pull Request
- Nice feature of GitHub API is that you can link a PR to an issue, so that when you merge the PR the issue is closed as well.
- Example: GeoTrellis Issue and Pull Request
- https://github.com/user/repo/issues

Milestones

- Organizes disparate issues into a single deliverable for time tracking purposes.
  - Multiple kinds of customer issues can be grouped under a single release milestone
- Set timelines for features
- Used in conjunction with Issues, Milestones are a great way to measure how your team’s expected efficiency matches up with its actual efficiency.
- https://help.github.com/en/github/managing-your-work-on-github/about-milestones

Pull Requests

- Asking permission to merge my branch revision history into the main branch
- **Place to peer-review code for correctness and readability.**
- PR Etiquette
  - Give your PR a thorough self-review before submitting it to someone else.
    - https://github.com/user/repo/compare/base-branch...your-branch
  - Describe of the issue you're trying to fix (or link back to an issue)
  - Include good testing instructions
  - When adding changes, avoid rebasing. This can cause merge conflicts for the person testing your commit locally.
  - Before merging, do one final rebase to put your work into logical commit steps
- https://github.com/user/repo/pulls

Git Flow

- Branching pattern tor teams that do rapid iterative development
- Active development happens on “feature” branches
- Completed work is merged from the Feature branch into develop.
- Release branches contain work that’s ready to be deployed
- Keeps in-progress work separate from completed work, and allows multiple teams to work on projects without getting in each other's’ way.

Useful Resources

- Intro to Git and a recommended workflow: https://guides.github.com/introduction/flow/
- Git Cheatsheet: https://github.github.com/training-kit/downloads/github-git-cheat-sheet/
- Very cool visual Git cheatsheet: https://ndpsoftware.com/git-cheatsheet.html
- Git reference manual: https://git-scm.com/docs
- Lengthier digital book: https://git-scm.com/book/en/v2/
- Intro to GitHub short course: https://lab.github.com/githubtraining/introduction-to-github
- My personal cheat sheet I've been adding to