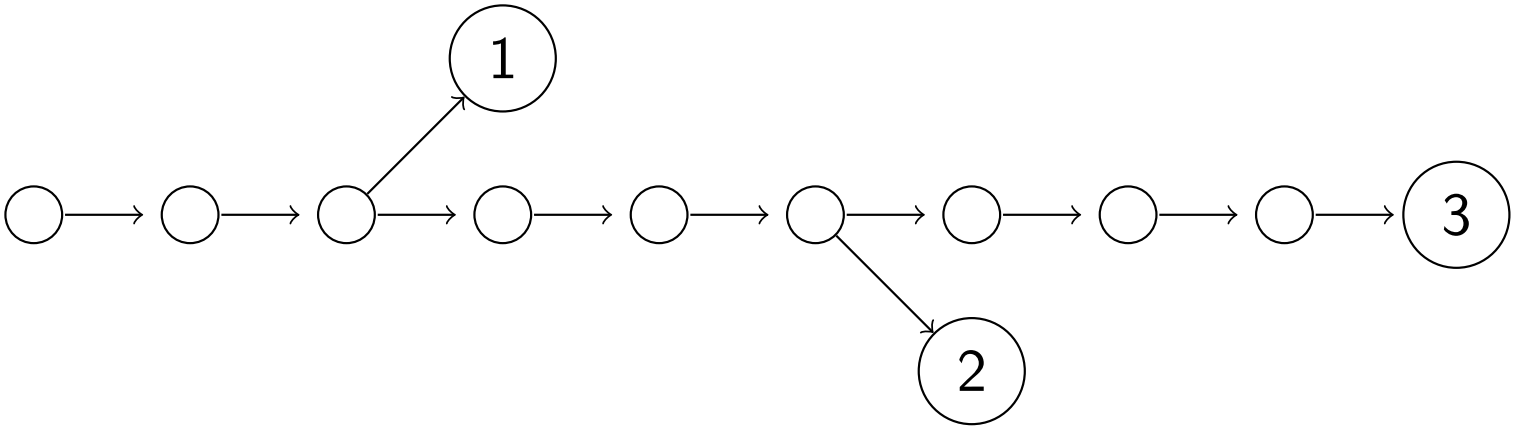
Why Use Version Control Software?

Introduction to Git

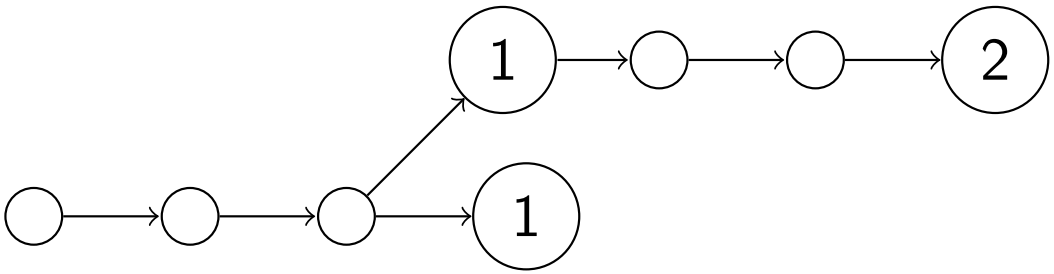
Nick Del Grosso

October 16, 2013

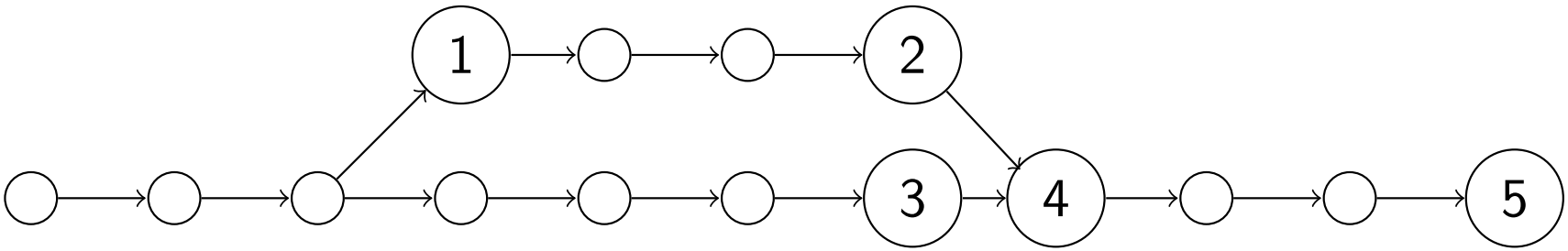
Files are Dynamic Things



Files are Dynamic Things



Files are Dynamic Things



Version Control Software Solves These Problems

Problems with Branching via Copy-Paste:

- High Memory Load
- Punishes High-Complexity Projects
- High Human Error Potential
- Enforces Slow and Low Collaboration
- Branches are Unlikely to Return to the Main Trunk

Benefits:

- Low Memory Load
- Grows with Your Project
- All Errors Can Be Corrected
- Encourages Experimentation and Creativity
- Makes Collaboration Easy and Non-Invasive

What Does Git Do?

What is a Git Repository?

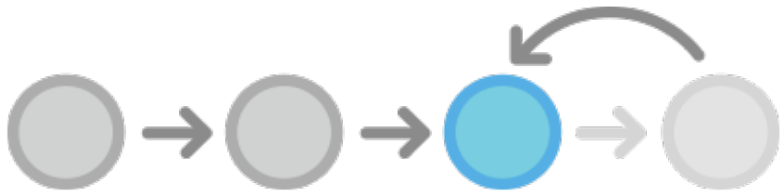
- A normal Directory
- Contains a **.git** folder
- This folder keeps information about file changes in the directory.
- It is only updated when it is told to do so.

What is Git's Purpose?

- Provide tools for organization.
- Protect Your Work History
- Allow Flexible Development
- Allow Distributed Development
- Works on **Any Size** Project.

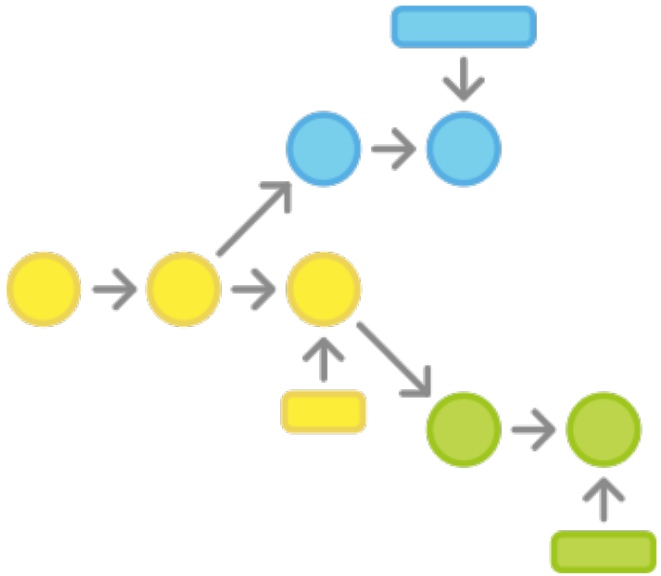
Killer Feature 1: Infinite Undo.

- Current Solution: Never Quit Matlab. Make only small experimental changes. Dont forget anything!
- Git can **Revert** to any version (a "**Commit**") of any file. It can even do it *Non-Linearly*!



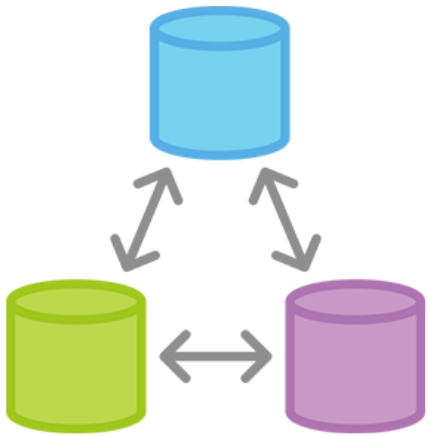
Killer Feature 2: Keep Track of Multiple Versions of a File.

- Current Solution: Save-As and Copy/Paste. Only work with one change at a time.
- Git tracks all versions of a file simultaneously and allows you to switch between any of them.
- Multiple versions can even be **merged**, allowing any number of branches to exist and actually be useful.



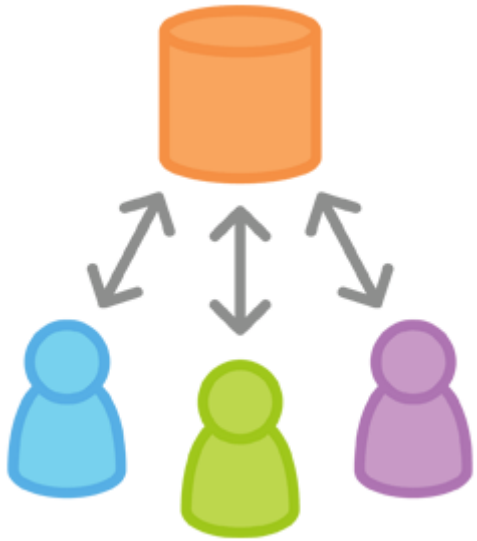
Killer Feature 3: Easy Backup and Syncing.

- Current Solution: Dropbox
- Git can sync with any other repository, **pushing** its updates *to* another repository, or **pulling** updates *from* another repository.
- Git repositories are *mobile*. They can be saved in a Dropbox folder with zero conflicts.
- Git repositories are full *duplicates*, allowing any repo to act as a backup.



Killer Feature 4: Tennis-less Collaboration.

- Current Solutions: Save-As, Server-Side Work, E-Mail
- Git repositories are *distributed*. All people have all versions of the repository, allowing them to work locally (no internet connection required).
- Everyone can work simultaneously on any file, incorporating changes on their own schedule, and always work off the latest version of a repository.



Why Git over Other Version Control Systems?

1. Git is a Distributed Version Control System.
   - Updates are instant.
   - All Repositories act as Backups
   - No Connectivity Requirements
2. Branching is Cheap.
   - Fits any Individual's Workflow
   - Promotes Experimentation in Isolated Sandboxes
3. Git is Lightweight.
   - Only adds a single folder to the entire LabBox directory.

What is the LabBox?

What is Needed from a Development Standpoint?

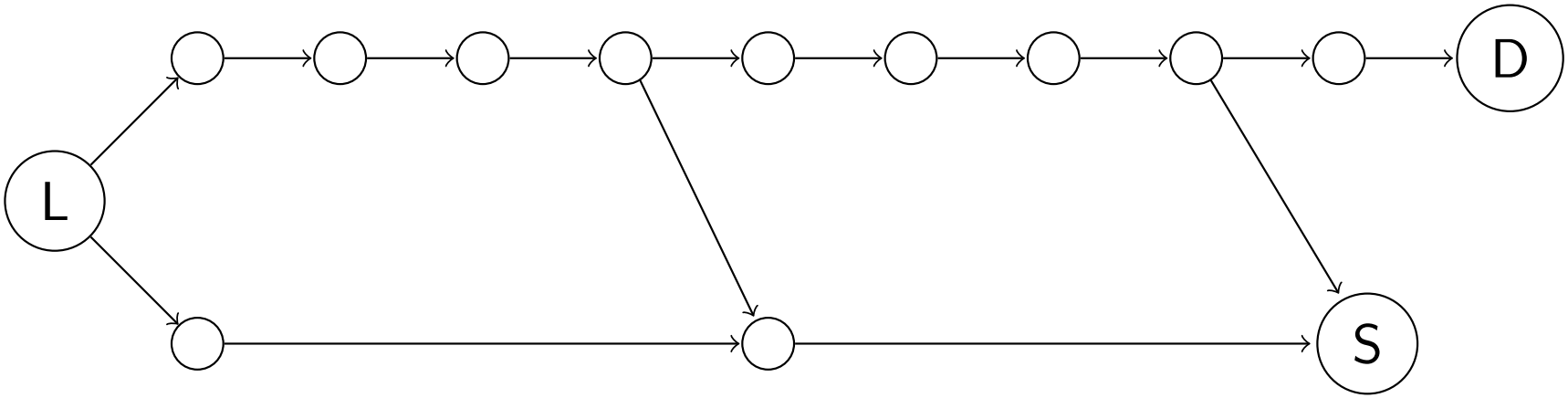
- Must be **stable** for all users, even those not developing.
- Must continue to work for old scripts.
- Updates must be **simple** to perform and **easy to learn**.
- Small updates should not break anyone else's work.
- When just accessing LabBox, Users should be able to **ignore** Git if they want.

Proposed Organization: Centralized, with Three Versions

Three Branches:

1. **"Legacy" Branch:** The state of the Labbox before Git Development began. Needed for really old scripts. Never updated.
2. **"Development" Branch:** The working branch that all developers use. All **Clones, Pulls,** and **Pushes** will be to this branch. Contains the latest features and bugfixes.
3. **"Stable" Branch:** Semi-Recent version of the Labbox, updated in periodic *"Releases"* by **Merging** with the Development Branch. This version will be the one directly available from the directory, and should contain the fewest bugs of the other branches.

Quick Look at the Proposed Workflow



Why Use this Organization?

- Stable
- Easy to Setup (First time only):
  - **git clone -b development [centralDirectory] [myDirectory]**
- Easy to learn to use:
  1. Download the latest updates from the central repository: **git pull**
  2. Commit changes to your local repository **git commit -a -m 'my comments'**
  3. Share your updates: **git pull**, then **git push**.
- Allows very few conflicts compared to other branching workflows, and developers aren't forced to make branches if they don't wish to.
- Easy to manage from the administrator side.

Thank You for Your Time!

Discussion

Color Photos taken from www.atlassian.org

See their Git tutorials and Git GUI Software