



giting Things Done

An Introduction to Git Source Control Software

Written by Explainly, Presented by Ahad Amdani

What Is git?










Simple versioning creates complete copies and uses significant space.

```
ahad@ahads-mbp Version 01 % git log
commit af8ea77e0b7a676e08708c243772a52bc0420f6c (HEAD -> main, experiment1)
Author: Ahad L. Amdani <ahad.amdani@gmail.com>
Date: Sat Jan 2 19:34:48 2021 -0600

Version 2

commit ca695831da9e8a766828a28342f085b75beb49d0
Author: Ahad L. Amdani <ahad.amdani@gmail.com>
Date: Sat Jan 2 19:33:09 2021 -0600

Version 1
```

| | |
|---|---|
|  Version 01 |   |
|  Version 02 - Experimental |   |
|  Version 03 |   |

Git captures transactional changes and incorporates merging, comparisons, and commentary directly into the tool.

What Is **git**?

Do you have experience with source control software?

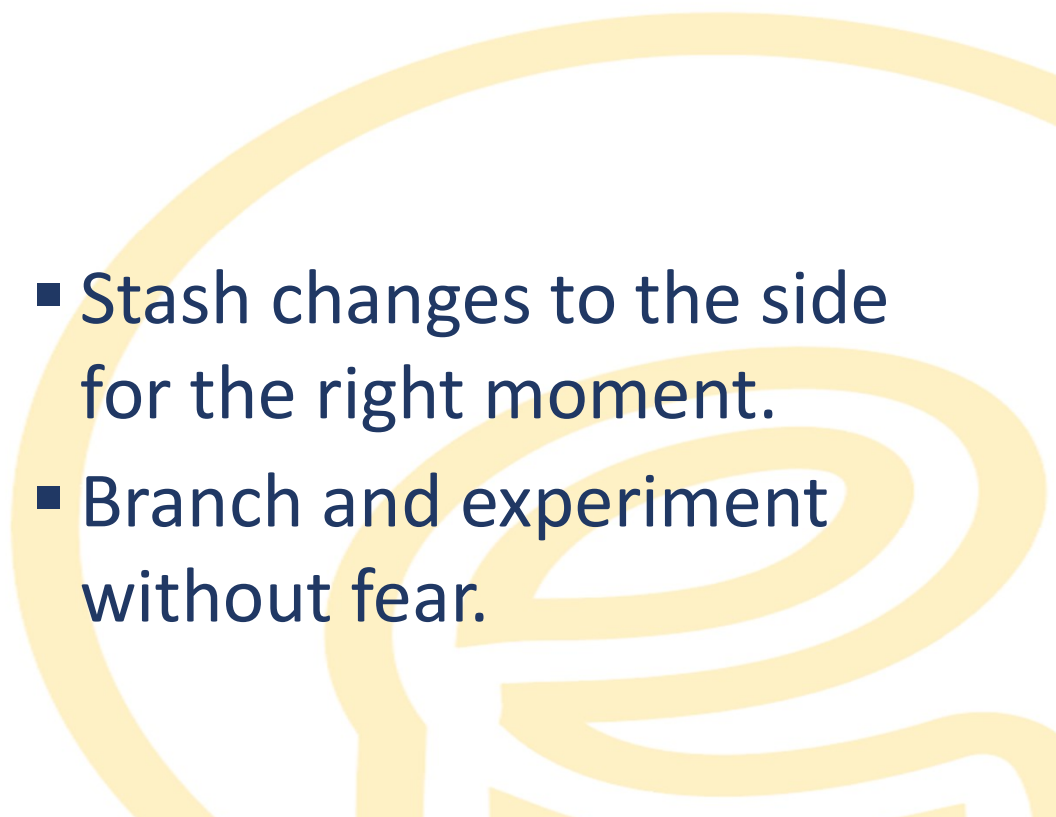
- 1) Never heard of it
- 2) As a feature inside software (R Studio / Visual Studio)
- 3) I've used a GUI tool here or there
- 4) I browse GitHub for fun

What Is **git**?

Git is a distributed source control software system.

The software's goal is to make managing and tracking multiple versions of information faster, easier, and more efficient than simple file versioning.

What Does Git Do?

- Track changes for any file type.
 - Save snapshots of any set of files (or changeset).
 - Stage the changes you want to version (which files are part of the set).
 - Stash changes to the side for the right moment.
 - Branch and experiment without fear.
- 

How Can Git Help Me?

Git enables you to rollback to any point in history of your tracked repository.

Git also allows for reporting and comparisons across different versions of history.

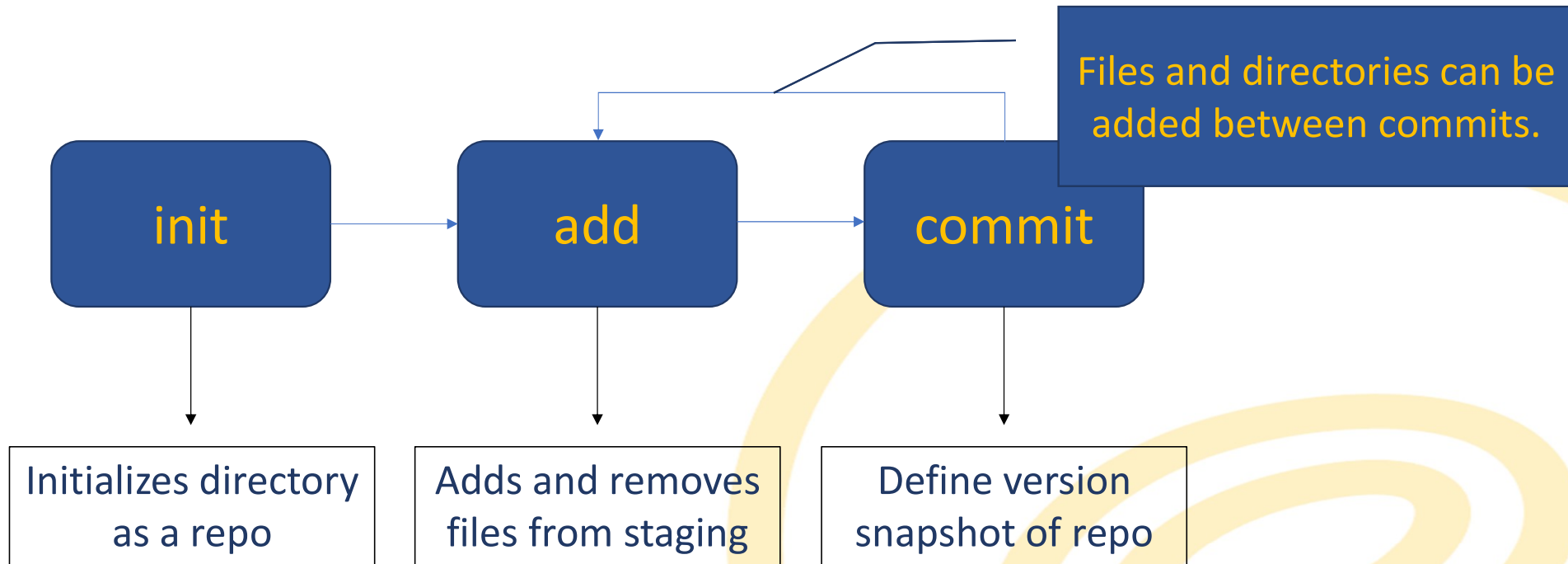
- Data Integrity
- Speed and performance
- Small footprint
- Improve efficiency
- Easy to learn

What Is A Repository?

- Commonly called a “repo”
- Reflects a version-controlled directory along with any specified contents.
- Encompasses all subfolders and files.
- Works with any file types.

Git works with differences across any filetypes from simple files such as a text or CSV document to more complex types such as Excel or binary files.

Repository Management Overview



Initialize Repo - init

- Create a directory or select a pre-existing directory.
- Execute “git init” command from within directory.
- Git repository will initialize.
- No files or subdirectories will be added to staging.

```
ahad@ahads-mbp Version 01 % pwd
/Users/ahad/code/explainly/git_intro_webinar/Version 01
ahad@ahads-mbp Version 01 % ls
DataTriangle.csv      actuarial.py          analysis_main.py
ahad@ahads-mbp Version 01 % git init
Initialized empty Git repository in /Users/ahad/code/explainly/git_intro_webinar/Version 01/.git/
ahad@ahads-mbp Version 01 %
```

Managing Repo Files - add

```
ahad@ahads-mbp Version 01 % git add actuarial.py
ahad@ahads-mbp Version 01 % git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   actuarial.py

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    DataTriangle.csv
    analysis_main.py
```

- The “add” command adds and removes files and directories from staging.
- Syntax is:
git add “filename”
- Adding the file to staging does not create a snapshot in time.
- Only files added to staging will have their history tracked.

Managing Repo Files - add

```
ahad@ahads-mbp Version 01 % git add -A
ahad@ahads-mbp Version 01 % git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   DataTriangle.csv
    new file:   actuarial.py
    new file:   analysis_main.py

ahad@ahads-mbp Version 01 % git commit -m "Version 1"
[main (root-commit) ca69583] Version 1
 3 files changed, 106 insertions(+)
 create mode 100644 DataTriangle.csv
 create mode 100644 actuarial.py
 create mode 100644 analysis_main.py
```

- Simple and easy shortcut is `git add -A`
- Command adds all files and subdirectories to staging.
- Also removes deleted files from staging.
- Still does not create a snapshot of files.

Managing Repo Files – status / reset

```
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   actuarial.py

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    DataTriangle.csv
    analysis_main.py

ahad@ahads-mbp Version 01 % git reset actuarial.py
ahad@ahads-mbp Version 01 % git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    DataTriangle.csv
    actuarial.py
```

- To see current staging status use:
git status
- Displays newly added files to staging and files with changes since last snapshot.
- To reset addition of new files:
git reset
- Does not undo changes to files, simply removes newly added files from staging.

Create A Snapshot – commit / diff

```
ahad@ahads-mbp Version 01 % git status
On branch main

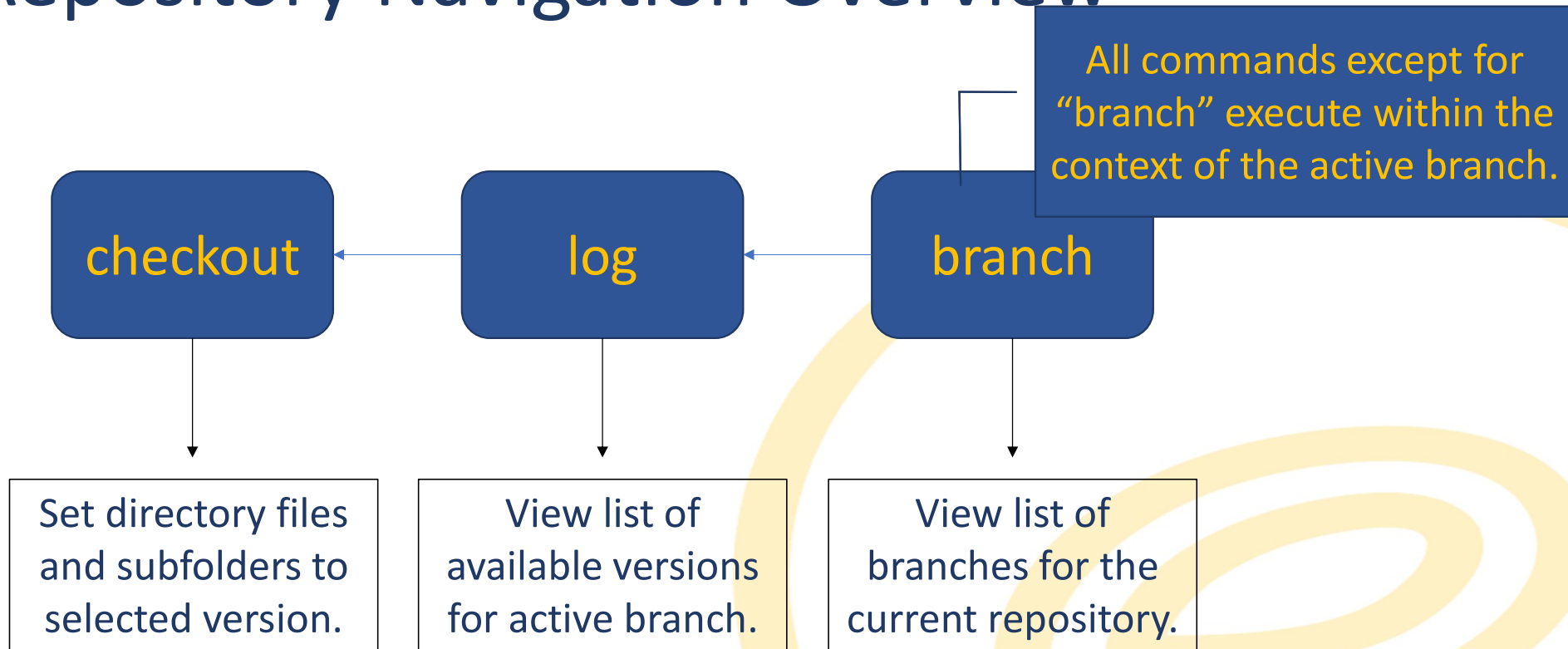
No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   DataTriangle.csv
    new file:   actuarial.py
    new file:   analysis_main.py

ahad@ahads-mbp Version 01 % git commit -m "Version 1"
[main (root-commit) ca69583] Version 1
 3 files changed, 106 insertions(+)
 create mode 100644 DataTriangle.csv
 create mode 100644 actuarial.py
 create mode 100644 analysis_main.py
ahad@ahads-mbp Version 01 % git status
On branch main
nothing to commit, working tree clean
```

- Git status will show list of new or changed files.
- To save a snapshot of the directory git commit –m “message”
- Saves changes made to staged files as a new commit.
- Best practices include a message describing the context for the changes.

Repository Navigation Overview



Navigating a Repo – log / checkout

```
ahad@ahads-mbp Version 01 % git log
commit ca695831da9e8a766828a28342f085b75beb49d0 (HEAD -> main)
Author: Ahad L. Amdani <ahad.amdani@gmail.com>
Date: Sat Jan 2 19:33:09 2021 -0600

Version 1
```

```
Version 2

commit ca695831da9e8a766828a28342f085b75beb49d0
Author: Ahad L. Amdani <ahad.amdani@gmail.com>
Date: Sat Jan 2 19:33:09 2021 -0600

Version 1
ahad@ahads-mbp Version 01 % git checkout ca695831da9e8a766828a28342f085b75beb49d0
Note: switching to 'ca695831da9e8a766828a28342f085b75beb49d0'.
```

- To view available commits:
git log
- Will display commit id, author, date, and message for each snapshot.
- Only displays commits for currently active branch.
- To set directory to a specific snapshot use:
git checkout <commitID>

Navigating a Repo – branch / checkout

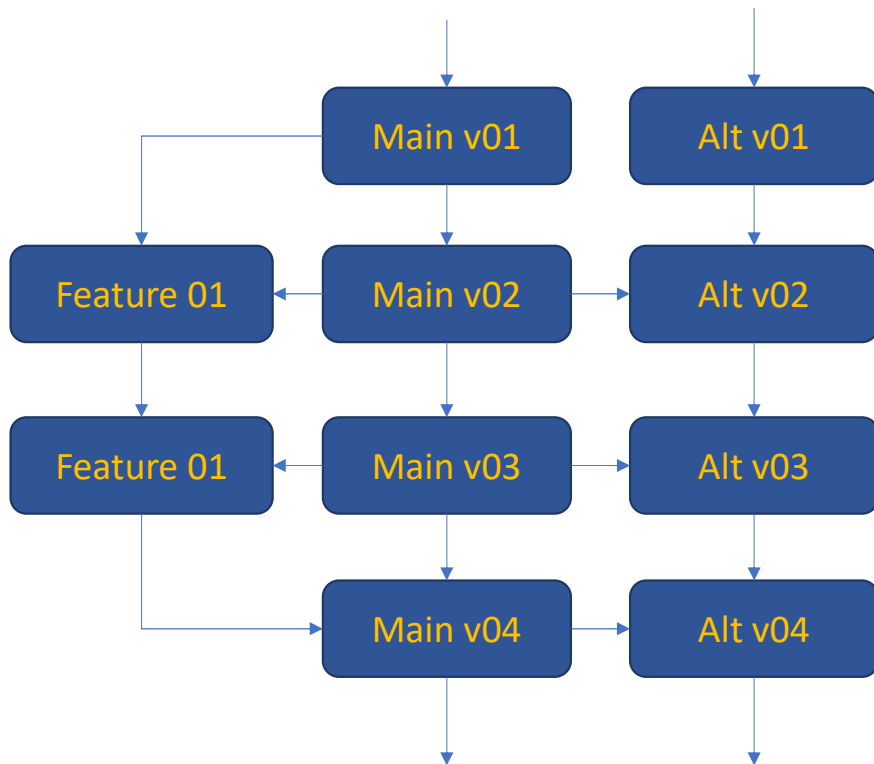
```
ahad@ahads-mbp Version 01 % git log
commit 97550110f1182bfb6fa8bc9ae3cef7104a8beab4 (HEAD -> experiment2)
Author: Ahad L. Amdani <ahad.amdani@gmail.com>
Date: Sat Jan 2 19:36:32 2021 -0600
```

Version 3

```
ahad@ahads-mbp Version 01 % git branch
* experiment2
  main
ahad@ahads-mbp Version 01 % git checkout main
Switched to branch 'main'
```

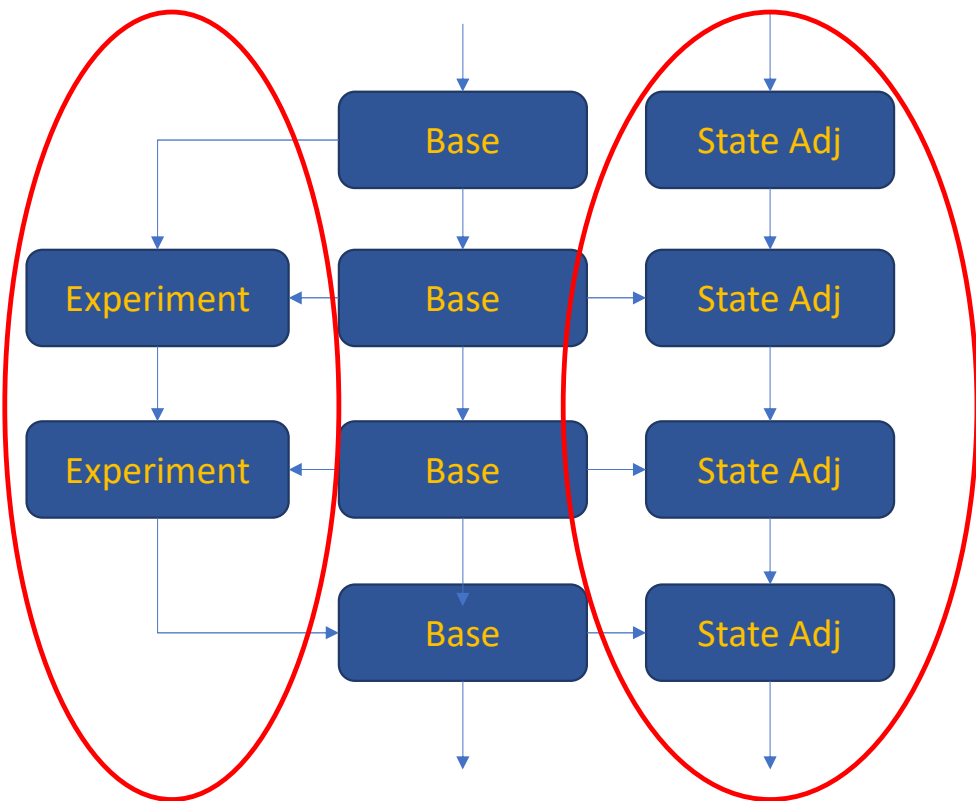
- To view available branches:
git branch
- Will display branch name for each branch within repo.
- To navigate to a branch use:
git checkout <branchName>
- This selects the latest version within that branch.
- Other versions of that branch can then be loaded via checkout.

Experiments – Branching and Merging



- A branch is a parallel version of the repository, like an alternate reality.
- Can be used for testing new functionality or for maintaining alternative versions built from a baseline case.
- Can be joined together via “merge” to reconcile changes into a single, unified version.

Experiments – Branching and Merging



- Merge commands can bring the changes from one version into another in either direction.
- A permanent branch for a special case version (such as state-specific forms or features) can allow for parallel maintenance and development across two paths.
- A temporary branch can allow for development of a new feature while maintaining the production version.

Branches

```
ahad@ahads-mbp Version 01 % git branch
* main
ahad@ahads-mbp Version 01 % git checkout -B experiment1
Switched to a new branch 'experiment1'
ahad@ahads-mbp Version 01 % git status
On branch experiment1
nothing to commit, working tree clean
```

```
ahad@ahads-mbp Version 01 % git branch
experiment1
* main
ahad@ahads-mbp Version 01 % git checkout experiment1
Switched to branch 'experiment1'
ahad@ahads-mbp Version 01 % git status
On branch experiment1
nothing to commit, working tree clean
ahad@ahads-mbp Version 01 % git checkout -B experiment2
Switched to a new branch 'experiment2'
ahad@ahads-mbp Version 01 % git branch -D experiment1
Deleted branch experiment1 (was af8ea77).
ahad@ahads-mbp Version 01 % git branch
* experiment2
main
```

- To create a branch:
git checkout -B “branchname”
- Will create a copy of the current branch as a starting point for the new branch, named “branchname”
- Git commands are local to the currently active branch.
- Delete a branch with:
git branch -D “branchname”

Merges – merge

```
ahad@ahads-mbp Version 01 % git checkout main
Switched to branch 'main'
ahad@ahads-mbp Version 01 % git merge experiment1
Updating ca69583..af8ea77
Fast-forward
 .gitignore      | 1 +
 DataTriangle.csv | 16 ++++++++
 publishing.py   | 22 ++++++++
 3 files changed, 37 insertions(+), 2 deletions(-)
 create mode 100644 .gitignore
 create mode 100644 publishing.py
ahad@ahads-mbp Version 01 % git status
On branch main
nothing to commit, working tree clean
ahad@ahads-mbp Version 01 % git log
commit af8ea77e0b7a676e08708c243772a52bc0420f6c (HEAD -> main, experiment1)
Author: Ahad L. Amdani <ahad.amdani@gmail.com>
Date: Sat Jan 2 19:34:48 2021 -0600

Version 2
```

- To merge one set of changes into the current version:
git merge “branch/commit”
- All of the changes within the named branch/commit will be merged into the active branch.
- The log will show the resulting overlap of branches.

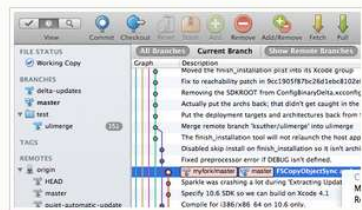
Git Interface Tools

GUI Clients

Git comes with built-in GUI tools for committing ([git-gui](#)) and browsing ([gitk](#)), but there are several third-party tools for users looking for platform-specific experience.

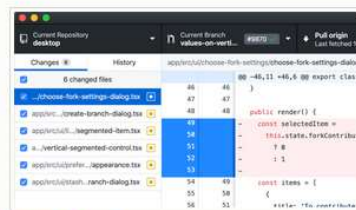
If you want to add another GUI tool to this list, just [follow the instructions](#).

All Windows Mac Linux Android iOS



SourceTree

Platforms: Mac, Windows
Price: Free
License: Proprietary



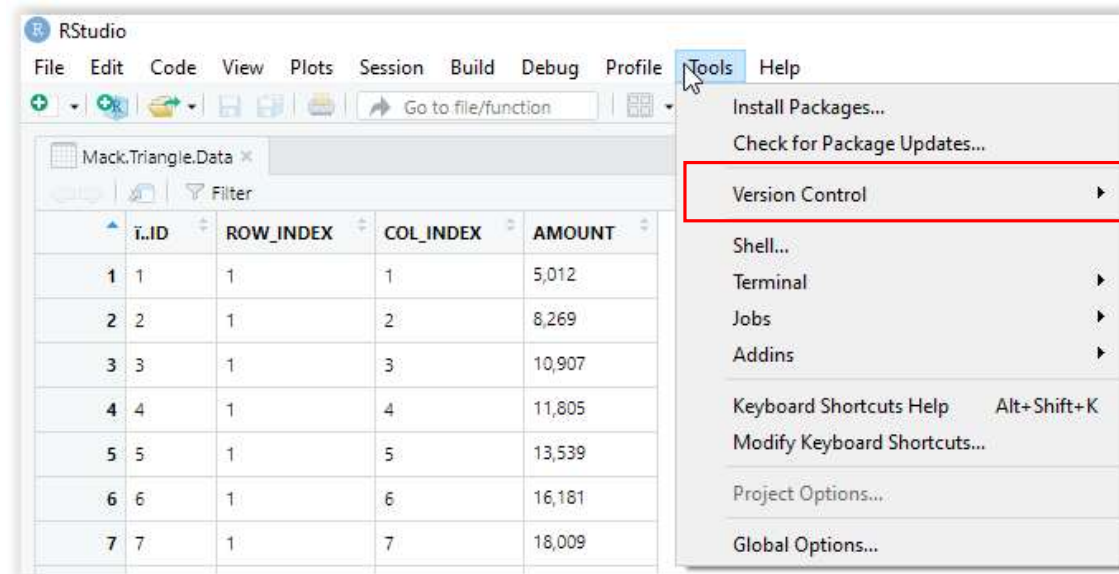
GitHub Desktop

Platforms: Mac, Windows
Price: Free
License: MIT

- Git is a popular tool with many third-party implementation options.
- GUI clients offer ways to implement git without extensive knowledge of the command line functions.
- Git GUIs tools include Tower, GitHub Desktop, Fork, among many others.

Git Built-In Support

- Many tools integrate Git directly into their interface, including some tools common among the actuarial community.
- R Studio, PyCharm (plus other JetBrains offerings), and Microsoft Visual Studio all have Git integrated directly into their software.

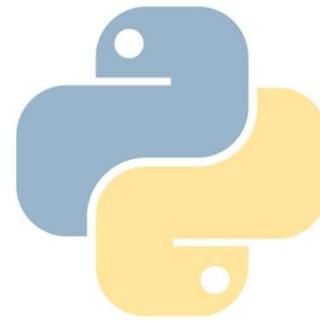


Collaborative Git Tools



- Webinar focuses on local git repositories.
- Online collaborative repositories allow multiple users to work on, interact, and merge changes within a single repo.
- Adds new commands including clone, push, fetch, and pull for remote interaction.
- Common providers include:
 - GitHub
 - GitLab
 - BitBucket

Git Ideas For Actuaries



- Track and manage versions of a rating manual produced from Python and published in LaTeX.
- Collaborate and manage R / Python functions library.
- Compare, contrast, and track client-provided files.
- Maintain and manage statistical tables (such as life or frequency tables) with change histories.

Conclusion



Git can improve business workflows in many different scenarios by assisting with version tracking for reports, code, manuals, and more.

While traditionally a software development tool, git has many applications across businesses and industries.



xplainly



Q&A

Explainly and I would like to thank you for participating in our webinar.

To learn more about us, visit us at <https://www.explainly.io>