Information Management Group

Git & GitHub Basics

A brief overview

- Git Introduction & Workflow
- Git Commands
- GitHub introduction & Workflow
- Hosting your webpage on GitHub
- Reference material

Git: What and Why

What is Git?

- It is a *Version Control System (VCS)*
  - Version control is a system that records changes to a file or set of files over time so that you can recall and revert to any specific versions later.
- Created by Linus Torvalds in 2005, to help maintain The Linux Kernel Project.

Why VCS is important?

Have you ever:

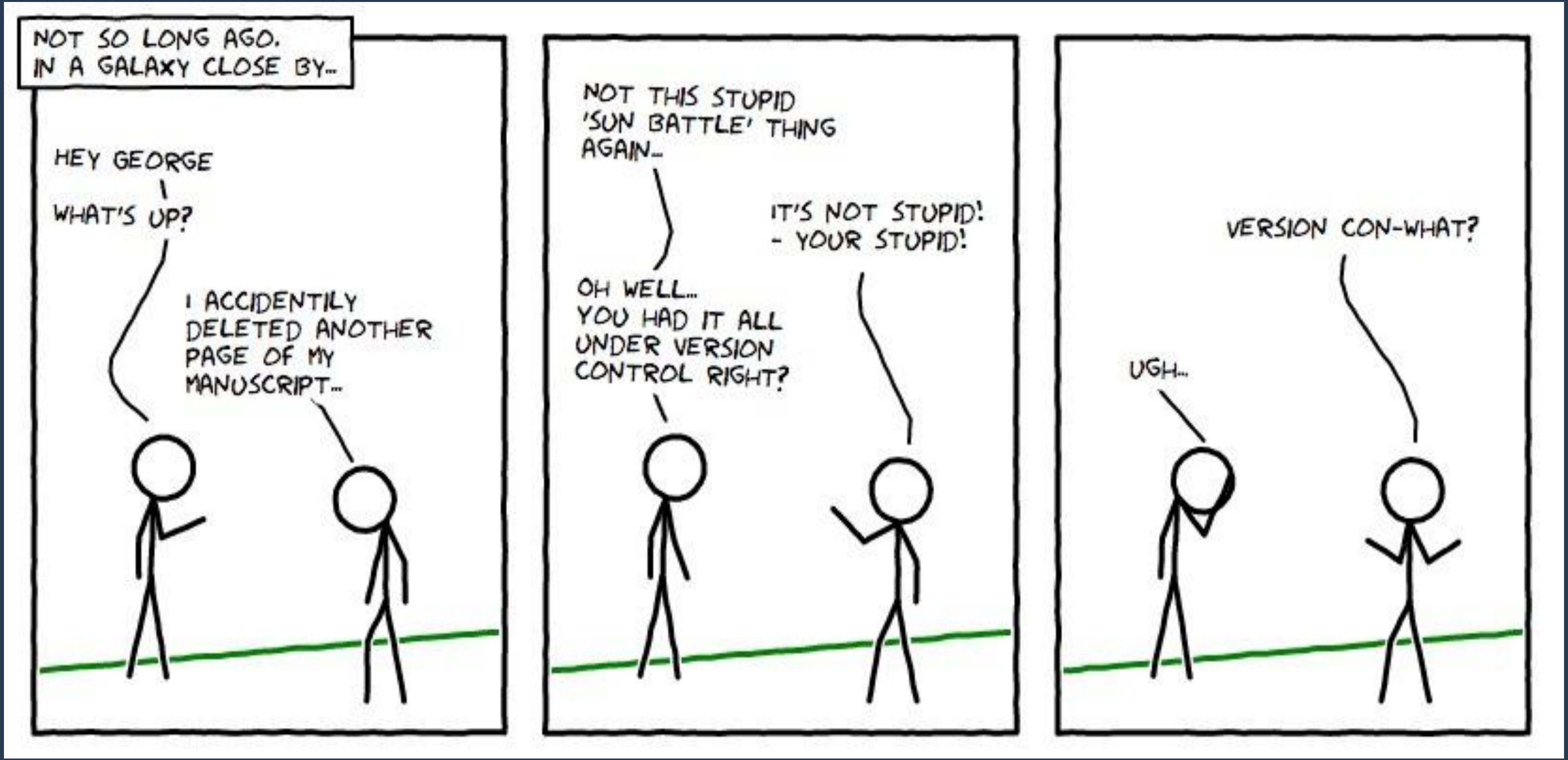
- Made a change to code, realised it was a mistake and wanted to *revert back*?
- Lost code or had a backup that was too old?
- Had to maintain multiple versions of a product?
- Wanted to see the difference between two (or more) versions of your code?
- Wanted to prove that a particular change broke or fixed a piece of code?
- Wanted to review the history of some code?
- Wanted to submit a change to someone else's code?
- Wanted to share your code, or let other people work on your code?
- Wanted to see how much work is being done, and where, when and by whom?
- Wanted to experiment with a new feature without interfering with working code?

Credits: stackoverflow.com/a/1408464/11752644

Why VCS is important?

VCS takes care of all the mentioned problems and much more!

- Every VCS has a concept of *commits*, using which it keeps a track record of all the changes done in your project.
- Using these commits, one can easily
  - Recall and revert back to any specific previous state.
  - Compare code changes.
  - And much more!


NOT SO LONG AGO.
IN A GALAXY CLOSE BY...
HEY GEORGE
WHAT'S UP?
I ACCIDENTILY DELETED ANOTHER PAGE OF MY MANUSCRIPT...
NOT THIS STUPID 'SUN BATTLE' THING AGAIN...
OH WELL... YOU HAD IT ALL UNDER VERSION CONTROL RIGHT?
IT'S NOT STUPID! - YOUR STUPID!
UGH...
VERSION CON-WHAT?

Why Git?

Git is

- Free and Open-Source software
- Distributed/Decentralized: Simply means that multiple people can work on same project without being in the same network
- Uses Snapshot approach
- Has offline git history.
- Very popular, widely used, and accepted as a standard version control system by the vast majority within the developer's community.

Some of the other VCS are Apache Subversion (SVN), Mercurial

Git Workflow

What happens when you use Git in your project?

After you initialize an empty git repository in any directory, it divides your project into three parts:

1. Working Tree
2. Staging Area
3. Git Repository

After this, you can create any number of *commits* (versions/snapshots) of your project as checkpoints where you can revert back to in future.

Working Tree

- The project directory that we see.
- Git will track all the files we add/edit/delete here.

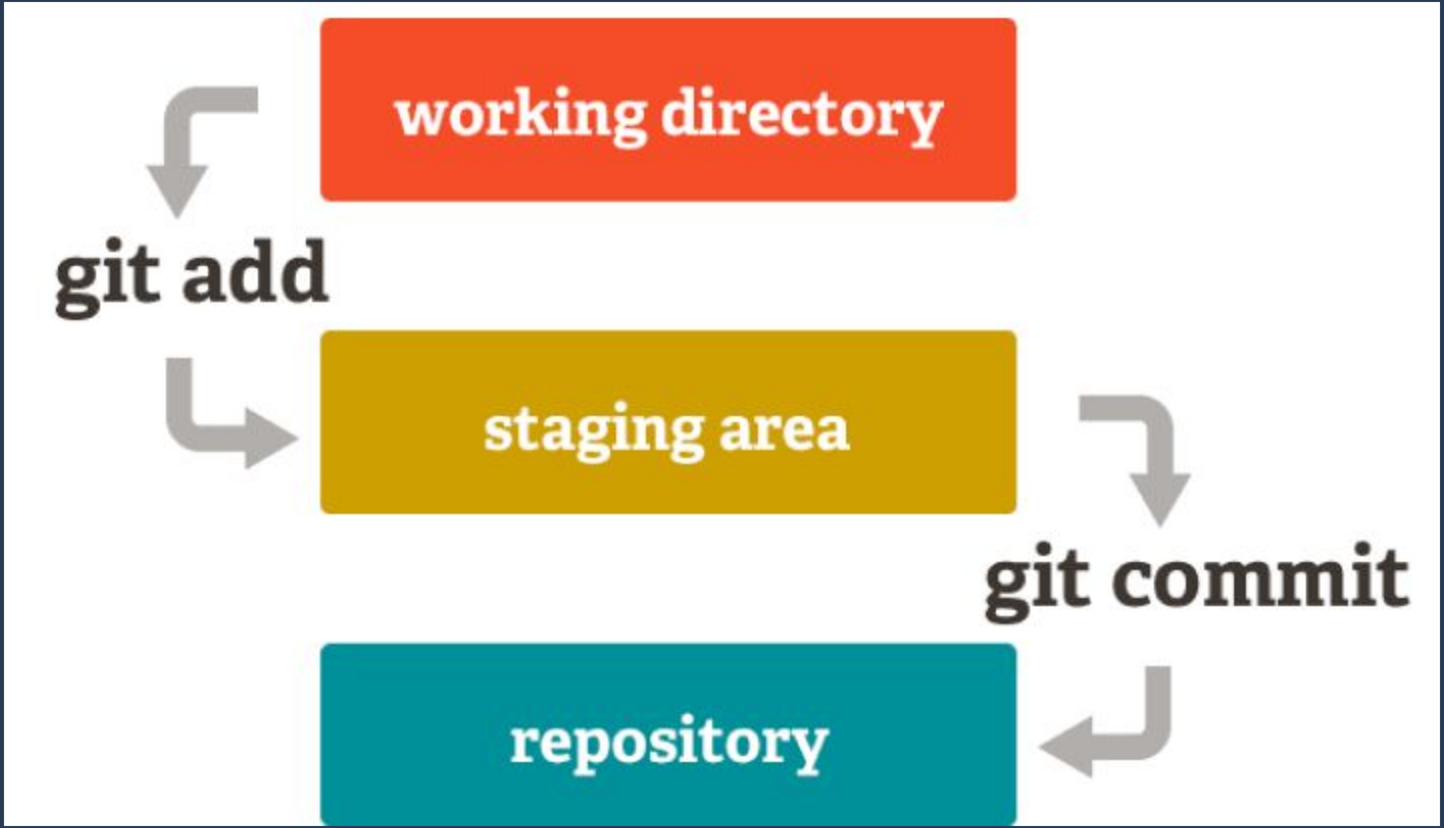
Staging Area

- We move files that we actually want to save in a commit, to the staging area using `git add` command
- Stage area is for reviewing files to prevent from committing unnecessary files.

Note: Apart from Git, most VCS doesn’t have this special intermediary step.

Git Repository

- This is the .git directory where git stores all the project information.
- This directory simply contains all the implementation details as to how git works under the hood.
- If you delete this .git directory, you’ll lose all your project history.


working directory
git add
staging area
git commit
repository

Git Commands

`git init`

- The magical command which can convert your normal directory into a Git Repository
- It creates a hidden directory called “`.git`”. This magical directory consist of objects and refs that git uses to maintain your project’s history.
- Command:
  - `git init`: Transform the current directory into a git repository
  - `git init <directory>`: Transform a directory in the current path into a git repository

`git status`

- `git status` shows the current state of your Git working directory and staging area.
- Command:
  - `git status`
  - `git status -s`

`git add`

- git add commands **adds new or changed (includes deleted) files** to the Git Staging Area.
- Command
  - `git add <filename>`: Puts that file into the staging area
  - `git add <path>`: Puts that directory or file into staging area

`git commit`

- `git commit` captures a snapshot of the project’s currently staged changes.
- Commit(Snapshot) includes lots of data such as contents, message, author, timestamp, etc.
- Command:
  - `git commit -m “Please leave your message for the commit”`

`git log`

- `git log` helps to list the commits in reverse chronological order
- Commands:-
    - `git log`
    - `git log --oneline`
    - `git log --stat`
    - `git log --since=”1 month ago”`

`git diff`

- `git diff` helps to show changes between commits and working tree
- Commands
  - `git diff`
  - `git diff --staged`
  - `git diff commit_2_id`

`git reset`

- `git reset` helps to undo repository to any particular state present in the history of the repository
- Soft Reset: Resetting repository to a given commit and adds all the changes to the staging area
  - Command: `git reset --soft`
- Hard Reset: Resetting repository to a given commit and deleting all the changes
  - Command: `git reset --hard`

Branching in Git

- `git branch` operates branching operations such as list, delete and creates branches in the repository
  - Commands:
    - `git branch`
    - `git branch feature-1`
    - `git branch -D feature-1`
- `git checkout` helps to switch between branches
  - Commands:
    - `git checkout feature-1`

Github

GitHub

- Founded in 2008, GitHub is the largest and most famous platform to host git code repositories.
- It allows code collaboration with anyone online!

GitHub

- Git != GitHub
- *You do not need GitHub to use git, but you cannot use GitHub without using git.*
- GitHub adds extra functionalities on top of git, such as
    - Great UI
    - Pull Requests
    - Documentation
    - Bug Tracking
    - Feature Requests and many more…

Git commands for Github

- `git clone`
- `git remote`
- `git push`
- `git pull`

`git clone`

- `git clone` helps to **copy** a specific repository
- Commands
    - `git clone` *`https://github.com/IMGIITRoorkee/git-workshop-2020.git`*

`git remote`

- `git remote` helps to maintain connections with other repositories.
- Remotes serve as convenient names of not-so-convenient url.
- Commands
    - `git remote -v`
    - `git remote add origin` *`https://github.com/IMGIITRoorkee/git-workshop-2020.git`*

`git push`

- `git push` uploads your local branches to the corresponding remote branches
- Commands
  - `git push origin`
  - `git push origin master`

git pull

- `git push` updates your local working branches from the corresponding remote branches
- Commands
  - `git pull origin`
  - `git pull origin master`

Hosting your webpage

Materials

- git-scm.com/doc: Official Documentation of Git
- try.github.io : To learn and practice advance git concepts
- freecodecamp.org/learn/: To learn more about HTML, CSS and other web programming tools

Thank You!