Git, GitHub, and Version Control

How to effectively code collaboratively

Jeff van Santen - Aug. 19, 2020

This tutorial will be recorded.

GitHub

A Git social network

GitHub is repository hosting system with many additional features, making it ideal for collaboration.

It allows for *organizations* (github.com/liningtonlab), which allow for sub-division into teams, and projects.

Teams are for subdividing people within an organizations. Ex.

https://github.com/orgs/liningtonlab/teams/hifan

Projects are for planning and collaborating on specific units of work. This is really just a Kanban-style list-making board which allows you to link ideas with specific “issues” or “pull-requests” on GitHub.

GitHub also features unlimited free private repositories!

Understanding Git

Background

Git is an open-source version control system (of which there are many), originally created by Linus Torvalds - creator of the Linux operating system.

Microsoft isn't evil, they just make really crappy operating systems.

Understanding Git

Background

Git is an open-source version control system (of which there are many), originally created by Linus Torvalds - creator of the Linux operating system.

Microsoft isn't evil, they just make really crappy operating systems.

Understanding Git

Background

Git is built on complex graph algorithms that take snapshots of your files and make it simple to compare changes.

“[Git] is amazingly fast, it’s very efficient with large projects, and it has an incredible branching system for non-linear development.”

See the Git - Book for basics on usage:

https://git-scm.com/book/en/v2

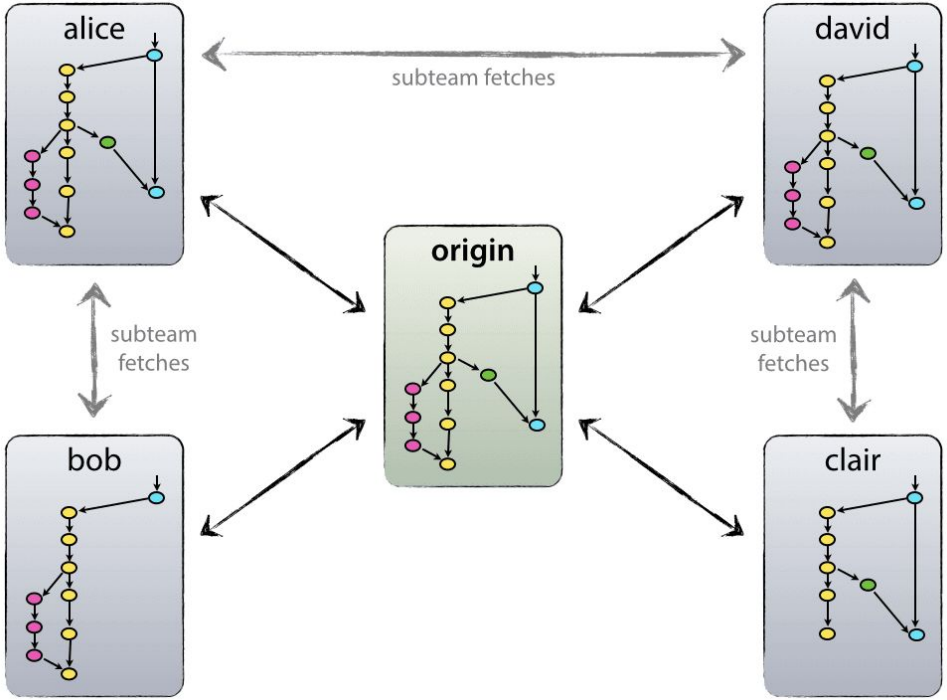
Understanding Git

Background

```
➜  ~ git
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
   clone     Clone a repository into a new directory
   init      Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
   add       Add file contents to the index
   mv        Move or rename a file, a directory, or a symlink
   restore   Restore working tree files
   rm        Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
   bisect    Use binary search to find the commit that introduced a bug
   diff      Show changes between commits, commit and working tree, etc
   grep      Print lines matching a pattern
   log       Show commit logs
   show      Show various types of objects
   status    Show the working tree status

grow, mark and tweak your common history
   branch    List, create, or delete branches
   commit    Record changes to the repository
   merge     Join two or more development histories together
   rebase    Reapply commits on top of another base tip
   reset     Reset current HEAD to the specified state
   switch    Switch branches
   tag       Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
   fetch     Download objects and refs from another repository
   pull      Fetch from and integrate with another repository or a local branch
   push      Update remote refs along with associated objects
```

Understanding Git

The Real Benefits

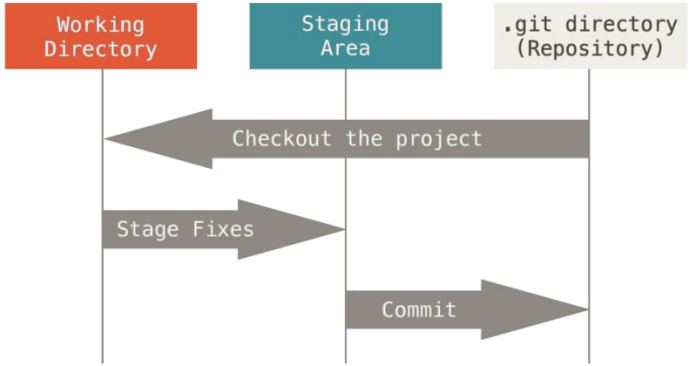


Understanding Git

The Basics

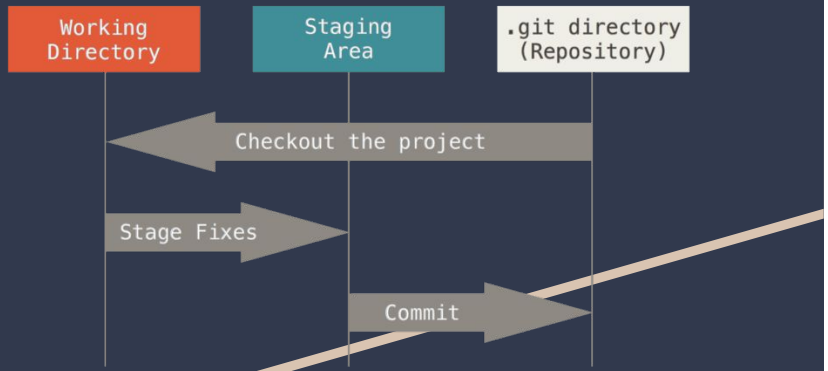
Some details for the Git Book:

1. Snapshots, Not Differences
2. Nearly Every Operation Is Local
3. Git Has Integrity
4. Git Generally Only Adds Data
5. The Three States



Understanding Git

The Basics



The Three States - basic workflow

1. You modify the files in your working directory.
2. You selectively stage just those changes you want to be part of your next commit, which adds only those changes to the staging area.
3. You do a commit, which takes the files as they are in the staging area and stores that snapshot permanently to your Git directory.

THE MISSING FOURTH STAGE

You are NOT collaborating yet, just using Git.

Git actions are all LOCAL, and nothing changes until your PUSH your changes to the repository hosting system - i.e. **GitHub**.

Git LFS

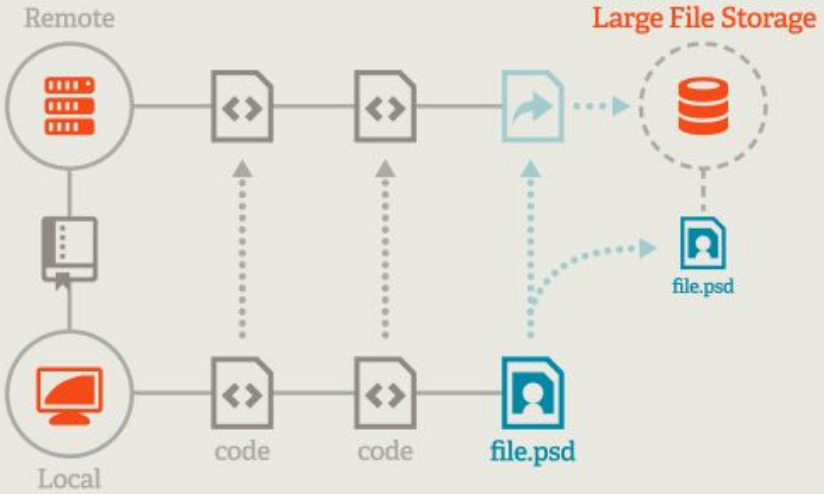
Git Large File Storage

Docs Downloads Source

An open source Git extension for versioning large files

Git Large File Storage (LFS) replaces large files such as audio samples, videos, datasets, and graphics with text pointers inside Git, while storing the file contents on a remote server like GitHub.com or GitHub Enterprise.

Download v2.11.0 (Mac)



Useful Software for Code + Git

VS Code

Text editor with sensible defaults, lots of useful extensions. Basically a full IDE. Also has support for Jupyter notebooks.

GitKraken

Git client with featureful GUI and integration with GitHub + Trello.

https://docs.anaconda.com/anaconda/user-guide/tasks/integration/python-vsc/
https://blog.axosoft.com/gitkraken-glo-boards-support-github-actions/

DRINK RESPONSIBLY.

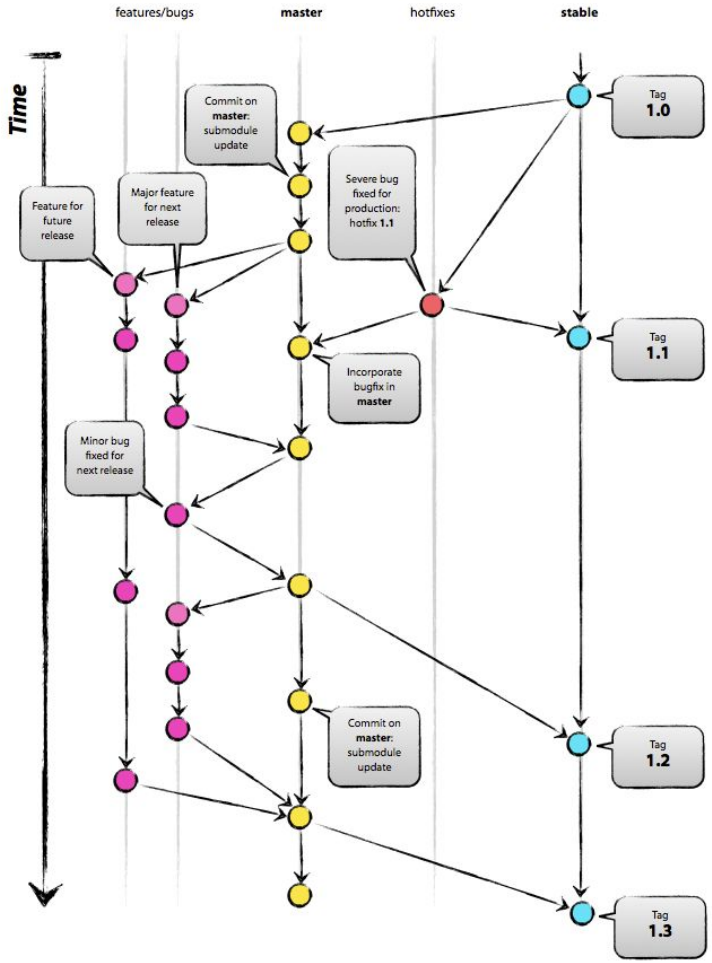
KRAKEN COFFEE METER

THIS MESSAGE BROUGHT TO YOU BY GITKRAKEN.COM

GitHub Best Practices

1. Don’t git push straight to master
2. Don’t commit code as an unrecognized author
3. Define code owners for faster code reviews
4. Don’t leak secrets into source control (SSH keys, API keys, Passwords…)
5. Don’t commit dependencies into source control
6. Don’t commit local config files into source control
7. Create a meaningful gitignore file
8. Archive dead repositories
9. Specify package versions
10. Use a branch naming convention
11. Delete stale branches
12. Remove inactive GitHub members
13. Enable security alerts
14. Use **Coding Standards!** (naming and documentation conventions, linter, code formatter, etc.)

https://www.datree.io/resources/github-best-practices


features/bugs
master
hotfixes
stable
Time
Commit on master: submodule update
Tag 1.0
Severe bug fixed for production: hotfix 1.1
Feature for future release
Major feature for next release
Tag 1.1
Incorporate bugfix in master
Minor bug fixed for next release
Commit on master: submodule update
Tag 1.2
Tag 1.3

Linington Lab Code Repo Guide

I will formalize some conventions that the Linington Lab should follow with regards to repository usage and code format.

https://github.com/liningtonlab/code_style_guide

Onto a Practical Example!