Introducción a Git

May 2018

Agenda

- Version Control, Git & GitHub
- Terminology
- GitHub Flow
- GitHub Features
- Exercises

Definitions

Version Control (aka Revision Control or Source Control)

"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."

Pro Git (2nd Edition)

Version Control Benefits

What do you get with Version Control?

- Add files to a version control system and remove them
- Track changes
- Revert files to a previous state (rollback)
- Label and Tag
- Compare changes and search through history
- Track file permissions
- And much more...

Git

- Git is a version control system for tracking changes in computer files and coordinating work on those files among multiple people.
- It is primarily used for source code management in software development, but it can be used to keep track of changes in any set of files.
- Git is distributed by nature. Users can push changes to one another if they wish or they can push to a single. common repository of their choice. The flexibility is there.
- Originally created by Linus Torvalds in 2005.

GitHub

- GitHub is a web-based hosting service for version control using git.
- It offers all of the distributed version control and source code management (SCM) functionality of Git as well as adding its own features.
- It provides access control and several collaboration features such as bug tracking, feature requests, task management, and wikis for every project.

Terminology

Repository

- Repository (or repo) is a storage space for your projects. Can be a directory or space on GitHub.
- There's always a special hidden .git subdirectory.
- Local repo is on your machine. Remote repo is elsewhere.

Terminology

Clone

- Copies an existing Git repository from a remote source.
- Brings down all files and history to your machine.
- Creates a remote connection called origin that points back to the original repository.

Terminology

Add and Commit

- Use *git add* to add changes in the working directory to the staging area. Git will include these changes in the next commit.
- Use *git commit* to commit the staged snapshot to history.
- Use git *commit -am "my message"* to add and commit in one command.

Terminology

Push

- Use *git push* to transfer commits from your repository to a remote repository.
- If your repo's history has diverged from the remote repo's history - Git will prevent the push (non-fast-forward).
- You could use --force to force push but it will overwrite history/commits in the remote repo so be careful!!!

Terminology

Branch

- A branch is an isolated copy of your working environment and history.
- Commits are recorded in the history for the current branch.
- Create and delete branches at will.
- You will always work on a branch. Master is a branch too!

Terminology

Fetch

- Use *git fetch* to grab commits from the remote repository
- These are stored as remote branches
- You can review them and run *git merge* to merge in to your repository

Terminology

Merge

- A way of taking one branch and merging its history/commits into another branch.
- If Git is unable to merge two files together it will stop and you will have to manually resolve a merge conflict.

Terminology

Pull

- Rolls *git fetch* and *git merge* in to one command.

Terminology

Pull Request

- In GitHub - a way to notify other users that there is a branch that is ready for a review.
- A Pull Request provides collaborators with the ability to discuss and make comments about changes made.
- Usually the branch is then merged in to another (master/central) branch and is discarded.

Terminology

Upstream

- The original remote repository from which you cloned is often referred to as upstream.

GitHub Flow

- **Understanding the GitHub Flow:**
https://guides.github.com/introduction/flow/

- **General: Learn Git Branching:**
https://learngitbranching.js.org

GitHub Features

- Repository Settings
- Issues
- Releases and Tags
- Collaborators
- Pull Requests
- Wiki
- Insights
- Integrations

Exercises

Exercise – Create Github account

- Go to https://github.com and follow the instructions to create a new account

Exercise – Create a repo on Github

- Create a repo on the GitHub User Interface
- Add a default README.md file
- Configure a .gitignore file
- Configure a License file (an Apache 2.0 license, for example)

Exercise - Review your repo details on GitHub

- Click on branches to see the current branches (master for now).
- Notice the tabs at the top.
- Click on the ”Clone or Download” button and see that a link is shown.

Exercise – Edit a file on GitHub

- Edit the README file but don’t commit it yet.
- Click on Preview.
- Commit the file on the default master branch, changing the commit message and description.
- Click on the History button to see previous changes.
- We’ll review the History functionality later.

Exercise – Install the Git CLI

- Go to https://git-scm.com and follow the instructions to install the Git Command Line Interface.

- Click on https://git-scm.com/downloads/guis to see a list of graphical interfaces for Git:
  - Review https://desktop.github.com (free)
  - Review https://www.sourcetreeapp.com (free)
  - Review https://www.git-tower.com

Exercise – Configure the Git CLI

Run:

git config --global user.name “yourname”

git config --global user.email “your@email.address”

git config --list

Exercise – Clone a repo to your local machine

Find the repo's URL and copy it.

Run:

git clone https://github.com/yourname/yourreponame.git

cd yourreponame

git status

Exercise – Add, Commit and Push

Edit README.md file

Run:

git status

git commit -am “My new change”

git status

git push origin master

git config --global push.default simple (to be able to type only git push instead of git push origin master)

git push

Exercise – Add new files

Create and edit a new file (test.txt)
Run:
git add --all (or git add test.txt)
git status
git commit -am “Added new file test.txt”
git status
git rm test.txt
git commit -m “Deleted file test.txt”
git status

Exercise – Pull changes from the remote repo

Go to GitHub and edit the README.md file again AND commit the change.

Go to your machine and review the contents of README.md.

Run:

git pull origin master

Exercise – The GitHub Flow I

Review https://guides.github.com/introduction/flow
Go to your GitHub repo and add a new branch (call it feature).
Run:
git fetch
git checkout feature
Create and edit a new file call testfile.txt
Run:
git add --all
git commit -m "Added testfile.txt"
git status
git push (equivalent to git push origin feature)
Go to your GitHub repo and see the changes in your new branch.

Exercise – The GitHub Flow II

On GitHub, press the “New pull request” button to merge the changes.

Review the screen, add a comment like “Added new file for the new feature” and click on “Create pull request”.

Go to the “Pull requests” tab.

Leave a comment on your own PR (Pull Request).

Supposing all collaborators are OK with the change, we should click on the “Merge pull request” button. Confirm the merge.

Finally, remove the branch by clicking on “Delete branch”.

Exercise – Merge conflicts

On GitHub modify and commit README.md

On your machine, run git pull (switch to master first by running git branch master if necessary).

On your machine, modify README.md by inserting a line at the top (something like “added on my machine”) and then run:

git commit -am “Added new line at the top of README.md”

On GitHub, modify the current README.md file by adding a different text at the top (like “added on GitHub”).

On GitHub, commit the changes.

On your machine, run:

git status (see we are ahead)

git pull origin master

There is a CONFLICT.

Exercise – Merge conflict resolution

Use a merge tool, like KDiff3 (see online documentation for setup).

Run:

git mergetool

Merge on KDiff3 and save.

Run:

git status

rm README.md.orig

git commit -am "Resolverd merge conflict"

git push

Git Summary

- Git is a distributed version control system. Every Git repo is standalone.
- Every user gets their own full copy of the files.
- Everyone always works in a branch. A branch is a copy of the working directory, the staging area and project history. A repository can contain one or more branches and you can switch between branches.
- Use clone to create a copy of the remote repository.
- Use add and commit to persist changes locally.
- Use push to send changes to the remote repository.

“Talk is cheap. Show me the code.”

Linus Torvalds