Udemy Course - GitHub Ultimate: Master Git and GitHub

16 October 2018 15:37

Objectives 1

- Gain understanding of Git & commands
- Use GitHub as a hosting service with its features.

Objectives 2

- Git core concepts
- Installation + detailed installation as bonus.
- Fundamental Git commands
  - Create Git project with standard operations for a developper
- Advanced topics
  - Comparing
  - Branching
  - Merging
  - Time travel
- GitHub as remote repository
  - Working with repos
  - Join and contribute to other projects
  - Sharing code
  - Issue tracking with Git issues
  - Grouping repos into organisations

Git is a decentralised, distributed version control system.

The Basics

17 October 2018 12:33

Overview:

- Create new repo or join existing project
- Informational commands
- Basic Git workflow (commits)
- File operations
- Exclude unwanted files from repo
- Undoing mistakes

Initialisation

- cd to desired path folder
- Create new empty repo: *git init RepoName*

Git States

- Working dir
  - Contains all files and folders
- Repository
  - All commits and saved changes
- Staging area
  - Middle state

New repo

1. Create new repo on GitHub
   a. Get remote from GH
2. Move to folder in local repo in Bash
3. Paste 'git remote add origin …'
   a. origin is the name of the remote
   b. By convention first remote is called origin
4. Local repo is now linked to GH

Pushing changes to GH

- Command: git push -u origin master --tags
  - sets up a tracking branch relationship between 'master' branch on local and remote repos 'origin'
  - Also pushes tags to GH
- For next changes: -u not needed

Git Commands

17 October 2018 13:10

git init *RepoName*		Create new empty repo
git status		
git add *fileName*		Pushes file to staging area
git commit -m "*message here*"		Commit files in staging area along with a message
Git show		Get info of last commit

CREATE

Clone an existing repository
```
$ git clone ssh://user@domain.com/repo.git
```
Create a new local repository
```
$ git init
```

LOCAL CHANGES

Changed files in your working directory
```
$ git status
```
Changes to tracked files
```
$ git diff
```
Add all current changes to the next commit
```
$ git add .
```
Add some changes in <file> to the next commit
```
$ git add -p <file>
```
Commit all local changes in tracked files
```
$ git commit -a
```
Commit previously staged changes
```
$ git commit
```
Change the last commit
Don't amend published commits!
```
$ git commit --amend
```

COMMIT HISTORY

Show all commits, starting with newest
```
$ git log
```
Show changes over time for a specific file
```
$ git log -p <file>
```
Who changed what and when in <file>
```
$ git blame <file>
```

BRANCHES & TAGS

List all existing branches
```
$ git branch -av
```
Switch HEAD branch
```
$ git checkout <branch>
```
Create a new branch based on your current HEAD
```
$ git branch <new-branch>
```
Create a new tracking branch based on a remote branch
```
$ git checkout --track <remote/branch>
```
Delete a local branch
```
$ git branch -d <branch>
```
Mark the current commit with a tag
```
$ git tag <tag-name>
```

UPDATE & PUBLISH

List all currently configured remotes
```
$ git remote -v
```
Show information about a remote
```
$ git remote show <remote>
```
Add new remote repository, named <remote>
```
$ git remote add <shortname> <url>
```
Download all changes from <remote>, but don't integrate into HEAD
```
$ git fetch <remote>
```
Download changes and directly merge/integrate into HEAD
```
$ git pull <remote> <branch>
```
Publish local changes on a remote
```
$ git push <remote> <branch>
```
Delete a branch on the remote
```
$ git branch -dr <remote/branch>
```
Publish your tags
```
$ git push --tags
```

MERGE & REBASE

Merge <branch> into your current HEAD
```
$ git merge <branch>
```
Rebase your current HEAD onto <branch>
Don't rebase published commits!
```
$ git rebase <branch>
```
Abort a rebase
```
$ git rebase --abort
```
Continue a rebase after resolving conflicts
```
$ git rebase --continue
```
Use your configured merge tool to solve conflicts
```
$ git mergetool
```
Use your editor to manually solve conflicts and (after resolving) mark file as resolved
```
$ git add <resolved-file>
$ git rm <resolved-file>
```

UNDO

Discard all local changes in your working directory
```
$ git reset --hard HEAD
```
Discard local changes in a specific file
```
$ git checkout HEAD <file>
```
Revert a commit (by producing a new commit with contrary changes)
```
$ git revert <commit>
```
Reset your HEAD pointer to a previous commit
...and discard all changes since then
```
$ git reset --hard <commit>
```
...and preserve all changes as unstaged changes
```
$ git reset <commit>
```
...and preserve uncommitted local changes
```
$ git reset --keep <commit>
```

From: https://www.git-tower.com/blog/git-cheat-sheet