Git & GitHub

by @kinbiko
on 2018-01-31

» Outline

* High Level Overview
* Git
* Learn by doing #1
* GitHub
* Learn by doing #2
* Closing remarks

» Why use Git and GitHub

* FINAL/DRAFT/v1.2 🚫
* Work is shared (distributed)
* Create a portfolio

» Difference between Git and GitHub

* Git
  * Versioning your files
  * Command-line program
* GitHub
  * Collaboration
  * Free web service (`github.com`)

» Git: Versioning

* Multiple versions, one file
* Go back to any version
* Any file type
* Plain text files

» GitHub: Collaboration

* Explore other projects
* Ask open source maintainers for help*
* Work on the same files in parallel
* Review each other's work
* Project management tools and issue tracking

» Git

* Command line
* GUI

» It's Git, innit?

* `git init`
* Directory => repository
* Used for new projects only
* GitHub makes this a lot easier

» Basic workflow

* `add`: marks the files you want for your next version
* `commit`: creates a version for the added files

» git add

* `git add <file1> <file2> ...`
* `git add .`
* Adding files partially is possible (LAAEFTR), but prefer smaller changes

» git commit

* `git commit -m "<what changed?>"`
* Remember the “-m”, or learn Vim basics.
* Prefer many commits to large commits

» Commit messages

* Commit messages require discipline
* Generally short messages - as your commits should be small
* But complete descriptions are better than being consistently short

» git status

* staged
* not staged
* New files (untracked)

» Learn by doing #1

* Create a folder called `repos`
* Create a folder inside `repos` called `my-first-repo`
* `cd my-first-repo`
* `git init`
* Create and edit a `README.md` file
* `git add` & `git commit` until confident
* `git status` is your friend

» Create a GitHub account

* `github.com/join`
* Tell me your username

» Explore

* Your favourite OSS projects are probably here.
* Your favourite developers are probably here.

» Public repos for everyone

* `github.com/new`
* Use kebab-cased names
* `README.md` and `LICENSE`
* Code loves `.gitignore`

» learning-github

* `clone`: get repo from somewhere else (GitHub)
* Only owners can push
* Anyone can clone & pull

» Push/Pull

* `git push` puts local work on GitHub
* `git pull` puts GitHub work on your local machine

» Forks

* Forks copy a project to your GitHub account
* The fork is YOURS to push and commit to
* Use the fork to contribute to other projects with pull requests

» Pull Requests (PRs)

* “Please pull my changes”
* Read the `CONTRIBUTING.md` before raising a PR
* Add a description, assign reviewers, look over your code in the diff view

» Working in teams

* 90% of the way there!
* What about updates to the original repo?
* Solve with branches!

» Branches

* Easy to create in GitHub
* `master` is holy
* One branch per task

» Checking out branches

* Create branch in GitHub
* `git pull` to update
* `git checkout <branch-name>`

» Get new commits by merging

* `git pull` to update
* `git checkout my-branch`
* To update:
  * `git pull`
  * `git merge origin/master` (usually)
* (Psst! `origin == GitHub`)

» Merge conflicts

* When two commits collide
* Creates new commit
* ```
  <<<<<<<
  <one commit's contents>
  =======
  <the other commit's contents>
  >>>>>>>
  ```
  ```

» Learn by doing #2

* Fork `bath-ml/learning-git`
* Read and understand the `CONTRIBUTING` file.
* Answer (using PRs) the questions in `README.md`
* Review each other's PRs

» Maybe don't use Git for...

* Large amounts of data
* Binary files that changes a lot
* Secrets/tokens/api keys

» Git is also used for

* CI/CD
* Hiring process
* Dotfiles

» GUIs for Git

* Git Kraken
* Sourcetree
* GitHub Desktop

» Next steps:

* GitHub bathroom tiling
* CodeHub Bristol
* Add me on social media (I'll endorse you for Git on LinkedIn)
* Contribute to Bath-ML. Blog posts and PRs welcome!