

**Institute** of  
**Data**

---

2020



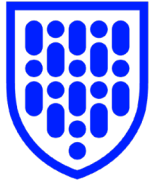
# Data Science and AI

## GIT Workflow

---

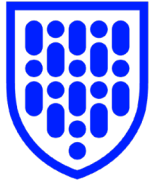
## Fork & Branch

---

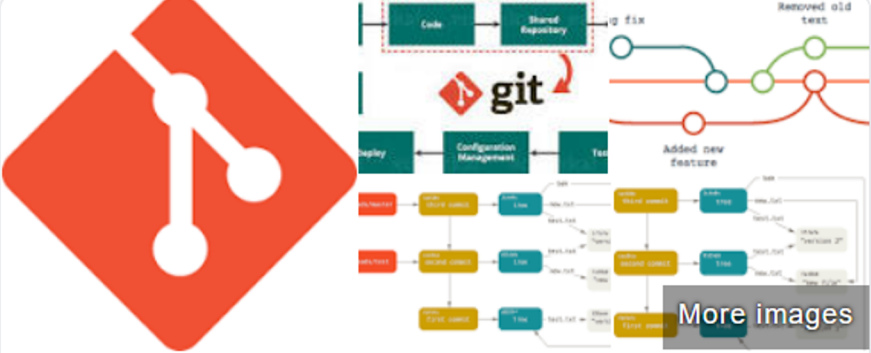


# Agenda

- What's GIT?
- Create a github account
- Install client
- Fork a GitHub repository
- Create a local clone
- Contribute to the project
- Pushing changes to forked repo
- Fork and add upstream link
- Why pushing is not encouraged to original repo?
- Advantage of Fork & Branch workflow
- Working on a Branch
- Guidelines
- Push changes from feature branch
- Open a PULL request
- Cleaning up after a Merged Pull Request
- What's the progress so far?
- What's happening in a healthy project?
- Summary



# What's GIT?



The image shows the Git logo, a red diamond with a white branching diagram, and a complex workflow diagram. The workflow diagram illustrates various Git operations such as cloning, committing, pushing, pulling, merging, and branching, with labels like 'Code', 'Shared Repository', 'git', 'Configuration Management', 'Test', 'Added new feature', and 'Removed old text'.

## Git

System software

Git is a distributed version-control system for tracking changes in source code during software development. It is designed for coordinating work among programmers, but it can be used to track changes in any set of files. Its goals include speed, data integrity, and support for distributed, non-linear workflows. [Wikipedia](#)

**License:** GPLv2, LGPLv2.1, and others

**Developer(s):** Junio Hamano and others

**Stable release:** 2.23.0 / 16 August 2019; 35 days ago

**Operating system:** POSIX: Linux, Windows, macOS

**Original author:** Linus Torvalds

**Initial release:** 7 April 2005; 14 years ago



# Create a Github account

The screenshot shows the GitHub homepage with a dark background. On the left, the text "Built for developers" is prominently displayed in white, followed by a paragraph describing GitHub as a development platform. On the right, a white sign-up form is overlaid. The form contains three input fields: "Username", "Email", and "Password". Below the "Password" field, there is a note about password requirements: "Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter. [Learn more.](#)". A large green button labeled "Sign up for GitHub" is positioned below the form. At the bottom of the form, a small line of text states: "By clicking 'Sign up for GitHub', you agree to our [Terms of Service](#) and [Privacy Statement](#). We'll occasionally send you account related emails."

Why GitHub? ▾ Enterprise Explore ▾ Marketplace Pricing ▾

Search GitHub / Sign in Sign up

## Built for developers

GitHub is a development platform inspired by the way you work. From open source to business, you can host and review code, manage projects, and build software alongside 40 million developers.

Username

Email

Password

Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter. [Learn more.](#)

[Sign up for GitHub](#)

By clicking "Sign up for GitHub", you agree to our [Terms of Service](#) and [Privacy Statement](#). We'll occasionally send you account related emails.



# Install GIT client

- Windows
  - <https://git-scm.com/download/gui/windows>
- MAC
  - <https://www.atlassian.com/git/tutorials/install-git#mac-os-x>



# Fork a GitHub repository

- Fork a repository:
  - login to your GitHub account
  - find the GitHub repository you need to work with
    - <https://github.com/DSIA-Education/<repo name>.git>
  - click the Fork button on the upper right-hand side
    - <https://github.com/<git user name>/<repo name>.git>



# Create a local clone

- Clone the repository to your local system
  - to enable making changes to the forked repository in your GitHub account
    - make a copy of a repository in your personal GitHub account
  - get the URL of the forked repository & clone
    - `git clone https://github.com/<git user name>/<repo name>.git`
- Clone your forked repository, not the original
  - git will copy the repository, both contents and commit history, to your system
  - git will add a Git remote called origin that points back to the forked repository in your GitHub account





# Contribute to the project

- If you were only interested in forking the project and not contributing back to the original project, you could stop here



# Pushing changes to forked repo

- Git added a git remote named “origin” to the clone of the git repository on your system
- Allows you to push changes back up to the forked repository in your GitHub account using git commit (to add commits locally) & git push
  - `git commit -m <your commit message>`
  - `git push origin master`
- git remote is necessary for any repository
  - to push changes to & to pull changes from



# Fork and add upstream link

- Add a git remote pointing back to the original repository and name it “upstream”
  - Let's say you forked <repo name> and cloned it on your system.
    - `cd <repo name>`
    - `git remote add upstream https://github.com/DSIA-Education/<repo name>.git`



# Why pushing is not encouraged to original repo?

- Fails as there is no permission to push changes directly to the repository
- It is not a good idea as other people might be working on the project
- Will be difficult to keep track of everyone's changes



# Advantage of Fork & Branch workflow

- Facilitates
  - multiple users
    - to make changes to a repository at the same time
    - issue pull requests (request-for-merge) for merging their commit(s) to the original repository



# Working on a Branch

- What to do on your local git repository?
  - create and checkout a feature branch
    - create a new branch and check it out:
      - `git checkout -b feature`
  - make changes to the files
  - commit your changes to the branch
- Can create multiple branches, if you so desire
  - advise – stick to a single feature branch at a time



# List local & remote branches

- Fetch Remote Tracking Branches
  - `git fetch` - update your remote-tracking branches under `refs/remotes/<remote_name>/`
- List all branches on local and all remote repositories
  - `git branch -a`
- Git List Remote Branches
  - `ls-remote` - to list remote repositories without updating your remote-tracking branches
  - `git ls-remote`
- If repository is connected with multiple remotes, use following command syntax
  - `git ls-remote --heads <remote_name>`



# Guidelines

- Projects have specific requirements around branch names for pull requests
  - let's use the name “feature”
- Once the new branch is created and checked out
  - you can make the necessary changes in this branch to implement the specific “feature”
- Rule of thumb - limit a branch to one logical change
  - definition of one logical change will vary from project to project and developer to developer
  - basic idea is that you should make necessary changes to implement one specific module or capstone.
- As you make changes to the files in the branch
  - commit changes
  - build changeset with git add
  - commit changes using git commit
  - projects may not want a bunch of commits in a single pull request
    - git rebase to “squash the commit history”





# Push changes from feature branch

- Changes necessary to implement the specific feature or module (the one “logical change”) is completed
- Changes committed to local repository
- Push changes to GitHub
  - `git push origin feature`
    - you are pushing changes in “feature” branch to the origin remote



# Open a PULL request

- Login to your Github account
  - GitHub will prompt you to create a pull request
- Pull requests should contain – as is relevant:
  - module number
  - exercise number & description
  - mini project number & description
  - capstone phase descriptions
- Maintainers of the original project:
  - uses the pull request to pull your changes across to the main repository
  - if changes are approved, merge them into the main repository
  - else, will send info/action requests



# Cleaning up after a Merged Pull Request

- Once maintainers accept your changes and merge them into the main repository
  - you should update your local clone by using
    - `git pull upstream master`
      - pulls changes from original repository's (indicated by upstream) master branch (indicated by master in the command) to your local cloned repository
- One of the commits in the commit history will be the commit that merged your feature branch,
  - after you `git pull`, your local repository's master branch will have your feature branch's changes committed.
    - you can delete the feature branch (because the changes are already in the master branch):
      - `git branch -d feature`
  - update the master branch in your forked repository:
    - `git push origin master`
  - push the deletion of the feature branch to your GitHub repository
    - `git push --delete origin feature`



# What's the progress so far?

- You have successfully created a feature branch
- Made some changes
- Committed those changes to your repository
- Pushed them to GitHub
- Opened a pull request
- Had your changes merged by the maintainers
- Cleaned up



# What's happening in a healthy project?

- Your forked repository does not automatically stay in sync with the original repository;
  - Keep your fork in sync with original repo
    - `git pull upstream master`
      - pulls changes from the original repository (the one pointed to by the upstream Git remote)
    - `git push origin master`
      - pushes them to your forked repository (the one pointed to by the origin remote)
- Multiple contributors are:
  - forking the repository
  - cloning it
  - creating feature branches
  - committing changes &
  - submitting pull requests



# Summary

- Fork a GitHub repository
  - go to `https://github.com/DSIA-Education/<repo name>.git`
  - fork the repo & you will see `https://github.com/<git user name>/<repo name>.git`
- Clone the forked repository to your local system
  - `git clone https://github.com/<git user name>/<repo name>.git`
- Add a Git remote for the original repository
  - `cd <repo name>`
  - `git remote add upstream https://github.com/DSIA-Education//<repo name>.git`
- Create a feature branch to place your changes
  - `git checkout -b feature`
- Work on your changes to the new branch
  - regular work – add, commit & push



# Summary...continued

- Commit the changes to the branch
  - `git commit -m "your message"`
- Push the branch to GitHub
  - `git push origin feature`
- Open a pull request from the new branch to the original repo
  - Login to your Github account
  - GitHub will prompt you to create a pull request



# Summary...continued

- Clean up after your pull request is merged
  - `git checkout master`
  - `git pull upstream master`
  - `git branch -d feature`
  - `git push origin master`
  - `git push --delete origin feature`
- Keep your fork in sync
  - `git pull upstream master`
  - `git push origin master`



# Questions!