Rakamin Academy

id/x partners

Project Based Internship

Git

Introduction about Git

Daftar Isi

Git Overview

Figure 1: Git and Github

Git is a widely used environment among developers and data scientists for effectively managing version control of source code files and projects, as well as facilitating collaboration with others. Understanding version control is essential when discussing Git. A version control system enables the tracking of document changes, making it easy to recover older versions and promoting smooth collaboration.

Imagine having a shopping list and wanting your roommates to contribute and add items. Without version control, it becomes messy. With version control, you maintain clarity about the needed items after everyone's input. Git, distributed under the GNU General Public License, is open source software. It's a distributed version control system, allowing anyone globally to have a copy of your project on their computer. After making changes, they can sync their version with a remote server to share it.

Git's distributed nature is a key factor in its popularity, though there are other version control systems. These systems extend beyond code to images, documents,

and various file types. While Git can be used via command line, GitHub is a leading web-hosted service for Git repositories, with alternatives like GitLab, BitBucket, and Beanstalk.

Understanding some basic terms is important. SSH protocol ensures secure remote login. Repositories hold project folders for version control. Forking creates a copy of a repository. Pull requests seek review and approval before finalizing changes. A working directory contains files associated with a Git repository.

Essential Git commands include:

- Initializing a repository: "`git init`" (locally).
- Moving changes to staging: "`git add`".
- Checking status: "`git status`".
- Committing changes: "`git commit`".
- Undoing changes: "`git reset`".
- Viewing changes: "`git log`".
- Creating isolated environments: "`git branch`".
- Changing branches: "`git checkout`".
- Merging changes: "`git merge`".

To master Git and collaborate globally, it's crucial to learn these commands. GitHub offers excellent resources, like cheat sheets and tutorials at try.github.io. In upcoming modules, we'll provide a crash course on setting up your local environment and initiating projects.

Git History

In the early 2000s, Linux development relied on BitKeeper, a free system. However, in 2005, BitKeeper switched to a paid model, causing issues for Linux developers. Linus Torvalds led a team to create a replacement – Git. Defined by a small group, it supported non-linear development, distributed collaboration, and compatibility. It efficiently managed large projects, ensured identical code updates, and enabled various integration strategies. Git's uniqueness lies in its distributed model. It was designed by Torvalds in 2005 for Linux kernel distribution. Serving as a collaboration hub, Git's local copies allow agile development. Unlike central systems, Git's branches permit simultaneous work and continuous integration. Git's administration offers access-level control. It can be used locally or via GitHub, an online hosting service. IBM Cloud and GitHub use Git repositories. GitHub, owned by Microsoft, provides free, professional, and enterprise accounts, boasting millions of repositories. A repository stores documents, including source code, tracking versions. GitLab, a DevOps platform, grants access to Git repositories. Developers collaborate, branch, merge, and streamline testing with built-in CI/CD in GitLab.

Github

Getting a free, individual GitHub account is a swift process. Begin by visiting the GitHub website, https://github.com. Afterward, proceed to join a free plan and choose a personal account – typically what you need. Opt for a default personal, free account setup. Confirm a linking email to GitHub. GitHub offers starting options like creating a repository, an organization, or taking an introductory course.

Remember, repositories store documents, including code, managing versions. Organizations comprise user accounts owning repositories, managed by owners with admin rights. Alternatively, skip this and dive into work. GitHub provides ample resources, including guides. A Git-based project's core is the repository housing code and related elements: a README explaining the project's purpose, a license dictating code use, and more. Repositories can be private (accessible to select accounts) or public (viewable by all).

Upon repository creation, tabs appear, and the Code tab opens. This is where source files reside; Git was initially for code but now includes various files. If you made a README and/or license, they're present. Issues help track and plan, Pull Requests enable collaboration by proposing changes for review before merging, Projects manage tasks, and Wiki, Security, Insights aid advanced users in communicating with external users. Settings allow customization, like renaming repositories and access control.

Git Branch

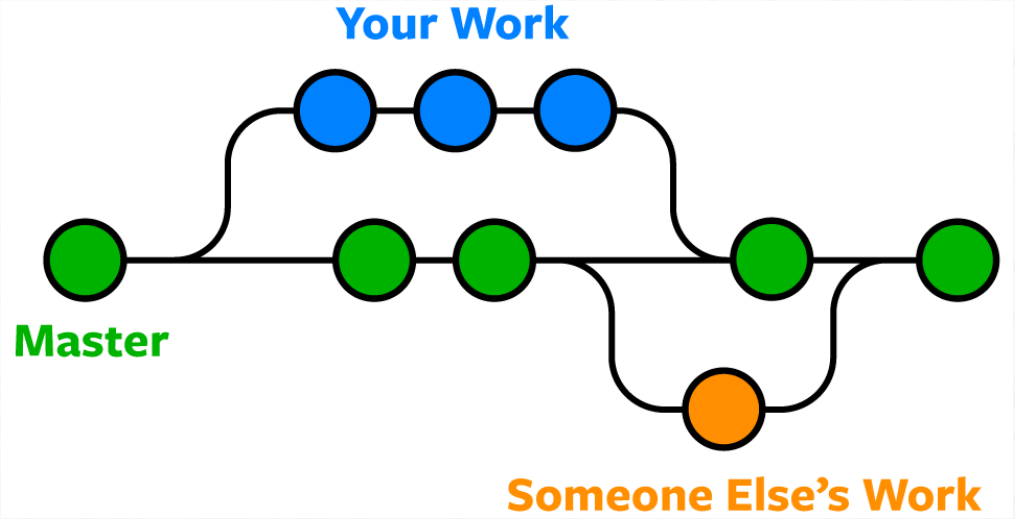

Figure 2: Example of Git Branch

Here's an example of how Git branches are useful. Let's say you need to work on a new feature for a website. You create a new branch and start working. You haven't finished your new feature, but you get a request to make a rush change that needs to go live on the site today. You switch back to the master branch, make the change, and push it live. Then you can switch back to your new feature branch and finish your work. When you're done, you merge the new feature branch into the master branch, and both the new feature and rush change are kept!

For All the Commands Below

The commands below assume you've navigated to the folder for the Git repo.

See What Branch You're On

- Run this command:
    - `git status`

List All Branches

NOTE: The current local branch will be marked with an asterisk (*).

- To see **local branches**, run this command:
    - `git branch`
- To see **remote branches**, run this command:
    - `git branch -r`
- To see **all local and remote branches**, run this command:
    - `git branch -a`

Create a New Branch

- Run this command (replacing **my-branch-name** with whatever name you want):
  - `git checkout -b my-branch-name`
- You're now ready to commit to this branch.

Switch to a Branch In Your Local Repo

- Run this command:
  - `git checkout my-branch-name`

Switch to a Branch That Came From a Remote Repo

1. To get a list of all branches from the remote, run this command:
   - **`git pull`**
2. Run this command to switch to the branch:
   - **`git checkout --track origin/my-branch-name`**

Push to a Branch

- If your local branch **does not exist** on the remote, run either of these commands:
  - `git push -u origin my-branch-name`
  - `git push -u origin HEAD`

NOTE: HEAD is a reference to the top of the current branch, so it's an easy way to push to a branch of the same name on the remote. This saves you from having to type out the exact name of the branch!

- If your local branch **already exists** on the remote, run this command:

- `git push`

Merge a Branch

1. You'll want to make sure your working tree is clean and see what branch you're on. Run this command:
   - `git status`
2. First, you must check out the branch that you want to merge another branch into (changes will be merged into this branch). If you're not already on the desired branch, run this command:
   - `git checkout master`
   - **NOTE:** Replace **master** with another branch name as needed.
3. Now you can merge another branch into the current branch. Run this command:
   - `git merge my-branch-name`
   - **NOTE:** When you merge, there may be a conflict. Refer to **Handling Merge Conflicts** (the next exercise) to learn what to do.

Delete Branches

- To delete a **remote branch**, run this command:
  - `git push origin --delete my-branch-name`
- To delete a **local branch**, run either of these commands:
  - `git branch -d my-branch-name`
  - `git branch -D my-branch-name`
- **NOTE:** The -d option only deletes the branch if it has already been merged. The -D option is a shortcut for --delete --force, which deletes the branch irrespective of its merged status.

Use Case

Background & Problem statement:

You are a data scientist in IDX Partners and working on a machine learning project. IDX Partners tech stack is using github to manage code and do version control. Please help me complete the git syntax in every step below:

- Initializing a Git Repository:
  - Write your syntax here
- Create new branch `experiment-new-algorithm`
  - Write your syntax here
- add `notebook.ipynb` and commit the changes with the commit message `Try a new ML algorithm`
  - Write your syntax here
- Merging and Pull Requests to branch `main`
  - Write your syntax here

Solution:

- Initializing a Git Repository:
  - `git init`
- Create new branch `experiment-new-algorithm`
  - `git checkout -b experiment-new-algorithm`
- add `notebook.ipynb` and commit the changes with the commit message `Try a new ML algorithm`
  - `git add notebook.ipynb`
  - `git commit -m "Try a new ML algorithm"`

- Merging and Pull Requests to branch `main`
    - `git checkout main`
    - `git merge experiment-new-algorithm`

References

https://docs.github.com/en/get-started/quickstart/set-up-git

https://www.geneatcg.com/using-git-like-a-pro/

https://www.nobledesktop.com/learn/git/git-branches