Git Knowledge

Installing GIT

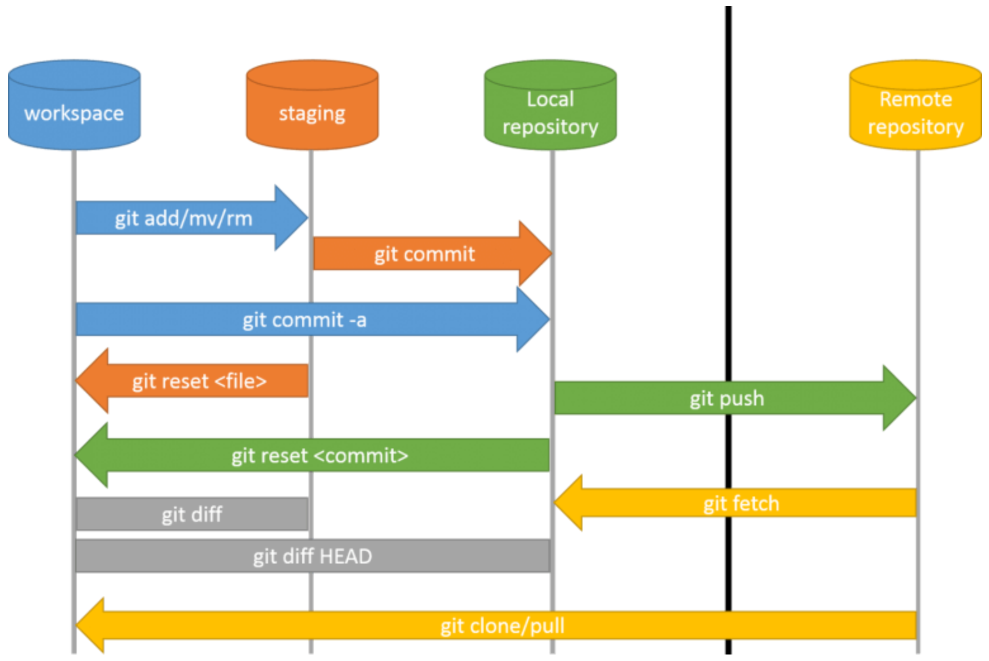
Go to the official Git website download URL:https://git-scm.com/downloads and start downloading Git for your Operating system.

What is Git

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. Git saves commit states not files, that’s why it is so fast and efficient.

Git workflow

Visual presentation of the Git workflow with their associated Git commands.



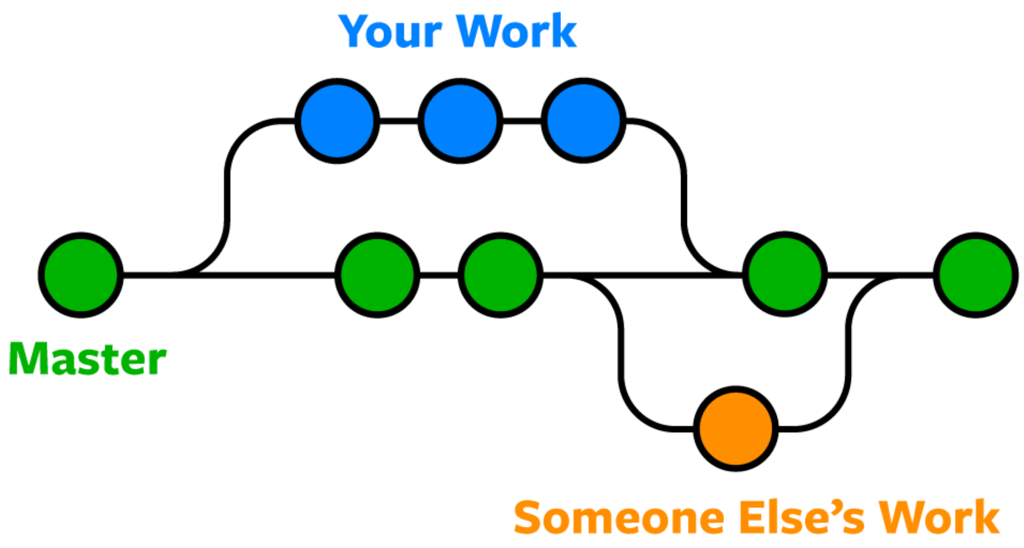
Workspace: your local workspace or project folder.
Staging: untracked files that are going to be a part of the next commit.
Local Repository: the local repository is a Git repository that is stored on your computer.
Remote Repository: a repository that all team members use to exchange their changes. In most cases, such a remote repository is stored on a code hosting service like GitHub.

Git vs Github

Git is a version control system that lets you manage and keep track of your source code history. GitHub is a cloud-based hosting service that lets you manage Git repositories

Git branches

In Git, a branch is a pointer to a specific commit. The branch pointer moves along with each new commit you make. There can exist multiple branches.

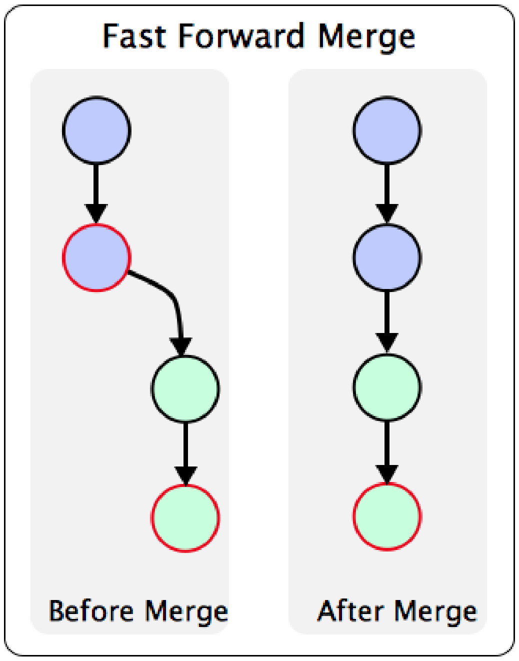


Git Commits

Git commits capture a snapshot of the project's currently staged changes. They have their own commit-ID and the user + email of the person who submitted the commit.

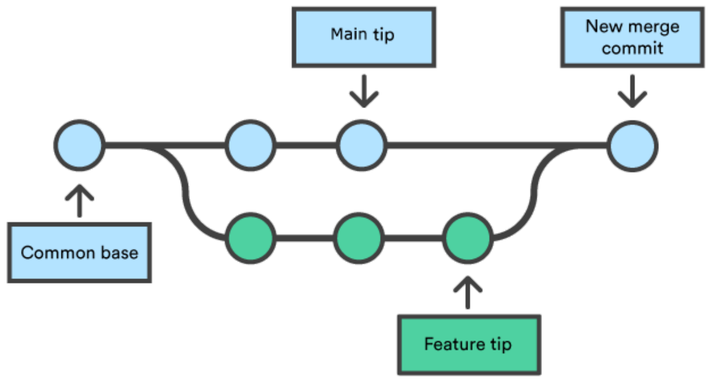
Fast Forward Merge

A fast forward merge can occur when there is a linear path from the current branch tip to the target branch. Instead of “actually” merging the branches, all Git has to do to integrate the histories is move (i.e., “fast forward”) the current branch tip up to the target branch tip.



Recursive 3-Way Merge

This can only resolve two heads using a 3-way merge algorithm. When there is more than one common ancestor that can be used for 3-way merge, it creates a merged tree of the common ancestors and uses that as the reference tree for the 3-way merge.



Merge conflicts

A merge conflict is an event that takes place when Git is unable to automatically resolve differences in code between two commits. Git can merge the changes automatically only if the commits are on different lines or branches.

Clone repositories

Used to point to an existing repo and make a clone or copy of that repository in a new directory, at another location. The original repository can be located on the local filesystem or on remote machine

Remote tracking (origin/*)

A remote-tracking-branch is first, just a branch name, like any other branch name. It points at a recent commit in your local git repository. But note that it effectively also points to the same commit in the remote repository that you cloned the commit from

Git pull request vs Git Pull

If you use git pull , you pull the changes from the remote repository into yours. If you send a pull request to another repository, you ask their maintainers to pull your changes into theirs (you more or less ask them to use a git pull from your repository).

Git origin

In Git, "origin" is a shorthand name for the remote repository that a project was originally cloned from.

Fork a repository

A GitHub fork is a copy of a repository (repo) that sits in your account rather than the account from which you forked the data from. Once you have forked a repo, you own your forked copy. This means that you can edit the contents of your forked repository without impacting the parent repo

Keep forked repository up-to-date
From GitHub: Go to your fork, click on Fetch upstream and then click on Fetch and merge to directly sync your fork with its parent repo. You may also click on the Compare button to compare the changes before merging.

Readme.md markdown

https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet

Api vs Webhook

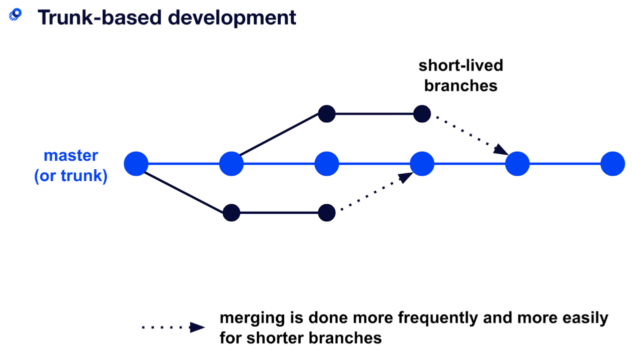
The main difference between them is that webhooks do not need to give a request to get a response, while APIs need to send a request to get a response. Webhooks let you receive, while APIs require you to retrieve.

Branching strategies: Trunk vs Git flow

Simply put, a branching strategy is something a software development team uses when interacting with a version control system for writing and managing code. As the name suggests, the branching strategy focuses on how branches are used in the development process
Dsdas

Trunk-based development (TBD)

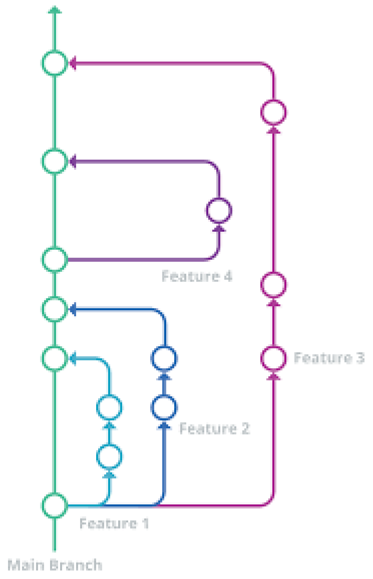
This is a branching model for software development where developers merge every new feature, bug fix, or other code change to one central branch in the version control system. This branch is called "trunk", "mainline", or in Git, the "master branch".



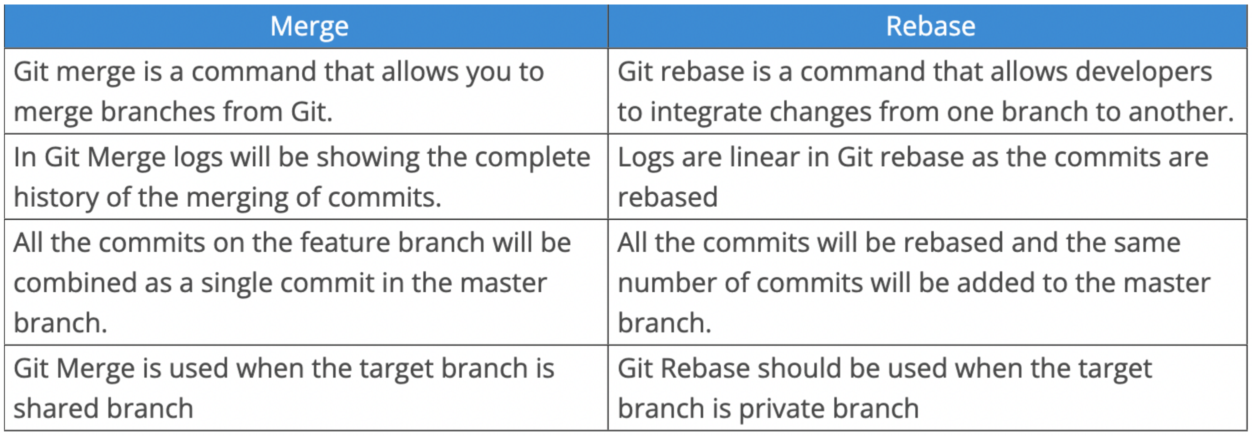
Trunk-based development
short-lived branches
master (or trunk)
merging is done more frequently and more easily for shorter branches

Git Flow

The GitHub flow branching strategy is a relatively simple workflow that allows smaller teams, or web applications/products that don't require supporting multiple versions, to expedite their work. In GitHub flow, the main branch contains your production-ready code.



Merge vs Rebase



Merge	Rebase
Git merge is a command that allows you to merge branches from Git.	Git rebase is a command that allows developers to integrate changes from one branch to another.
In Git Merge logs will be showing the complete history of the merging of commits.	Logs are linear in Git rebase as the commits are rebased
All the commits on the feature branch will be combined as a single commit in the master branch.	All the commits will be rebased and the same number of commits will be added to the master branch.
Git Merge is used when the target branch is shared branch	Git Rebase should be used when the target branch is private branch

Git best practices

1. Don’t ‘git push’ straight to master.
2. Adequately configure the commit authorship.
3. Write descriptive and meaningful commit messages.
4. Commit only related work
5. Avoid rewriting the master’s history.
6. Rebase your working branch frequently.
7. Know many Git commands and use them.

Git Revert vs Git Reset

You can think of git revert as a tool for undoing committed changes, while git reset HEAD is for undoing uncommitted changes.

Git Commit history

Git stores history as a graph of snapshots of the entire repository. These snapshots, called commits in Git, can have multiple parents, creating a history that looks like a graph instead of a straight line.

Merge strategies

A merge happens when combining two branches. Git will take two (or more) commit pointers and attempt to find a common base commit between them. Git has several different methods to find a base commit, these methods are called "merge strategies".