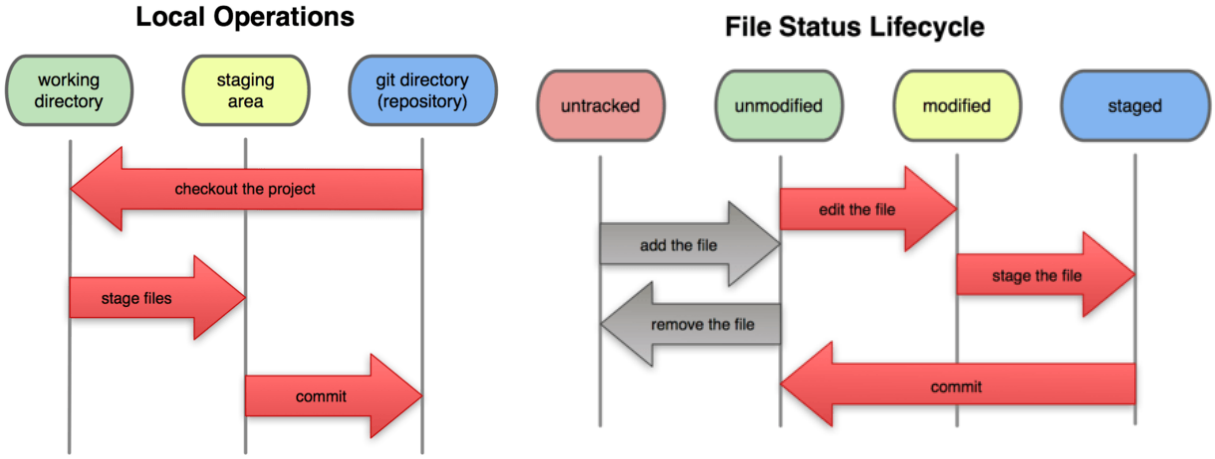
The Ultimate Guide to Git

A ”not so” comprehensive guide to version control

Version control in software development is almost a necessary part of any project, small or large. Developing without version control makes it almost impossible to keep track of older versions without making hundreds of files with the suffix “final” or “latest”. Version control keeps track of every change to every file for the entire history of the project. If a mistake is made the clock can be turned back the working version of the code. Git is the most widely adopted version control system and it is used by almost all software companies that exist.

Git

Git has three main sections: Working directory, staging area, and git directory(repository). When you clone a project from a website like GitHub or GitLab you download that entire project to your local git directory, your own copy of the remote repository. When you modify files you modify your working directory. The changes are added to the staging area automatically and when the changes are committed they are moved from the staging area to the local git repository. “git push” pushes the commits from your local git directory to the remote repository which is located



where you cloned your local repo from.

The command “git status” shows the current status of your staging area, files and changes listed under “Changes not staged for commit:” are changes that will not be included if you commit now. The changes or new files and be added to the staging area by using “git add”. There are different parameters that can be used with “git add”, -A: adds all files and changes, <filename>: writing the filename(without the brackets) adds that specific file or change. Normally all the changes are automatically staged and do not need to be added manually.

A very useful feature that git does very well is branching. When creating a branch an already existing branch is copied and the new branch can now be worked on without affecting the original branch. A common use case for branches are the so-called feature branches. Normally when an employee is working a new small feature they create a new branch just for that specific

feature so that they don’t affect the rest of the codebase. When they are finished and they code is tested and reviewed the feature branch is merged into production. There are differet types of branches, a branch can be local to a single machine or a branch can exist on the git server itself.

Common commands:

- Git add: Adds all the unstaged files/changes to the staging area Parameters: *(Adds all the files/changes), -A(Adds all the files/changes)
- Git commit: Parameters: -a(commits all changes currently staged), -m(adds a message to the commit using quotes””). Adds all the changes currently in the staging area to the local git repo.
- Git push: Parameters: –f(DANGER! Never use this parameter without permission) pushes all commits from the local git repo to the remote repo.
- Git fetch: Fetches all the changes from the default remote but does not integrate them into the current local repo.
- Git merge: Parameters: <branch-to-merge>(takes the changes from the named branch since they diverged and integrate them into the current branch).
- Git pull: Performs a git fetch followed by a git merge in order to integrate all the changes from the remote into your local repo.
- Gitk: Show all the current branches and their history on a neat GUI.
- Git status: Shows the current status for your local git repo, which files that are untracked and which files/changes are ready to be commited. This command should always be ran before any other command.
- Git checkout: Parameters: <commit-hash>(switch your local repo to the state of a specific commit), <branch-name>(switch your local git repo to a specific branch).