**Git & github:**

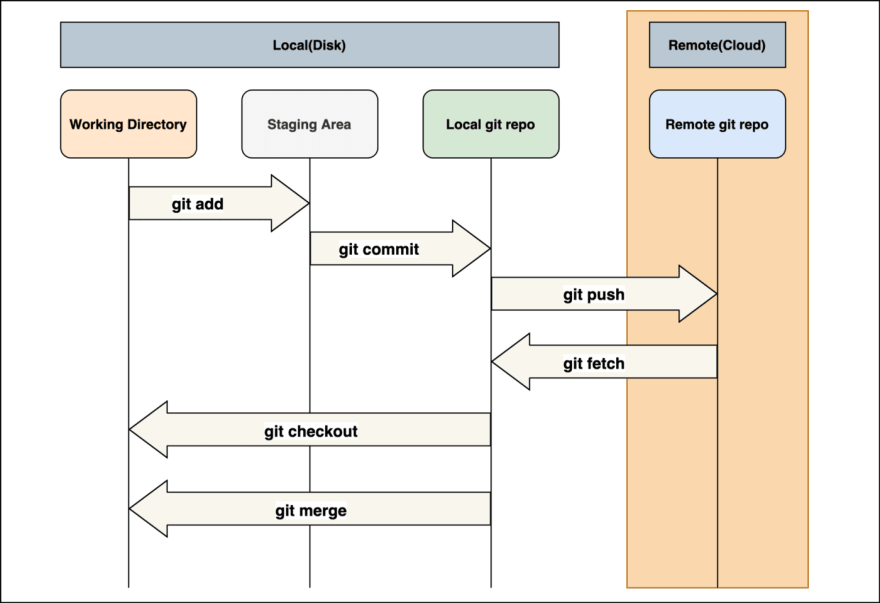
Git is an open source version control system that works locally to help developers work together on software projects that matter. This cheat sheet provides a quick reference to commands that are useful for working and collaborating in a Git repository (repo).

**Getting a Git Repository**

You typically obtain a Git repository in one of two ways:

1. You can take a local directory that is currently not under version control, and turn it into a Git repository, or
2. You can **clone** an existing Git repository from elsewhere.

In either case, you end up with a Git repository on your local machine, ready for work.



**Tracking New Files**

In order to begin tracking a new file, you use the command git add.

**Viewing Your Staged and Unstaged Changes**

If the git status command is too vague for you — you want to know exactly what you changed, not just which files were changed — you can use the git diff command.

What have you changed but not yet staged? And what have you staged that you are about to commit? Although git status answers those questions very generally by listing the file names, git diff shows you the exact lines added and removed — the patch, as it were.

**Committing Your Changes**

Now that your staging area is set up the way you want it, you can commit your changes. Remember that anything that is still unstaged — any files you have created or modified that you haven’t run git add on since you edited them — won’t go into this commit. They will stay as modified files on your disk. In this case, let’s say that the last time you ran git status, you saw that everything was staged, so you’re ready to commit your changes.

**Git Branching**

Branching means you diverge from the main line of development and continue to do work without messing with that main line.



**The GitHub Flow**

GitHub is designed around a particular collaboration workflow, centered on Pull Requests. This flow works whether you’re collaborating with a tightly-knit team in a single shared repository, or a globally-distributed company or network of strangers contributing to a project through dozens of forks. It is centered on the [Topic Branches](https://git-scm.com/book/en/v2/ch00/_topic_branch) workflow covered in [Git Branching](https://git-scm.com/book/en/v2/ch00/ch03-git-branching).

Here’s how it generally works:

1. Fork the project.
2. Create a topic branch from master.
3. Make some commits to improve the project.
4. Push this branch to your GitHub project.
5. Open a Pull Request on GitHub.
6. Discuss, and optionally continue committing.
7. The project owner merges or closes the Pull Request.
8. Sync the updated master back to your fork.