Git and Github Reference Material

Contents

[1.0 What is Git / GitHub? 3](#_Toc56282421)

[2.0 Structure of Git 3](#_Toc56282422)

[3.0 Common Terminology 3](#_Toc56282423)

[4.0 Git Setup 3](#_Toc56282424)

[5.0 Git Commands 3](#_Toc56282425)

[6.0 Git Workflows 3](#_Toc56282426)

# What is Git / GitHub?

Definition for Git from Wikipedia:

“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.”

Essentially, Git is a program that tracks code changes and allows people to work on different parts of a code base. It can also be used to identify and discuss bugs or code changes before they enter production. For Team ASTRAS, this accomplishes a couple things:

* Our code is safe. We can revert changes and have a clear revision history.
* Multiple people can work on the same code base without conflicting issues.
* If needed, we have a center for coordinating tasks and fixes.

GitHub is a website that hosts remote Git repositories. This is essentially the ‘cloud’ in which our files are stored. When we want to do some work, we pull this remote repository to our local computer, make our changes, then push our updates back to the remote repository.

*Why do I need to learn all of this? Wouldn’t it be easier to just do on Teams?*

We certainly could do all of this on Teams – however, there are a couple reasons to use Git instead:

* Version control – to keep records of versions on Teams, we would either start adding folders or files with those names. Before long, we will have a ton of content that is very obscure in how it has changed. With Git, you add a comment line on every ‘commit’, or update, that keeps track of changes.
* Anybody can pull the latest repository from GitHub. This includes the raspberry Pi. Otherwise, we would need to constantly move files from our computers to the Pi or develop directly on the Pi. Both of these options, in my experience, aren’t great.
* It is an industry standard. It is something you can add to your resume and will look great. Programming is something that is becoming more prominent in every industry, and Git/GitHub is one of the largest hosts for collaborating with programmers.

Expect to take a few hours out of your day to learn how Git works – I’ve tried to make this document as straight forward as possible, and teach you just enough to work with it.

# Repositories

GitHub is where our code is stored on the cloud – this is our **Remote Repository.** This is where we pull the current code from and push our changes to. The remote repository is also called **origin.** Our remote repository is found at <https://github.com/Team-ASTRAS/Prudentia>.

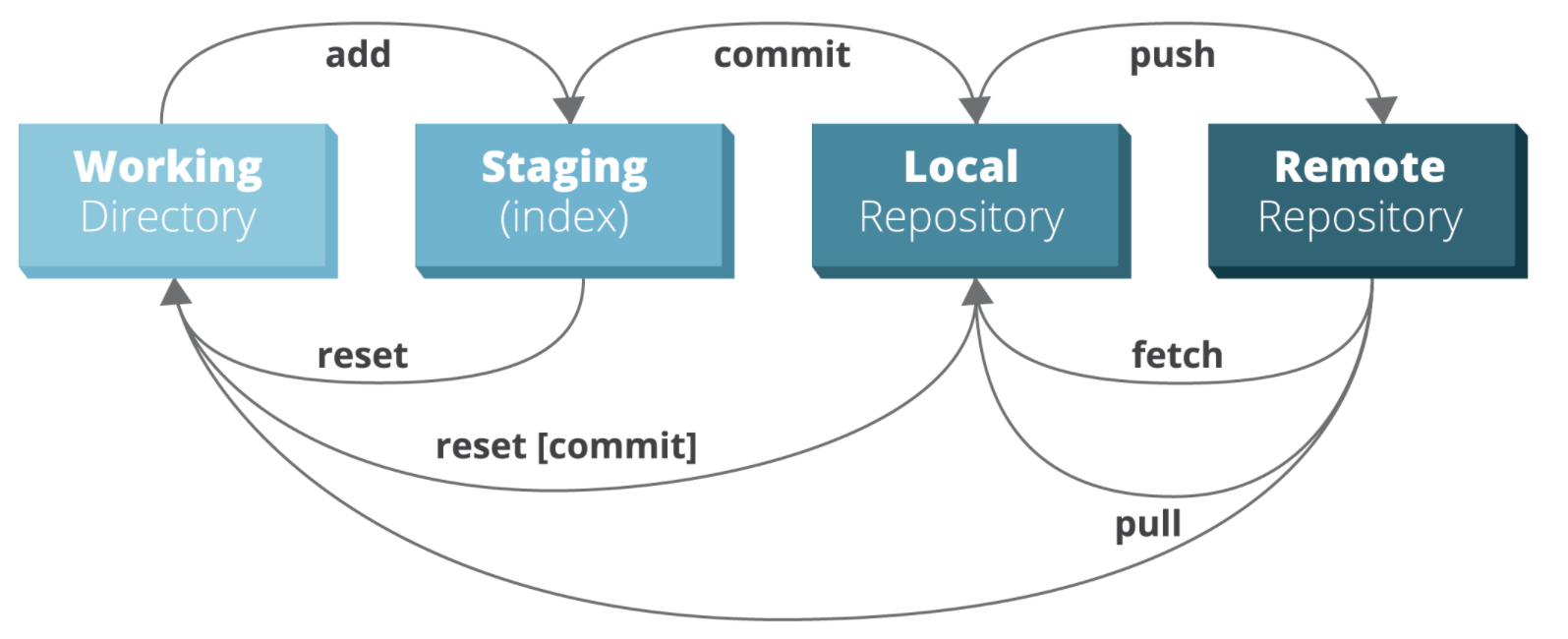
While you can make changes to the remote repository directly, typically you copy the remote repository to a **local repository.** This is a repository that sits on your computer, and changes to this repository won’t affect the origin until you push those changes to origin.

The local repository keeps track of all your files and changes. You can’t change your local repository’s current state directly; instead, you update files in the **working directory.** This is a file (which can be found under file explorer) that will have all your files and folders in it. From here, you can add/edit/delete. When you are ready to apply your changes (known as a **commit**), you first add your modified files to the **Staging Area**, then commit those changes.

Again, modifying the local repository does not affect the remote repository automatically. To make your changes visible on the remote repository, you **push** your local repository to the origin. This will update the remote repository with your latest committed local repository.

So, that’s a lot of text to take in. Below is a figure to show what this looks like graphically.

**GitHub Repository Structure**



Let’s walk through a scenario; a user logs on for the day and wants to edit something in the remote repository.

* First, the user **pulls** the origin to update the local repository with the latest files.
* Next, the user makes edits within the working directory.
* Next, the user **adds** those edits to the staging area.
  + If the user wanted to remove something from the area, **reset** can be used to revert this.
* Once all desired changes are added to the staging area, the user then **commits** the changes to the local repository.
  + If the user wanted to revert the repository to a previous commit, **reset <commitID>** can be used.
* Finally, the user could **push** the local repository to origin.

Examples of how this is done will be shown in later sections, for now the important thing is to know:

* How repositories are structured.
* Use pull to update your local repository from the origin.
* Use push to update the origin from your local repository.
* The process of apply changes in your working directory to your local repository is committing**.**

# Branches and Merging

# Git Commands

# Git Workflows