1/ What is Github?

Github is a hosting platform for git repositories.

You can put your own Git repos on Github (cloud) and access them from anywhere and share them with people around the world.

Beyond hosting repos, Github also provides additional collaboration features that are not native to Git (but are super useful).

Basically, Github helps people share and collaborate on repos.

Difference between Git and Github:

+ Git is the version control software that runs locally on your machine. You don't need to register for an account.

You don't need the internet to use it. You can use git without ever touching Github.

+ Github is a service that hosts Git repositories in the cloud and makes it easier to collabarte with other people.

You do need to sign up for an account to use Github. Ít's an online place to share work that is done using Git.

Github is not your only option... There are tons of competing tools that provide similar hosting and collaboration features,

including GitLab, BitBucket, and Gerrit.

2/ Cloning

So far we've created our own Git repositories from scratch, but often we want to get a local copy of an existing repository instead.

To do this, we can clone a remote repository hosted on Github or similar websites. All we need is a URL that we can tell Git to clone for use.

To clone a repo, simply run git clone <url>.

Git will retrieve all the files associated with the repository and will copy them to your local machine.

In addition, Git initializes a new repository on your machine, giving you access to the full Git history of the cloned project.

Command Syntax: git clone <url>

(Make sure you are not inside of a repo when you clone)

Note: git clone is a standard git command.

It is NOT tied specifically to Github. We can use it to clone repositories that are hosted anywhere! It just happens that most of the hosted repos are on Github these days.

3/ Permissions?

Anyone can clone a repository from Github, provided the repo is public. You do not need to be an owner or collaborator to clone the repo locally to your machine.

You just need the URL from Github.

Pushing up your own changes to the Github repo...that's another story entirely! You need permission to do that.

4/ Register for Github

Configuring SSH Keys

You need to be authenticated on Github to do certain operations, like pushing up code from your local machine.

Your terminal will prompt you every single time for your Github email and password, unless...

You generate and configure an SSH Key! Once configured, you can connect to Github without having to supply your username/password.

5/ Creating our first Github repo

- Option 1: Existing Repo

If you already have an existing repo locally that you want to get on Github...

+ Create a new repo on Github

+ Connect your local repo (add a remote)

+ Push up your chages into Github

- Option 2: Start From Scratch:

If you haven't begun work on your local repo, you can...

+ Create a brand new repo on Github

+ Clone it down to your machine

+ Do some work locally

+ Push up your changes to Github

6/ A Crash course on git remote

- Remote

Before we can push anything up to Github, we need to tell Git about our remote repository on Github.

We need to setup a "destination" to push up to.

In Git, we refer to these "destinations" as remotes. Each remote is simply a URL where a hosted repository lives.

- Viewing remotes

To view any existing remotes for you repository, we can run git remote or git remote -v (verbose, for more info)

This just displays a list of remotes. If you haven't added any remotes yet, you won't see anything!

Command Syntax: git remote -v

- Adding a new remote

A remote is really two things: a URL and a label. To add a new remote, we need to provide both to Git

Command Syntax: git remote add <name> <url>

Okay Git, anytime i use the name 'origin', I'm referring to this particular Github repo URL.

Example: git remote add origin https://github.com/blah/repo.git

- Origin

Origin is a conventional Git remote name, but it is not at all special. It's just a name for a URL.

When we clone a Github repo, the default remote name setup for us is called origin. You can change it. Most people leave it.

- Other commands

They are not commonly used, but there are commands to rename and delete remotes if needed.

Command Syntax: git remote rename <old> <new>

git remote remove <name>

7/ Introducing Git Push

- Pushing

Now that we have a remote set up, let's push some work up to Github!. To do this, we need to use the git push command.

We need to specify the remote we want to push up to AND the specific local branch we want to push up to that remote.

Command syntax: git push <remote> <branch>.

Example: git push origin master

git push origin master tells git to push up the master branch to our origin remote.

- Push in detail

While we often want to push a local branch up to a remote branch of the same name, we dont't have to!

To push our local pancake branch up to a remote branch called waffle we could do: git push origin pancake:waffle

Command syntax: git push <remote> <local-branch>:<remote-branch>

Example: git push origin pancake:waffle

- The -u option

The -u option allows us to set the upstream of the branch we're pushing. You can think of this as a link connecting our local branch to a branch on Github

Running git push -u origin master sét the upstream of the local master branch so that it tracks the master branch on the origin repo.

Command Syntax: git push -u origin master

What this means....

Once we've set the upstream for a branch, we can use the git push shorthand which will push our curent branch to the upstream.

Command syntax:

git push -u origin master -> git push

8/ Fetching and Pulling

- Remote Tracking Brkanches

"At the time you last communicated with this remote repository, here is where x branch was pointing"

They follow this pattern <remote>/<branch>.

+ origin/master references the state of the master branch on the remote repo named origin.

+ upstream/logoRedesign references the state of the logoRedesign branch on the remote named upstream (a common remote name).

Remote branches

Run git branch -r to view the remote branches our local repository knows about

Command Syntax: git branch -r

Result: origin/master

- Working with remote branches

Remote Branches

Once you've cloned a repository, we have all the data and Git history forr the project at the moment in time.

However, that does not mean it's all in my workspace!

The github repo has a branch called puppies, but when i run git branch i do not see it on my machine.!!!

All i see is the master branch. What's going on?

If we use git branch -r we will see origin/puppies

I want to work on the puppies branch locally!

I could checkout origin/puppies, but that puts me in detached HEAD.

I want my own local branch called puppies, and i want it to be connected to origin/puppies, just like my local master branch is connected to origin/master.

=> It's super easy!

Run git switch <remote-branch-name> to create a new local branch from the remote branch of the same name.

git switch puppies makes me a local puppies branch AND sets it up to track the remote branch origin/puppies

Command syntax: git switch <remote-branch-name>

Example: git switch puppies

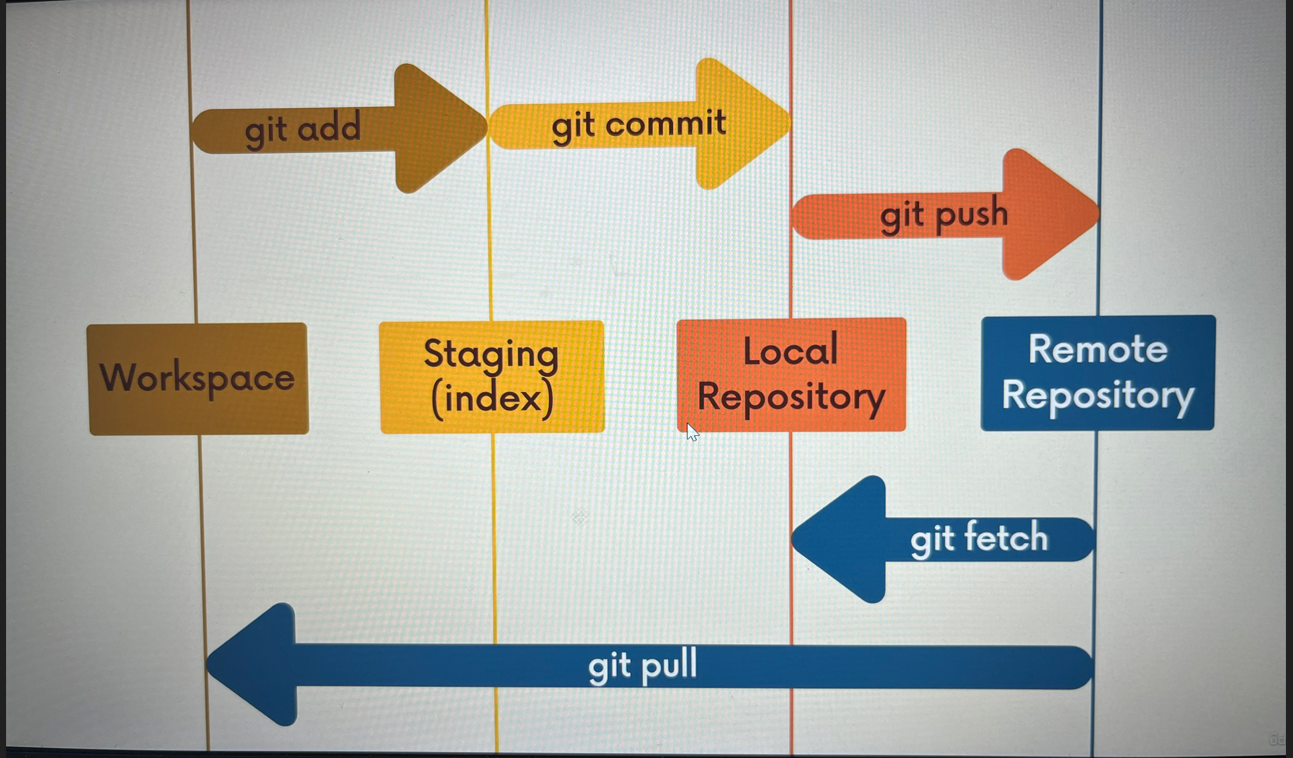
NOTE:

The new command git switch makes this super easy to do! It used to be slightly more complicated using git checkout!

Command syntax: git checkout --track origin/<branch-name>

Example: git checkout --track origin/pupies

- Git Fetch: The basics



Fetching:

Fetching allows us to download changes from a remote repository, BUT those changes will not be automatically integrated into our working files.

It lets you see what others have been working on, without having to merge those changes into your local repo.

Think of it as “Please go and get the latest information from Github, but do not screw up my working directory”

Git Fetch

The git fetch <remote> command fetches branches and history from a specific remote repository. It only updates remote tracking branches.

Git fetch origin would fetch all changes from the origin remote repository

Command Syntax: git fetch <remote>

(If not specified, <remote> defaults to origin)

We can also fetch a specific branch from a remote using git fetch <remote> <branch>

For example, git fetch origin master would retrieve the latest information from the master branch on the origin remote repository.

Command Syntax: git fetch <remote> <branch>

I now have those changes on my machine, but if I want to see them I have to checkout origin/master. My master branch is untouched.

* Git Pull: The basics

Pulling

Git pull is another command we can use to retrieve changes from a remote repository. Unlike fetch, pull actually updates our HEAD branch with whatever changes are retrieved from the remote.

“Go and download data from Github AND immediately update my local repo with those changes”

Git pull = git fetch + git merge

= update the remote tracking branch with latest changes from the remote repository + update my current branch with whatever changes are on the remote tracking branch

To pull, we specify the particular remote and branch we want to pull using git pull <remote> <branch>. Just like with git merge, it matters WHERE we run this command from. Whatever branch we run it from is where the changes will be merged into.

Git pull origin master would fetch the latest information from the origin’s master branch and merge those changes into our current branch.

Command Syntax: git pull <remote> <branch>

Note: git pull can cause merge conflict

An even easier syntax!

If we run git pull without specifying a particular remote or branch to pull from, git assumes the following:

+ remote will default to origin

+ branch will default to whatever tracking connection is configured for your current branch.

Note: this behavior can be configured, and tracking connections can be changed manually. Most people do not mess with that stuff.

Command syntax: git pull

Git fetch:

+ gets changes from remote branch(es)

+ updates the remote-tracking branches with the new changes

+ Does not merge changes onto your current HEAD branch

+ Safe to do at anytime

Git pull:

+ Gets changes from remote branch(es)

+ Updates the current branch with the new changes, merging them in

+ Can result in merge conflicts

+ Not recommend if you have uncommitted changes!