**42)Git Fetch**

When we are working with other collaborators on a GitHub repo and one of our teammate’s pushed up the changes to the master branch, but my local repo doesn’t know!

Then how do we get those changes?

This is where git fetch, and git pull come into play.

Git fetch and pull get those changes from the GitHub repo to your local repo.

But these two commands have some differences.

Timeline

Description automatically generated

What git fetch does it takes those updates from GitHub and stores them in the local repository.

This will not affect our current working directory.

Fetching allows us to download changes from remote repository.

But those changes will not be automatically integrated to our working files

It just lets you see what others have been working in, without merging those changes into your local repo.

Just think of git fetch as link

Please go and get the latest information from GitHub, but don't add it into my working directory.

Command for git fetch is

Git fetch <remote>

This command fetches branches and history from a specific remote repository. It only updates the remote tracking branches.

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

If not specified remote it defaults to origin.

We can also fetch a specific branch from a remote using the command

Git fetch <remote> <branch>

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

After fetching the changes. I will have those changes. on my machine,

But if I want to see it then I must do the checkout to origin/master.

Your local master branch will be untouched.

**43)Usage of Git Fetch**

Suppose we clone a repository from GitHub onto the local machine

git branch tells us about the branch in local repo currently

git branch -r shows all branches in remote

Now let’s create a new file in GitHub

git fetch origin downloads those files to local repo

Text

Description automatically generated

The above image tells us that we got new files from origin/main and origin/login

When we do git status we can see that

Text

Description automatically generated

Tells our branch is behind origin main by one commit.

When we do git log – oneline

We only see one file(one commit)

Text

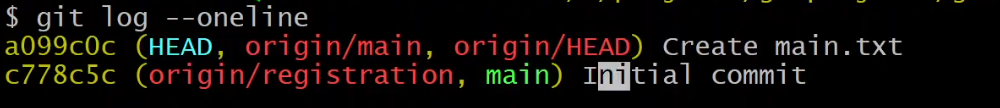
Description automatically generated

So to get the updated file list we do git checkout origin/main

Text

Description automatically generated

Now when we do ls we get the file list



When we do git log –oneline

The top line says Head is at create main.txt commit

The bottom line says the local branch is at Initial commit

**Scenario 2**

When we fetch all branches will be downloaded so now when we switch to another branch

the head will be automatically updated to the latest commit in that branch.

Text

Description automatically generated

**Scenario 3**

Now let’s say we created a local branch named branch 3

Now let’s create a branch branch 3 in GitHub and fetch it

When we see status, we see that local head is behind by 1 commit.

**Scenario 4**

Now let’s create a new branch branch 4 in GitHub

Now when we see git branch -r in GitHub we will not see branch 4 because it is not fetched yet

So, when we do git fetch origin and then we do git branch -r

We see the new branch 4

**44)Git Pull**

Pull command is somewhat like fetch.

Git pull is the command we can use to retrieve changes from the remote repository.

Unlike Fetch, pull updates our HEAD branch with whatever changes are retrieved from the remote.

In simple Words we can say that Go and download data from GitHub and immediately update my local repo with those changes

We can say that

Git pull = Git fetch + git merge

Git fetch - Update the remote tracking branch with the latest changes from the

remote repository

Git merge - update my current branch with whatever changes are on the remote

tracking branch

Syntax for git pull:

Git pull <remote> <branch>

To pull, we specify the remote and branch we want to pull using the git pull <remote> <branch><branch>.

What really matters is where we are on. Whatever the branch I am in, that is where the changes will be merged where I am pulling down to.

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

The above two lines means when we pull those changes will be merged into the current branch we are in.

Pull can result in the merge conflicts!!

We need to resolve the conflicts just like a normal merge.

Sometimes you might have some work locally that is not on GitHub and GitHub have some commits.

When you pull down, there may be conflicts.

We also have shorter syntax for the pull command just like fetch

> Git pull

If we run git pull without specifying a particular remote or branch to pull from,

Git assumes the following

* Remote will be default to origin
* Branch will default to whatever tracking connection is configured for the current branch

Text

Description automatically generated

**45)Implementation of Git Pull, Resolve merge conflicts**

We resolve merge conflicts in git pull, like normal conflicts.

We open the file to see both versions of the file and remove the unwanted version and save the file with the wanted version and push that into GitHub.

**46)Git Readme.md**

A Readme file is used to communicate important information about a repository including:

* What the project does
* How to run the project
* Why it's noteworthy
* Who maintains the project

If you put a README in the root of your project

GitHub will recognize it and automatically display it in the repo's home page.

Readme is like an entry point to learn more about the project or application.

If you observe, the extension of the README is .md.

READMEs are markdown files, ending with the .md extension. Markdown is a convenient syntax to generate formatted text. It's easy to pick up!

**47)GitHub Gists**

GitHub gists are the simple way to share code snippets and useful fragments to others.

Gists are much easier to create, but offer few features compared to normal Git repository.

Gist meaning- the substance or essence of a speech or text. (like summary)

So, we use gists when we want to share in a single file or less files the whole repository.

Every gist is a Git repository, meaning it can be cloned.

If you are signed into GitHub when you create a gist, the gist will be associated with your account and you will see it in your list of gists when you navigate to your gist home page.

<https://gist.github.com/>

Gists can be public or secret.

Public gists show up in Discover, where people can browse new gists as they're created. They're also searchable, so you can use them if you'd like other people to find and see your work.

Secret gists don't show up in Discover and are not searchable. Secret gists aren't private.

If you send the URL of a secret gist to a friend, they'll be able to see it.

However, if someone you don't know discovers the URL, they'll also be able to see your gist.

Once we create a gist in a way (Secret or public) it can’t be reverted.

**48)GitHub Free Hoisting**

GitHub pages are the public web pages that are hosted and published by GitHub.

They allow you to create a website by simply pushing your code to GitHub.(Like live server).

GitHub Pages is a hosting service for serving static web pages.

It does not support service side code like PHP, Python, Ruby or Node.

GitHub Pages supports only Html, CSS and js only.

You get unlimited project sites in the GitHub

Each GitHub repo can have a corresponding hosted website.

It is as simple as just telling the GitHub which branch to take for hosting the repo.

The default URL in GitHub pages follow this pattern

http://username.qithub.com/repo-name

Remember that you can get only one website for an account or organization

But unlimited repo project sites.

**49)Creating GitHub Pages in GitHub Repository**

We can view our website frontend on GitHub by accessing Git pages.

**50)Pull Requests**

Pull Requests are a feature built into products like GitHub & Bitbucket.

They are not native to Git itself.

They allow developers to alert team members to work that needs to be reviewed.

They provide a mechanism to approve or reject the work in each branch.

They also help facilitate discussion and feedback on the specified commits.

Pull Requests is nothing but merging in Feature branches

At some point the work we did on the feature branch need to be merge into the master branch!

There are couple of options for how to do this

1. Merge at will, without any sort of discussion with teammates. Just do whatever you want.

2. Send an email or chat message or something to your team to discuss if the changes should be

merged in

3. Pull Requests!

Pull Requests workflow

1. Do some work locally on a feature branch

2. Push up the feature branch to GitHub.

3. Open a pull request using the feature branch just pushed up to GitHub.

4. Wait for the PR to be approved and merged. Start a discussion on PR. This part depends on the team structure.

When we commit and push changes onto GitHub it’ll ask us Compare and pull request.

It means the file will be merged from that branch into the main branch.

We can create a comment if we want.

It’ll ask us to add reviewers.

The reviewers review the code and allow it to merge into the main branch.

Now in local repo if we pull the repo from GitHub we can observe that the files in the feature branch are added into main branch.

**51)Resolve Conflicts in Pull requests**

Suppose we worked on a file in a branch and sent it to pull request.

And from the main branch the same file is already updated.

So, when we try to merge the branch to main this creates a conflict.

We resolve merge conflicts in git pull, like normal conflicts.

We open the file to see both versions of the file and remove the unwanted version and save the file with the wanted version and push that into GitHub.