**30)GITHUB Introduction**

* GitHub has evolved over the years by adding a bunch of features.
* We can say in single line that GitHub is a hosting platform for the git repositories.
* GitHub allows us to host our git repository in the cloud.
* This means that we can access the code from anywhere and share the code with people around the world.

**Git Vs GitHub**

* Git has huge difference with GitHub
* Git is a version control software that runs locally on your machine.
* You don't need to register for an account.
* You don't need internet to use it.
* GitHub is a service that hosts repositories in the cloud and makes it easier to collaborate with other people.
* You do need to sign up for an account to use GitHub
* It’s an online place to share work that is done using Git.
* There are so many tools that provide similar hosting and collaboration features like Gitlab, Bitbucket, Gerrit
* These are other alternatives to GitHub

**31)GitHub and Its Advantages**

* One main thing is collaboration.
* If you ever plan on working on a project with at least one other person, GitHub will make it easier.
* Whether you want to share with your friend or to the entire world GitHub is essential.

**32)GIT Clone**

To work with the remote repositories hosted in the GitHub we need to get the local copy into the computer.

To achieve that we need is URL that we can tell git to clone the repository.

If you go to the documentation Git clone gets the repository that is not present in your machine based on the URL we provide.

git-clone - Clone a repository into a new directory

To clone a repo

git clone <url>

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

In addition to that git initializes a new repository on your machine giving you access to the fill history of the cloned project.

Anyone can clone a repository from GitHub, provided the repo is public.

You do not need to be owner or collaborator to clone the repo locally to your machine

You just need the URL from GitHub.

Pushing up your changes to the GitHub repo involves another process. You need permission to do that.

git-clone is a standard git command. It works with not just GitHub but also Gitlab, Bitbucket, Gerrit.

**Cloning process**

* Go to the repository you want to clone
* Click on Clone green button Copy URL.
* In the local system Git folder, gitprojects subfolder open GitBash and do
* git clone <URL>
* The entire repository will be cloned along with the commit history.

**33)GitHub SSH Config setup in the local machine**

We can connect to GitHub using two methods

1. Using HTTPS

2. Using SSH

When working with a GitHub repository, you'll often need to identify yourself to GitHub using your username and password. (Https method).

Earlier we cloned used HTTPS

An SSH key is an alternate way to identify yourself that doesn't require you to enter your username and password every time.

Using the SSH protocol, you can connect and authenticate to remote servers and services.

With SSH keys, you can connect to GitHub without supplying your username and personal access token at each visit.

To implement SSH keys in the system, we need to create the SSH key pair using

ssh-keygen -t ed25519 -C "your\_email@example.com"

Next add your SSH key to the ssh agent, ensure the ssh-agent is running using command

eval 'ssh-agent -s'

Next add the ssh key to the ssh agent

ssh-add -/.ssh/id ed25519

We need to go to the GitHub profile and add the ssh key pair in the settings.

We can check that we are authenticated to the GitHub or not using

Ssh -T [git@github.com](mailto:git@github.com)

**34)Creating the first repository in GitHub**

How do I get my code in GitHub?

Option 1:

Existing Repo

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

1. Create a new repo on GitHub

2. Connect your local repo (add a remote)

3. Push up your changes to GitHub

Option 2:

Start from scratch

If you haven't begun working on the local repo.

1. Create new repo in GitHub

2. Clone it down to your machine

3. Do some work locally

4. Push your changes to GitHub

We need to make somehow the local repo know about the GitHub repo

We need to tell about our remote repository present on GitHub.

We need to set up the "destination" in the local repo.

In git we refer to these destinations as "remotes".

Each remote is simply a URL where a hosted repository lives.

When we type

git remote

We will see if the local repository is associated to any remote

**35)Add, Rename, Remove remote origin URL**

**Adding Remote**

Git remote add origin <URL>

Origin is the short name for the URL.

That means whenever we use the name origin, we are referring to the GitHub URL like an alias name.

It is like whatever URL we save, it is stored in a variable named origin.

The default name is origin, we can name it anything we want.

When we clone a GitHub repo the default name is origin.

We can also use like this

Git remote add mygithuburl <URL>

This means anytime you use the name "mygithuburl", we are referring to this particular

GitHub repo URL.

Commonly used remote name is origin.

**Renaming the remotes**

Git remote rename <old-name> <new-name>

**We can also remove the remote using**

Git remote remove <name>

git remote add origin <https://github.com/leelawebdev/sample-project.git>

The above means we are adding a URl and it will named as origin

git remote It tells us the variable name

origin

git remote -v It tells us the URL

<https://github.com/leelawebdev/sample-project.git>

git remote rename origin neworigin we are renamimg origin as neworigin

git remote

neworigin

Git remote remove neworigin We are deleting the URL saved

Text

Description automatically generated

Text

Description automatically generated

**36)Push the local repository and commits to GitHub repository**

git push originname branchname (Only that branch is pushed)

git push origin master

git push origin newfeature (To push another branch named newfeature)

**37)Pushing the local changes from one branch to another remote branch**

When we create a new GitHub repository without creating a readme file, the first branch created is the master branch.

When we create a new GitHub repository with a readme file, the first branch created is the main branch.

Lets create a file named index,txt

git remote add origin <https://github.com/leelawebdev/git-push.git>

git push origin master

We notice that the since there is no master branch in remote, a new branch named master is created in remote which references master of the local.

We can push changes to not only to master we can push the changes in master to the master or any branch to the different branch in GitHub.

Git push <remote> <local-branch>:<remote-branch>

git push origin master:newfeature

**38)Git Push -u, Set upstream for the local branch**

**-u option**

The -u option allows us to set the upstream of the branch we are pushing.

You can think of this as a link between the local branch to a branch in the GitHub.

When we git remote add origin <https://github.com/leelawebdev/git-upstream.git>

It means we are adding this url as remote

When we add from master branch in local it will be added to master in remote(If there is no master in remote it will be created)

**Git push -u origin master**

Running this command sets the upstream of the local master branch so that it tracks the master branch on the origin repo.

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

So now when we git push it’ll automatically add to previous URL

git push origin master (Normal method)

git push -u origin master

git push (From next time it will automatically push) No need to write origin master

The above will work for only master branch we cant do it with other branch.(Because git push -u origin master we did it with master branch)

If we try to add from another branch and directly use git push it won’t work.

**39)Difference Between Main and Master Branch**

When we create a new GitHub repository without creating a readme file, the first branch created is the master branch.

When we create a new GitHub repository with a readme file, the first branch created is the main branch.

**40)Remote Tracking Branches**

Diagram, schematic

Description automatically generated

Local is at top.

Remote is at bottom.

When we clone a repository from GitHub.

Initially the local master and remote master are set at the same commit.

When we do a new commit. (In below pic 2)

The local moves by 1, but remote stays at same, because the remote stays at the same point where the GitHub repository is pointing to.

The remote pointer will move only when we push the commits to the remote.

When we type git status, we can see a message that says

Local main ahead to remote main by n commits.

Diagram

Description automatically generated

**41)Checkout the Remote Tracking Branches in the local git repository**

Let’s say in GitHub we create a repository with multiple branches.

When we clone it to local machine, we by default are in main branch

But when we type git branch in local machine it won’t show all branches but only the main branch.

But if we type git branch -r (It’ll show the branches in remote too)

So, if we want to work on other branches

git switch newfeature

(A message will pop saying this branch is local is set to track remote branch newfeature)