

## What is git ?

- ☐ **Git** is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.
- ☐ 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

## Why a Version Control System like Git is needed

- ☐ Real life projects generally have multiple developers working in parallel. So a version control system like Git is needed to ensure there are no code conflicts between the developers.
- ☐ Additionally, the requirements in such projects change often. So a version control system allows developers to revert and go back to an older version of the code.
- ☐ Finally, sometimes several projects which are being run in parallel involve the same codebase. In such a case, the concept of branching in Git is very important.

Difference B/W GIT and SVN

	SVN		GIT			
	➤ Centralized Repository		➤ Distrubuted centralized Repository			
	➤ Code Develops at Centrailized Repo		Code Develops at Local Centrailized Repo			
	➤ Internet is needed always		Only Needed to push code			

## Step1:

## Installation and uninstalition of git in your machine

Install Git in windows

### Downloading Git :-

<https://git-scm.com/downloads>

Default path of Git In windows  
C:\Program Files\Git

### Debian/Ubuntu:-

```
sudo apt-get update
sudo apt-get install git -y
```

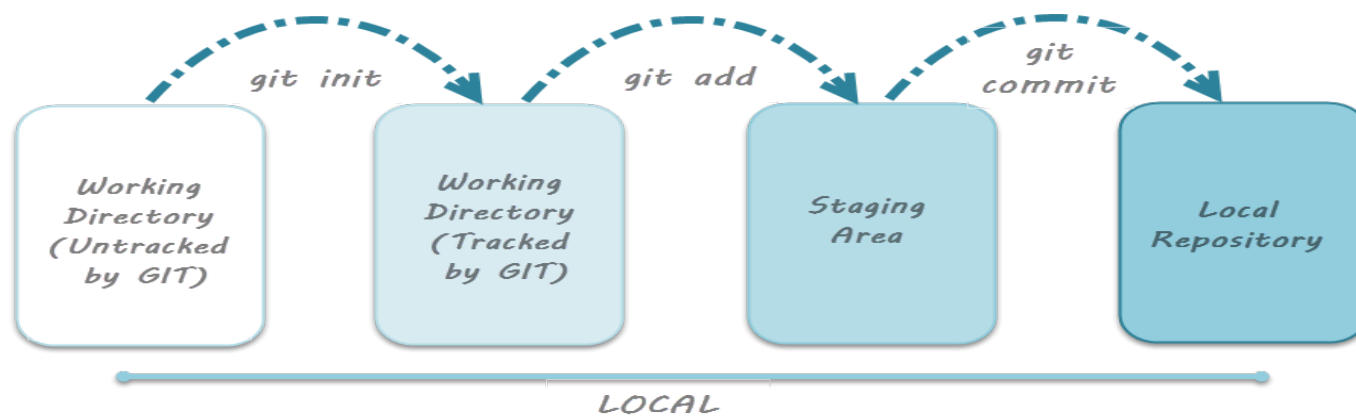
To check git is installed in windows or not use below command  
git --version

### Uninstall git in windows:-

Control panel --> Programs --> Uninstall Program

Uninstall git in Ubuntu:  
sudo apt-get remove

## Phases In GIT : -



## Step2:

### Setting your username in Git

Git uses a username to associate commits with an identity. The Git username is not the same as your GitHub username.

You can change the name that is associated with your Git commits using the `git config` command. The new name you set will be visible in any future commits you push to GitHub from the command line. If you'd like to keep your real name private, you can use any text as your Git username.

Changing the name associated with your Git commits using `git config` will only affect future commits and will not change the name used for past commits.

#### Setting your Git username for every repository on your computer

Open Git Bash.

Set a Git username & mail

1.

```
$ git config --global user.name "Rushi"
```

```
$ git config --global user.email "Rushi@yourcompany.com"
```

2.

```
$ git config --global user.name
```

```
> Rushi
```

```
$git config --global user.email
```

```
> Rushi@yourcompany.com
```

## Step3:

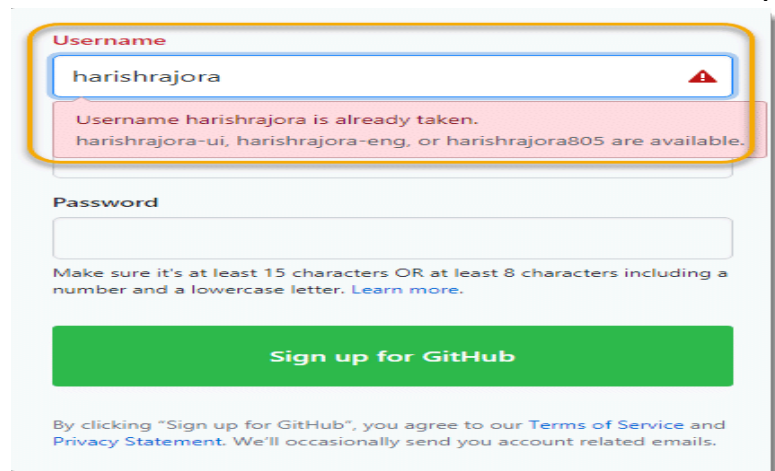
**GitHub** is a web-based [centralized system for repositories](#). It used by millions of people to work on millions of projects. Although it is not a direct part of the Git project, it is very rare that one might escape it. Not only hosting the repositories but many other functions such as **issue tracking**, **code review**, etc. can be done on GitHub with **GitHub Account**.

This chapter is about the first step in contributing to the world where so many collaborations happen in a day. We will learn about creating our account and logging in into the GitHub throughout this tutorial.

### Setting Up GitHub Account

To set the account visit <https://github.com/join?source=header-home>  
Setting your GitHub account is easy and very simple.

The login form will appear on the same page. Fill out the form with your details to create an account on GitHub.

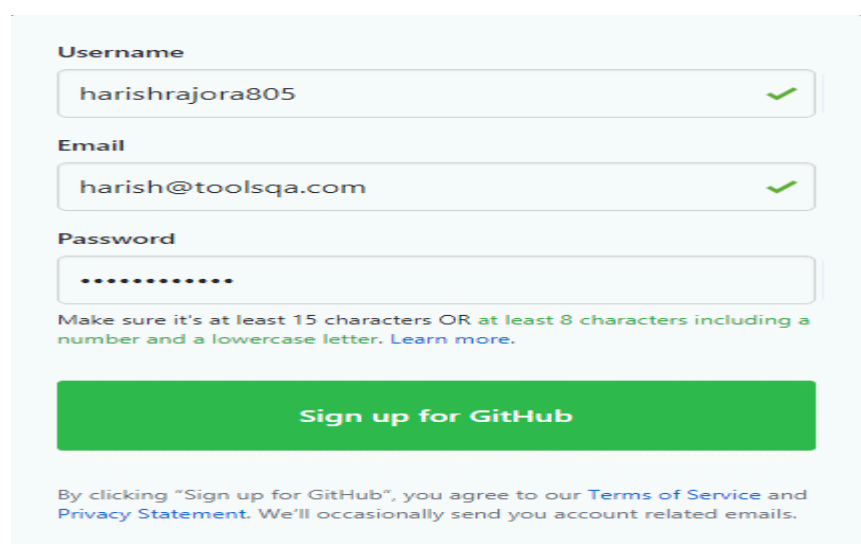


The screenshot shows the GitHub sign-up form. The 'Username' field contains 'harishrajora'. A red error message box is displayed below the field, stating: 'Username harishrajora is already taken. harishrajora-ui, harishrajora-eng, or harishrajora805 are available.' The 'Password' field is empty. Below the password field, there is a green button labeled 'Sign up for GitHub'. At the bottom, there is a small text: 'By clicking "Sign up for GitHub", you agree to our [Terms of Service](#) and [Privacy Statement](#). We'll occasionally send you account related emails.'

**Note:** GitHub will warn you if there are any duplicate entries i.e. if that username is already taken by some other individual or not etc. Along with the error, GitHub will suggest you the available attributes also. For example, your username could be duplicate because whatever you are thinking as your username, might have been already taken by someone else.

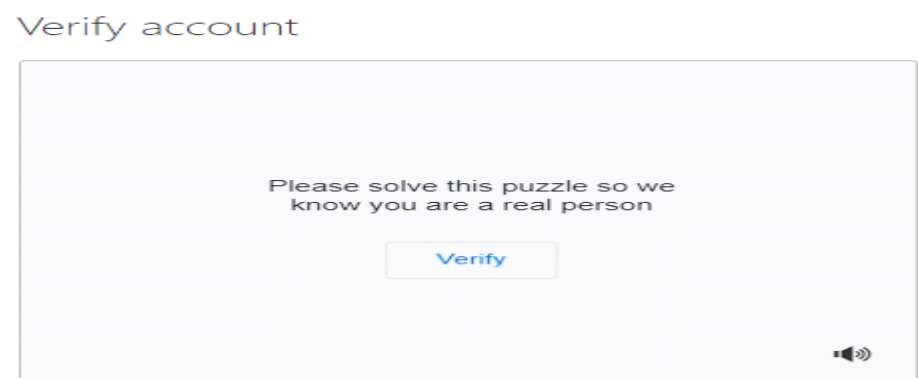
In GitHub, you will be known by your **username**. So, it has to be unique. This error can also be seen in the below screenshot. Here, the username **harishrajora** is already taken by someone else so GitHub is raising the error and suggesting some unique usernames.

Avoid those duplicate entries and fill the form with your details. The green tick will symbolize correct and unique entries.



The screenshot shows the GitHub sign-up form with successful entries. The 'Username' field contains 'harishrajora805' and has a green checkmark. The 'Email' field contains 'harish@toolsqa.com' and has a green checkmark. The 'Password' field is filled with dots. Below the password field, there is a green button labeled 'Sign up for GitHub'. At the bottom, there is a small text: 'By clicking "Sign up for GitHub", you agree to our [Terms of Service](#) and [Privacy Statement](#). We'll occasionally send you account related emails.'

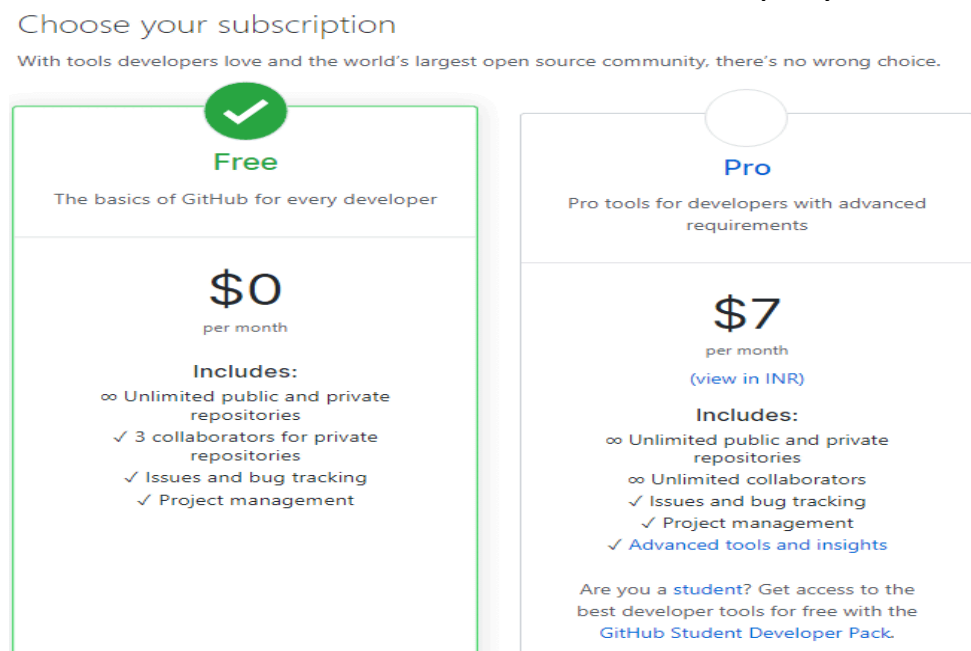
Once you press **Sign up for GitHub** button, you will be prompted to verify that you are not a robot.



The screenshot shows the 'Verify account' page. It contains a large box with the text: 'Please solve this puzzle so we know you are a real person'. Below this text is a button labeled 'Verify'. In the bottom right corner, there is a small speaker icon.

## Choosing a GitHub Account Plan

Once you have verified your identity, you can choose the GitHub plan you want to subscribe for.



For this tutorial and in general as a beginner, **GitHub Free** plan is more than enough.

**GitHub Pro** is for those who would like to have more private repositories and people contributing to these repositories are high in number. These are generally organizations. You also get advanced tools if you choose GitHub Pro such as protected branches or graphs which denotes insights of your repositories like contributors, traffic, commits, etc. You can visit this [link](#) to read more about GitHub Pro plan.

Tailoring your experience is a survey that GitHub takes while you create your account. It will be the next step in the process.

**What is your level of programming experience?**

- ☐ None—I don't program at all
- ☒ New to programming
- ☐ Somewhat experienced
- ☐ Very experienced

**What do you plan to use GitHub for? (Select up to 3)**

- ☐ Learning to code
- ☒ Learning Git and GitHub
- ☒ Host a project (repository)
- ☐ Creating a website with GitHub Pages
- ☒ Collaborating with my team
- ☐ Finding a project to contribute to
- ☐ School work / School-related project
- ☐ The GitHub API
- ☐ I don't know yet
- ☐ Other (please specify)

**Note:** You might or might not get this survey window.

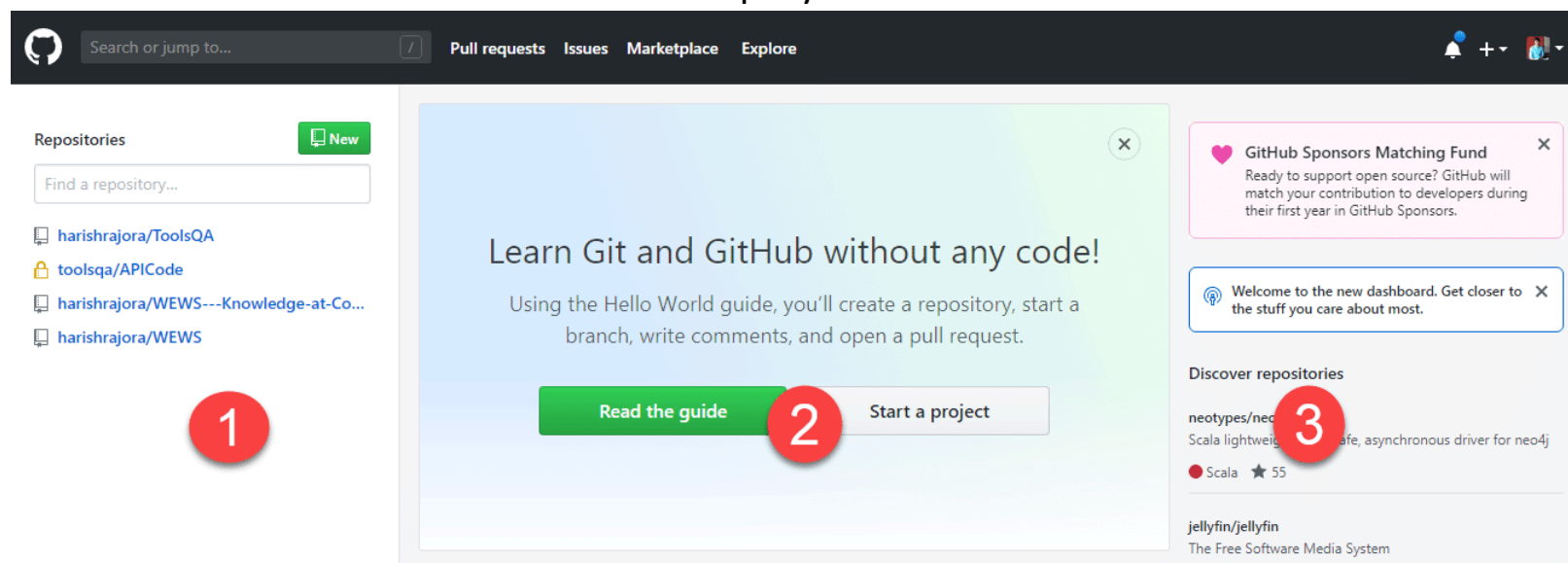
This survey is created to collect data about the users and the reason they have decided to create the account on GitHub. It contains three questions. You can skip this step if you want to.

As the next step, you would be asked to verify your email address. You can verify it by clicking the link GitHub sent you on your email.

## GitHub Account Dashboard

Now that GitHub account is all set up, you can log in through your credentials on the [GitHub's website](#). Logging in will land on the GitHub dashboard which is personalized for everyone according to interests.

A GitHub dashboard will contain three sections.

**Note:**

The header section contains important concepts and discussing them here will create complexity. Please ignore the section for now.

1. **GitHub Repositories**
2. **GitHub Discover Repositories**

These are briefed in the below section but these will be discussed in detail in the upcoming tutorials.

## GitHub Repositories

GitHub repositories section will contain all the repositories on which the user is working. For the ease, one can just toggle between these repositories and start working on them again.

## GitHub Discover Repositories

This section is newly introduced by GitHub on the dashboard. In this section, a person will be able to see some repositories which match his interests. If you are not working on any repository, you can always explore the repositories through this section and build your reputation on GitHub.

This must be enough for you to start on GitHub. It is the first step so even if you do not understand the interface or how are things listed on the website, don't worry. In the forthcoming tutorials, we will cover these things in detail.

But, since the sign up on the website is done and you have become a member of the most popular community (with respect to computer science), try to explore GitHub on your own a little bit. Discover some amazing repositories and see some issues raised by the people on your favorite software

### Task1:

#### Create Multiple Repositories in Github and Push code from Local to Github Repository

Solution:

Step1: `git init`

Step2: `touch file1 file2 file3`

Step3: `git status`

(note if the files are in red colour it means those are in working area it shows in green colour those are in Indexing/staging area)

Step4: move files from working to indexing area

Use any below command for that

```
--> git add filename
--> git add filename1 filename2
--> git add .
```

```
--> git add *  
--> git add -A
```

**Step5:git status**

Step6:move the file from indexing to local repository  
git commit -m "your commit message"

Step6: git log

Step7:Add github Repository url in your local repository  
git remote add origin <https://github.com/devopsrishi/helloworld>

Step8: Now enter below command to push code from local to central  
git push -u origin master

**Class2:****Checking Logs:****To Check Logs of your repository**

```
$ git log
```

☐ **To Check Logs of your repository in one line**

```
$ git log --oneline
```

☐ **To check particular n Number of logs**

```
$ git log -n
```

```
$ git log --oneline -3
```

**➤ Checking Logs based with author:-**

```
$ git log --author=devopsrishi
```

```
$ git log --author=devopsrishi -3
```

**➤ Checking Logs based on time:-**

```
$ Syntax: git log --since=yy-mm-dd
```

```
$git log --since=2019-04-15
```

### ➤ Checking logs till date:

\$ Syntax: git log --until=yy-mm-dd

```
$git log --until=2019-04-15
```

### Rollback Commands Reset:

Stages	Reset Commands
➤ Local to staging	➤ git reset --soft commitid
➤ Staging to workspace	➤ git reset head filename
	➤ git reset head *
➤ Local Repo to Workspace	➤ git reset --mixed commitid

Note: To get data from 4<sup>Th</sup> commit Id we need to provide 3<sup>rd</sup> commit id

### Task5: How to Give Collaboration acesses to Team Members

Collaboration Acesses :-

Loginto Repo --> settings -->Collaborators -->ProvideGithub-ID--> Add Collaborator

### Task6: Branching & Conflict Error

A branch in Git is simply a lightweight movable pointer to one of these commits. The default branch name in Git is master. As you initially make commits, you're given a master branch that points to the last commit you made. ... Multiple branches pointing into the commit's data history.

Git Branch commands	Explination
<input type="checkbox"/> git branch	<input type="checkbox"/> list all branches
<input type="checkbox"/> git branch branch-name	<input type="checkbox"/> creates new branch
<input type="checkbox"/> git checkout branch-name	<input type="checkbox"/> switching branch
<input type="checkbox"/> git checkout -b branch-name	<input type="checkbox"/> creating new branch and switching to new
<input type="checkbox"/> git branch -D <branch>	<input type="checkbox"/> delete branch

<input type="checkbox"/> git merge branchname	<input type="checkbox"/> merging (come to main branch and merge)
---	--

## What is Conflict Error?

When two developers modifies same file in two different branches and when they merge code conflict error will occur.

**Follow below steps to get conflict error:**

Do the below steps in master branch

## Git bash

- ✓ Do the below steps in master branch
- ✓ touch rushi
- ✓ vi rushi
- ✓ i
- ✓ Hello Good Morning
- ✓ Esc :wq
- ✓ git add rushi
- ✓ git status
- ✓ git commit -m "file modified by developer1"

Now Create a another branch make changes on same file

- ✓ git checkout -b master1
- ✓ vi rushi
- ✓ i
- ✓ Hello Good Morning
- ✓ hello good evening
- ✓ Esc :wq
- ✓ git add rushi
- ✓ git status
- ✓ git commit -m "file modified by developer2"

Now come to master branch and merge code

```
git checkout master
```

now merging

```
git merge master1
```

**Solving the issue:-**

Open the conflict error file

vi file1

## Remove unwanted data

git add --> git commit --> git push

## What is Stash?

Stash is a Temporary storage available in git. To save the files which are ava

[illegible]



### Alias Commands:-

Description
-------------

➤ git config --list	➤ list out alias & user details
➤ git config --global alias.l "log"	➤ Configuring alias command for log
➤ git config --global alias.b "branch"	➤ Alias command for branch
➤ git config --global alias.l1 "log --oneline"	➤ Alias command for log --oneline

## 403 Permission Denied Error:

```
$ git push -u origin master
remote: Permission to kiran415415/zoom.git denied to devopsrishi.
fatal: unable to access 'https://github.com/kiran415415/zoom.git/': The requested URL returned error: 403
```

### Steps to Resolve:

Go to search bar --> Credentials --> Windows Credentials --> Remove git credentials

### Task :

## Push Code from one Repository to another Repository

```
DELL@DESKTOP-QFK4SRT MINGW64 /d/devops-Nov/ICICI/zoom (master)
$ git remote add origin https://github.com/kiran415415/project2.git
fatal: remote origin already exists.
```

```
solu: git remote rm origin
```

If any updated changes are there in central in cant push the code u wil get below error

```
Solution : You can use git pull or -f
DELL@DESKTOP-QFK4SRT MINGW64 /d/devops-Nov/ICICI/project2 (master)
$ git push -u origin master
To https://github.com/kiran415415/project2.git
 ! [rejected]      master -> master (fetch first)
error: failed to push some refs to 'https://github.com/kiran415415/project2.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

DELL@DESKTOP-QFK4SRT MINGW64 /d/devops-Nov/ICICI/project2 (master)

## GIT

N.Rushikesh 7013882648  
[www.devopsbyrushish.com](http://www.devopsbyrushish.com)

```
$ git push -u origin master -f
```

GIT TAGS: