
Introduction to Git and Github

DAY 2

What will be covered in today's class

- Intro and History (brief)
- Difference between Git and GitHub
- Git features
- Installation

What is Git and its history

- Distributed Version Control.
- The user keeps the entire code history and its location on individual machines.
- **Advantage-**
- User can make any changes without access to internet(except the push and pull changes).

Git was introduced in **2005**.

Created by Linus Torvald to help in Linux Kernel Development.

What is Git and its history

- The Linux Kernel - an open source project used archived files and patches to pass around changes. This was before 2002.
- From 2002 They started using BitKeeper (DVCS). BitKeeper was free for the Linux community till 2005 when a dispute caused BitKeeper revoke the privileges.
- Within one week Linus torvalds announced he was taking a working “Vacation” to decide what to do about finding a new VCS for Linux.
- The outcome of this vacation was Git.
- Why named Git? There are many made-up full forms. But according to Torvalds, He just liked the name from the Beatles Song ‘I am So Tired’ verse 2. He also says Git is British slang for ‘Stupid Person’.
- Also more popularly believed is that Git stands for ‘Global Information Tracker’

Git History

- **2002**
 - Linus uses BitKeeper to track Linux.
 - And BK gets Better, and Linux scale better.
- **April 6, 2005**
 - BitKeeper drops free license.
 - Linus write his own SCM, Git.
- **April 18, 2005**
 - Git can merge.
- **June 16, 2006**
 - Git is officially used to track Linux.
- **Feb 14, 2007**
 - Git 1.5.0 is released.
 - Major usability efforts.

What is GitHub?

GitHub is a Git repository hosting service, but it adds many of its own features. While Git is a command line tool, **GitHub** provides a Web-based graphical interface. It also provides access control and several collaboration features, such as a wikis and basic task management tools for every project.


It allows code collaboration.

Apart from this it also has various other uses like UI, code documentation, bug tracking, pull requests(pr),etc.

It was formed in 2008.

Difference in Git and GitHub

Git vs. GitHub comparison	
GIT	GITHUB
Installed locally	Hosted in the cloud
First released in 2005	Company launched in 2008
Maintained by The Linux Foundation	Purchased in 2018 by Microsoft
Focused on version control and code sharing	Focused on centralized source code hosting
Primarily a command-line tool	Administered through the web
Provides a desktop interface named Git Gui	Desktop interface named GitHub Desktop
No user management features	Built-in user management
Minimal external tool configuration features	Active marketplace for tool integration
Competes with Mercurial, Subversion, IBM, Rational Team Concert and ClearCase	Competes with Atlassian Bitbucket and GitLab
Open source licensed	Includes a free tier and pay-for-use tiers

©2018 TECHTARGET. ALL RIGHTS RESERVED. 

Git Features

Git is the most important version control system as it is open-source, easy to understand and handle.

It allows a team of people to work together, all using the same files.

Git provides each developer a local copy of the entire development history, and changes are copied from one such repository to another.

Important Features of Git

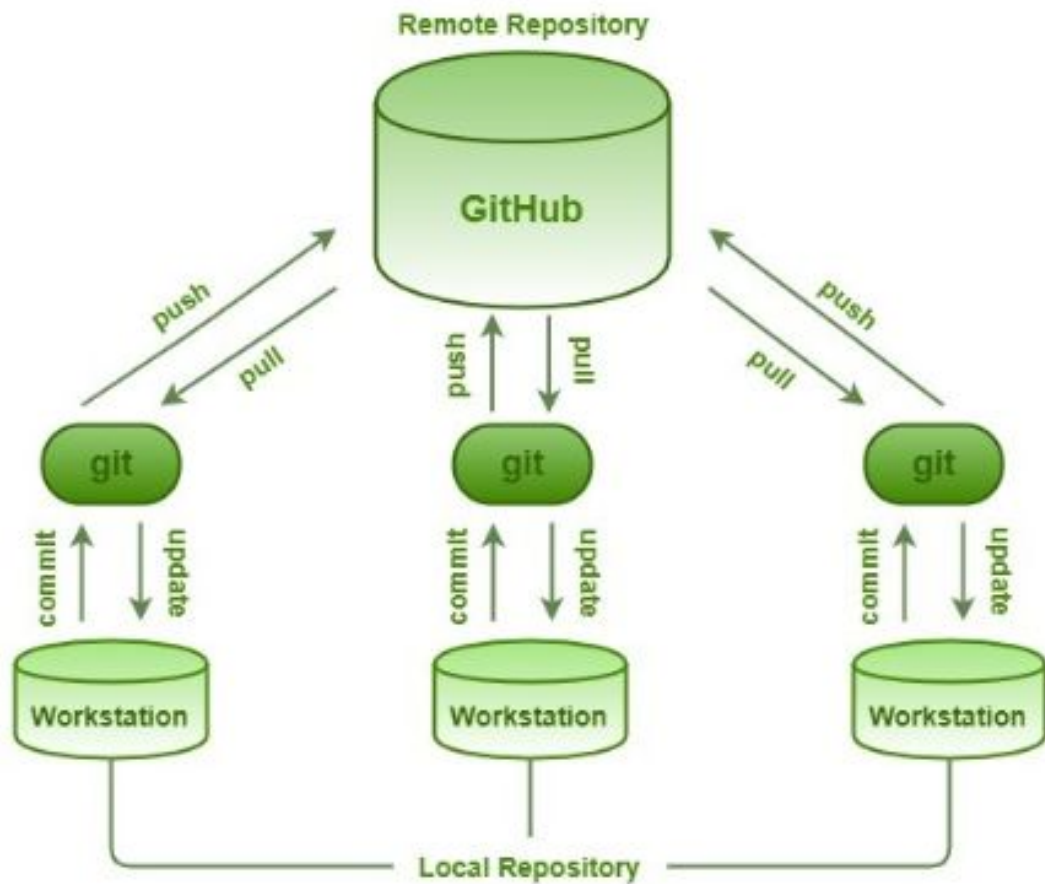
Git is the version control system.

Having a Central Server results in a problem of Data Loss or Data disconnectivity in case of a system failure of the central server. For this, Git mirrors the whole repository on each snapshot of the version that is being pulled by the user.

Having a distributed system, Git allows the users to work simultaneously on the same project, without interfering with others' work.

When a particular user gets done with their part of the code, they push the changes to the repository and these changes get updated in the local copy of every other remote user which pulls the latest copy of the project.

Distributed Version Control System



Git is compatible with all the Operating Systems that are being used these days.

The users who were not using Git in the first place can also switch to Git without going through the process of copying their files from the repositories of other VCS's into Git-VCS.

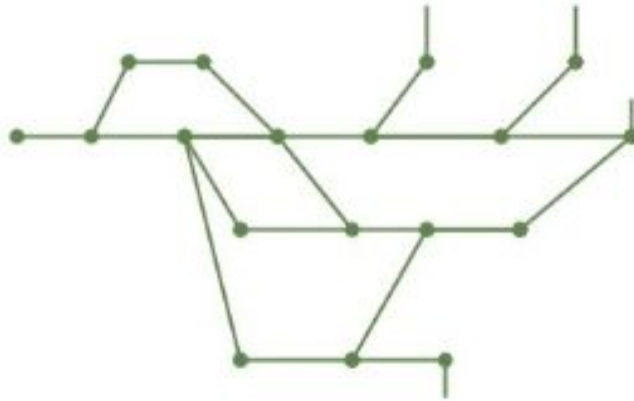
Git allows users from all over the world to perform operations on a project remotely. A user can pick up any part of the project and do the required operation and then further update the project.

This can be done by the Non-linear development behavior of the Git.

Git supports rapid branching and merging and includes specific tools for visualizing and navigating a non-linear development history.

Git allows its users to work on a line that runs parallel to the main project files. These lines are called branches.

Branches in Git provide a feature to make changes in the project without affecting the original version. The master branch of a version will always contain the production quality code.



Git keeps a record of all the commits done by each of the collaborators on the local copy of the developer.

A log file is maintained and is pushed to the central repository each time the push operation is performed.

These features have made Git the most reliable and highly used Version Control System of all times.

GitHub allows to perform all the Push and Pull operations with the use of Git.

In subsequent days we will see all the functioning of git and github.

Quiz Time (Kahoot!)

PIN - 6242071

Time for Some Work!

Installing Git

Debian / Ubuntu (apt-get)

1. From the Terminal
 - a. `sudo apt-get update`
 - b. `sudo apt-get install git` (Fedora users can use `sudo dnf/yum install git`)
2. Verify the installation was successful by typing
 - a. `git --version`
3. Configure your Git username and email using the following commands
 - a. `git config --global user.name "LoneWolfKJ"`
 - b. `git config --global user.email "kanjaldalal1000@gmail.com"`

Windows

Go to <https://git-scm.com/download/win>

Downloading Git



You are downloading the latest (**2.26.1**) **64-bit** version of **Git for Windows**. This is the most recent **maintained build**. It was released **6 days ago**, on 2020-04-14.

[Click here to download manually](#)

Other Git for Windows downloads

Git for Windows Setup

32-bit Git for Windows Setup.

64-bit Git for Windows Setup.

Git for Windows Portable ("thumbdrive edition")

32-bit Git for Windows Portable.

64-bit Git for Windows Portable.

The current source code release is version **2.26.1**. If you want the newer version, you can build it from [the source code](#).

Windows

Step 1 :

Right click on downloaded Git file and run as administrator, then you can see the below information dialogue to go further installation. Click on next

Step 2:

It will ask for Git installation directory by default it will be in ***C:\Program Files\Git.*** Click on next.

Step 3: Selecting the git components you want. As shown in next slide.



Git 2.16.1.4 Setup



Select Components

Which components should be installed?



Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

- ☐ Additional icons
 - ☐ On the Desktop
- ☒ Windows Explorer integration
 - ☒ Git Bash Here
 - ☒ Git GUI Here
- ☒ Git LFS (Large File Support)
- ☒ Associate .git* configuration files with the default text editor
- ☒ Associate .sh files to be run with Bash
- ☐ Use a TrueType font in all console windows
- ☐ Check daily for Git for Windows updates

Current selection requires at least 221.6 MB of disk space.

<https://gitforwindows.org/>

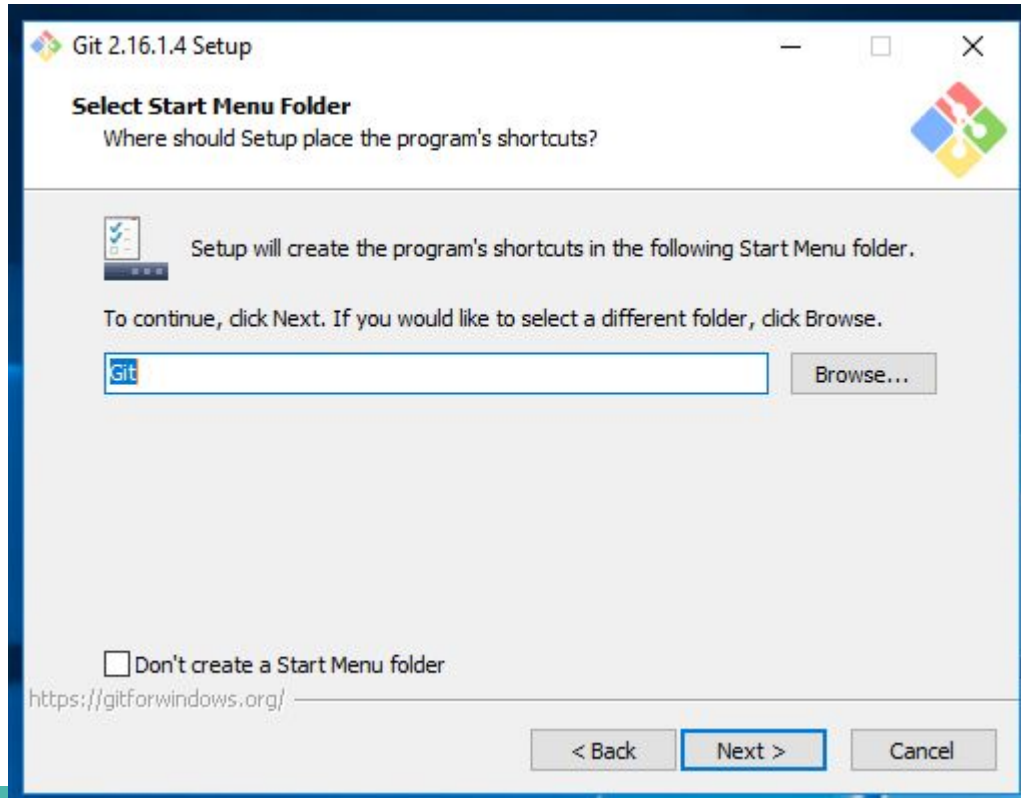
< Back

Next >

Cancel

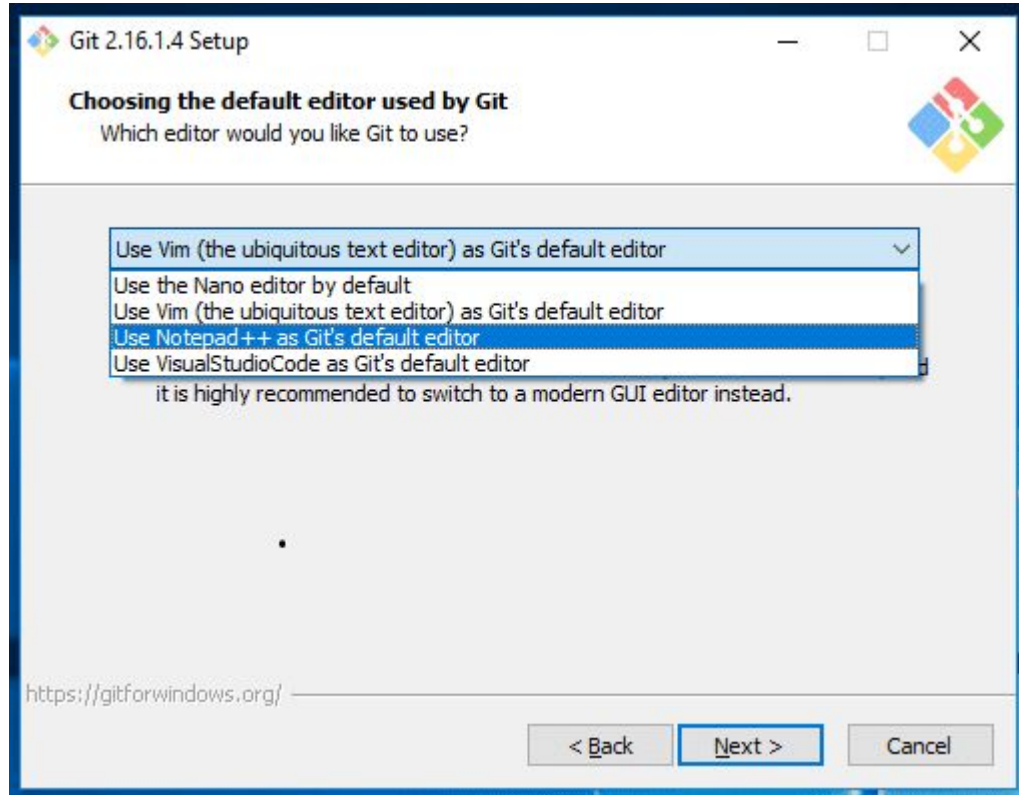
Windows

Step 4: Leave as Default, click Next.



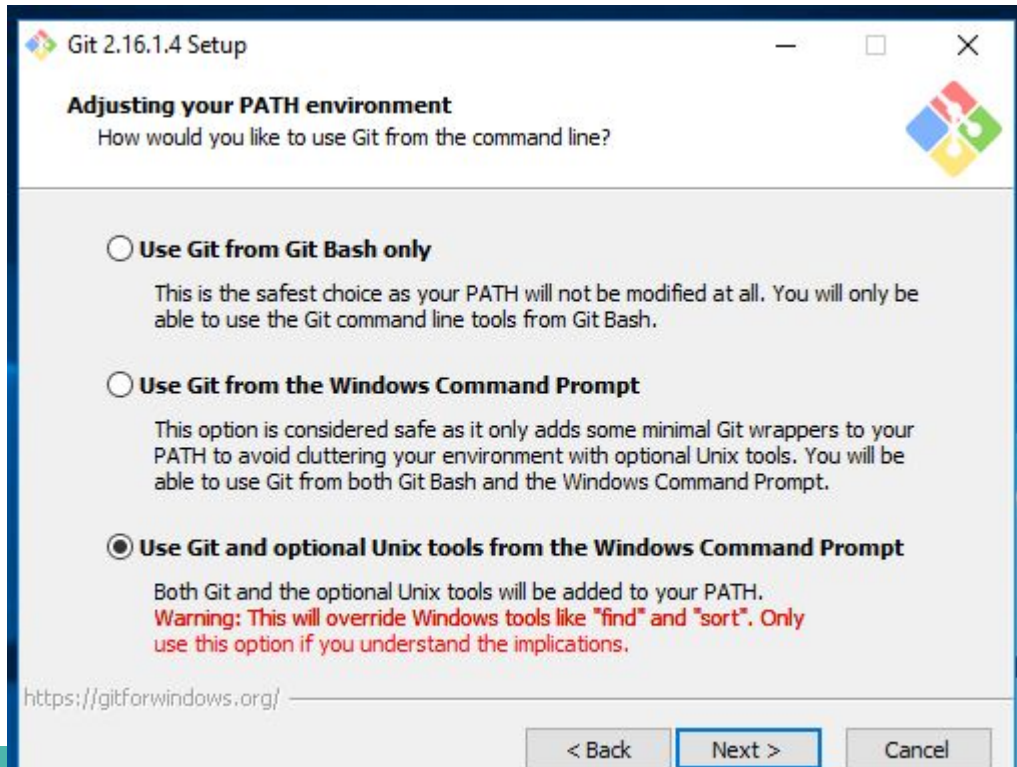
Windows

Step 5: Select editor of your choice and click next



Windows

Step 6: This dialogue asking you to how do you want to use git to set the PATH, go with the recommended option. **IMP - Select The First option - Git Bash only.**



Windows

Step 7,8,9 & 10: Leave the default options and proceed.

Step 11: Verification.

Open git bash and type 'git --version'

Step 12 : Configure your git username and email

- a. `git config --global user.name "LoneWolfKJ"`
- b. `git config --global user.email "kanjaldalal1000@gmail.com"`

Mac

Installing using Homebrew

1. `brew install git`
2. Verify the installation was successful by typing
 - a. `git --version`
3. Configure username and password
 - a. `git config --global user.name "LoneWolfKj"`
 - b. `git config --global user.email "kanjaldalal1000@gmail.com"`

Generating SSH key-pair

SSH uses a pair of keys to initiate a secure handshake between remote parties. The key pair contains a public and private key. The private vs public nomenclature can be confusing as they are both called keys. It is more helpful to think of the public key as a "lock" and the private key as the "key". You give the public 'lock' to remote parties to encrypt or 'lock' data. This data is then opened with the 'private' key which you hold in a secure place.

Generating SSH key-pair

1. Execute the following to begin the key creation
 - a. `ssh-keygen -t rsa -b 4096 -C "kanjaldalal1000@gmail.com"`

Press enter in the below prompt (ie- leave it as blank)

```
kanjal@LoneWolfKJ:~$ ssh-keygen -t rsa -b 4096 -C "kanjaldalal1000@gmail.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/kanjal/.ssh/id_rsa):
```

- b. The next prompt will ask for a secure passphrase.
 - c. > Enter passphrase (empty for no passphrase): [Type a passphrase]
 - d. > Enter same passphrase again: [Type passphrase again]

You can just leave it blank by simply pressing enter or enter some password (**you would need to remember this**)

Generating SSH key-pair

2. Before adding the new SSH key to the ssh-agent first ensure the ssh-agent is running by executing: (you should see an output like Agent pid 59566)

```
eval "$(ssh-agent -s)"
```

3. Once the ssh-agent is running the following command will add the new SSH key to the local SSH agent. (To verify you can run the command 'ssh-add -l')

```
ssh-add -k {your_home_directory}/.ssh/id_rsa
```

Eg - ssh-add -k /home/kanjal/.ssh/id_rsa

Using '~' (tilde) [if not able to find home directory]

```
ssh-add -k ~/.ssh/id_rsa
```

Thank You