**Introduction to Git and GitHub**

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.

Git was created by Linus Torvalds in 2005 for development of the Linux kernel, with other kernel developers contributing to its initial development. Its current maintainer since 2005 is Junio Hamano.

As with most other distributed version-control systems, and unlike most client–server systems, every Git directory on every computer is a full-fledged repository with complete history and full version-tracking abilities, independent of network access or a central server.

Git is free and open-source software distributed under the terms of the GNU General Public License version.

* 1. **Git History**

Git development began in April 2005, after many developers of the Linux kernel gave up access to BitKeeper, a proprietary source-control management (SCM) system that they had formerly used to maintain the project. The copyright holder of BitKeeper, Larry McVoy, had withdrawn free use of the product after claiming that Andrew Tridgell had reverse-engineered the BitKeeper protocols. (The same incident would also spur the creation of another version-control system, Mercurial.)

Linus Torvalds wanted a distributed system that he could use like BitKeeper, but none of the available free systems met his needs. Torvalds cited an example of a source-control management system needing 30 seconds to apply a patch and update all associated metadata, and noted that this would not scale to the needs of Linux kernel development, where synchronizing with fellow maintainers could require 250 such actions at once. For his design criteria, he specified that patching should take no more than three seconds, and added three more points:

* Take Concurrent Versions System (CVS) as an example of what not to do; if in doubt, make the exact opposite decision.
* Support a distributed, BitKeeper-like workflow.
* Include very strong safeguards against corruption, either accidental or malicious.

These criteria eliminated every then-extant version-control system, so immediately after the 2.6.12-rc2 Linux kernel development release, Torvalds set out to write his own.

The development of Git began on 3 April 2005. Torvalds announced the project on 6 April; it became self-hosting as of 7 April. The first merge of multiple branches took place on 18 April. Torvalds achieved his performance goals; on 29 April, the nascent Git was benchmarked recording patches to the Linux kernel tree at the rate of 6.7 patches per second. On 16 June Git managed the kernel 2.6.12 release

Torvalds turned over maintenance on 26 July 2005 to Junio Hamano, a major contributor to the project. Hamano was responsible for the 1.0 release on 21 December 2005 and remains the project's maintainer.

**1.2 Releases**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Original release date** | **Latest (patch) version** | **Release date (of patch)** |
| 0.99 | 2005-07-11 | 0.99.9n | 2005-12-15 |
| 1.0 | 2005-12-21 | 1.0.13 | 2006-01-27 |
| 1.1 | 2006-01-08 | 1.1.6 | 2006-01-30 |
| 1.2 | 2006-02-12 | 1.2.6 | 2006-04-08 |
| 1.3 | 2006-04-18 | 1.3.3 | 2006-05-16 |
| 1.4 | 2006-06-10 | 1.4.4.5 | 2008-07-16 |
| 1.5 | 2007-02-14 | 1.5.6.6 | 2008-12-17 |
| 1.6 | 2008-08-17 | 1.6.6.3 | 2010-12-15 |
| 1.7 | 2010-02-13 | 1.7.12.4 | 2012-10-17 |
| 1.8 | 2012-10-21 | 1.8.5.6 | 2014-12-17 |
| 1.9 | 2014-02-14 | 1.9.5 | 2014-12-17 |
| 2.0 | 2014-05-28 | 2.0.5 | 2014-12-17 |
| 2.1 | 2014-08-16 | 2.1.4 | 2014-12-17 |
| 2.2 | 2014-11-26 | 2.2.3 | 2015-09-04 |
| 2.3 | 2015-02-05 | 2.3.10 | 2015-09-29 |
| 2.4 | 2015-04-30 | 2.4.12 | 2017-05-05 |
| 2.5 | 2015-07-27 | 2.5.6 | 2017-05-05 |
| 2.6 | 2015-09-28 | 2.6.7 | 2017-05-05 |
| 2.7 | 2015-10-04 | 2.7.6 | 2017-07-30 |
| 2.8 | 2016-03-28 | 2.8.6 | 2017-07-30 |
| 2.9 | 2016-06-13 | 2.9.5 | 2017-07-30 |
| 2.10 | 2016-09-02 | 2.10.5 | 2017-09-22 |
| 2.11 | 2016-11-29 | 2.11.4 | 2017-09-22 |
| 2.12 | 2017-02-24 | 2.12.5 | 2017-09-22 |
| 2.13 | 2017-05-10 | 2.13.7 | 2018-05-22 |
| 2.14 | 2017-08-04 | 2.14.5 | 2018-09-27 |
| 2.15 | 2017-10-30 | 2.15.3 | 2018-09-27 |
| 2.16 | 2018-01-17 | 2.16.5 | 2018-09-27 |
| 2.17 | 2018-04-02 | 2.17.2 | 2018-09-27 |
| 2.18 | 2018-06-21 | 2.18.1 | 2018-09-27 |
| 2.19 | 2018-09-10 | 2.19.2 | 2018-11-21 |
| 2.20 | 2018-12-09 | 2.20.1 | 2018-12-15 |
| **2.21** | 2019-02-24 | 2.21.0 | 2019-02-24 |

* 1. **What is version control?**

Version control systems are software that help you track changes you make in your code over time. As you edit to your code, you tell the version control system to take a snapshot of your files. The version control system saves that snapshot permanently so you can recall it later if you need it.

Without version control, you’re tempted to keep multiple copies of code on your computer. This is dangerous-it’s easy to change or delete a file in the wrong copy of code, potentially losing work. Version control systems solve this problem by managing all versions of your code but presenting you with a single version at a time.

* 1. **What is Distributed Version Control System?**

In software development, distributed version control (also known as distributed revision control) is a form of version control where the complete codebase - including its full history - is mirrored on every developer's computer. This allows branching and merging to be

managed automatically, increases speeds of most operations (except for pushing and pulling), improves the ability to work offline, and does not rely on a single location for backups.

In 2010, software development author Joel Spolsky, described DVCS as "possibly the biggest advance in software development technology in the past ten years.

* 1. **What is Git?**

Git is the most commonly used version control system today and is quickly becoming the standard for version control. Git is a distributed version control system, meaning your local copy of code is a complete version control repository. These fully-functional local repositories make it is easy to work offline or remotely. You commit your work locally, and then sync your copy of the repository with the copy on the server. This paradigm differs from centralized version control where clients must synchronize code with a server before creating new versions of code.

Git’s flexibility and popularity make it a great choice for any team. Many developers and college graduates already know how to use Git. Git’s user community has created many resources to train developers and Git’s popularity make it easy to get help when you need it. Nearly every development environment has Git support and Git command line tools run on every major operating system.

* 1. **What is Git for?**

The version control system has the following benefits:

• You have access to all versions of all files in Git repository at any time, it’s almost impossible to lose any part of a code.

• Multiple developers can work on one project at the same time without interfering with each other, and without fear of losing any changes made by a colleague. In Git, the possibilities of collaborative work are unlimited.

You will have to use Git every day, and this is a tool you should have a perfect command of.

Be sure to sign up on GitHub or Bitbucket. These services offer great manuals on how to start working with remote repositories. I like bitbucket for the possibility of creating unlimited number of private repositories. However, in terms of all other characteristics, GitHub is way ahead of bitbucket, and has turned into the standard and Mecca of the whole open source community.

**1.6 Install and setup Git**

Git isn’t usually set up by default on your computer, so you need to install and configure it before you can start using it to manage your code. It’s important to keep Git up to date, just like all the other software on your machine. Updates protect you from security vulnerabilities, fix bugs, and give you access to new features.

The recommended method of installing and maintaining Git is provided for three major platforms below.

Windows

Download and install Git for Windows. Once installed, you’ll be able to use Git from the command prompt or PowerShell. We recommend that you stick with the defaults selected during the installation unless you have a good reason to change them.

Git for Windows does not automatically update. Update Git for Windows by downloading a new version of the installer, which will update Git for Windows in place and keep all of your settings.

macOS

macOS 10.9 (Mavericks) or higher will install Git the first time you try to run Git from the Terminal. This is an easy way to get Git on your system, but you can’t control exactly how often you get updates or security fixes for Git.

We recommend installing Git through Homebrew and using the Homebrew tools to keep Git up to date. Homebrew is a great way to install and manage open source development tools on your Mac from the command line.

Install Homebrew and run the following to install an up to date version of Git on your Mac:

> brew install git

To update your Git install, use Homebrew’s upgrade option:

> brew upgrade git

A graphical installer for Git on macOS is also available from the official Git website.

Linux

Use your Linux distribution’s package management system to install and update Git. For example, on Ubuntu:

> sudo apt-get install git

Configure Git

Set up your name and email address before starting to work with Git on your computer. Git attaches this information to your changes and lets others identify which changes are yours.

Run the following commands from the command prompt after installing Git to configure this information:

> git config --global user.name "AnujChourange"

> git config --global user.email "chourangeanuj@gmail.com"

* 1. **What is GitHub?**

In layman terms you can think of GitHub as a social network of source codes and by that means here you can host all the projects you have made online with facility to make them private or public . Each project that you host can have one or many files .

GitHub makes Git easier to use in two ways. First, if you download the GitHub software to your computer, it provides a visual interface to help you manage your version-controlled projects locally. Second, creating an account on Build software better, together brings your version-controlled projects to the Web, and ties in social network features for good measure.

The power of GitHub is that is facilitates collaborative work. That means suppose i have a repository on my GitHub account. So now any one can just clone it (NB: Cloning creates a exact copy of project on local machine i.e. your laptop ) and make changes to the existing source codes i.e. add something new or delete something or make any changes to the source code.

* 1. **Difference Between Git and GitHub.**

As told, Git is a version control system which tracks the changes when working with computer codes while GitHub is a Web-based Git version control repository hosting service. It provides all of the distributed version control and source code management (SCM) functionalities of Git while topping it with a few of its own features. It is a heaven for the developers where they can store their projects and get connected with like-minded people. You can think it as “cloud for codes”.

So basically it is a place to store your identical working directories or repositories. It literally is a hub for the Git repositories which you can use, simply by creating a free account on the GitHub. These accounts come with an abundant storage space where you can store your repositories and build a proper profile which holds a great value. By default the repositories are public i.e., everyone can see your codes but you can make it private as well. So if you are a good coder you can upload your own applications and programs and get recognition from others.

|  |  |
| --- | --- |
| git logo - Git vs GitHub - Edureka | GitHub logo - Git vs GitHub - Edureka |
| 1. It is a software | 1. It is a service |
| 2. It is installed locally on the system | 2. It is hosted on  Web |
| 3. It is a command line tool | 3. It provides a graphical interface |
| 4. It is a tool to manage different versions of edits, made to files in a git repository | 4. It is a space to upload a copy of the **Git** repository |
| 5. It provides functionalities like Version Control System Source Code Management | 5. It provides functionalities of Git like VCS, Source Code Management as well as adding few of its own features |

* 1. **Summary**