# 



GitHub - "Build software better, together."



# [GitHub mascot/logo]

Download and install GitHub desktop: <https://desktop.github.com/>

GitHub Desktop is a seamless way to contribute to projects on [GitHub](https://github.com/).

# Getting Started with GitHub Desktop: https://help.github.com/desktop/guides/getting-started/

**GitHub Desktop Documentation:** <https://help.github.com/desktop/>

## Your GitHub workflow in one native app

Clone repositories

Create branches

Commit changes

Share code

# What is Git/Github?

Git is the distributed version control system. Git is responsible for keeping track of changes to content (usually source code files + other documents), and it provides mechanisms for sharing that content with others.

GitHub is a company that provides Git repository hosting. GitHub Inc. also developed graphical Git clients: GitHub for Mac and GitHub for Windows. Each is an application that lets you interact with Git repositories without using the command line.

Git is primarily developed for Linux and has the highest speeds on there.

**Concepts from Git**: Repositories, clone/fork, branches, remotes, committing, pushing, pulling, merging, rebasing, reverting, and cherry-picking.

**Concepts from GitHub**: Pull requests, issues, wikis, Gists, [github.com](https://github.com/), fork

# Why GitHub?

When multiple people work on a single project **without a version control system in place** things get chaotic. Version Control (also known as Revision Control or Source Control Management) is a great way to solve the file sharing problem. Version control allows individual programmers or project managers tackle a project from different angles without getting in each other’s way and without doing damage that can’t be undone. **Even if you don’t work with a team, version control is also a lifesaver.**

GitHub Pages created after June 15, 2016 and using github.io domains are served over HTTPS. For more information, see "[Securing your GitHub Pages site with HTTPS](https://help.github.com/articles/securing-your-github-pages-site-with-https)."

### [Intro to Git for Web Designers](http://www.webdesignerdepot.com/2009/03/intro-to-git-for-web-designers/)

# <http://www.webdesignerdepot.com/2009/03/intro-to-git-for-web-designers/>

This Git tutorial for Web designers is geared towards web designers. It outlines the benefits of version control in the context of creating websites and assumes the reader prefers a GUI for working with Git instead of the traditional command-line interface.

One of Git’s big differentiators is that unlike SVN and CVS it is a [**distributed version control system**](http://en.wikipedia.org/wiki/Distributed_revision_control). This means that every user has a complete copy of the repository data stored locally on their machine. What’s so great about that? A few things:

* **Everything is local**, so you can work offline
* There is **no single point of failure**. It doesn’t rely on one central server that could crash and burn, taking the only repository for your project with it.
* Because it doesn’t have to communicate with a central server constantly, **processes run much faster**

# Top 10 Git Tutorials for Beginners

http://sixrevisions.com/resources/git-tutorials-beginners/

Github Tutorial For Beginners - Github Basics for Mac or Windows & Source Control Basics:

<https://www.youtube.com/watch?v=0fKg7e37bQE>

This video has over 1 million views.

Published on 16 Jan 2014

Github Tutorial For Beginners - learn Github for Mac or Github for windows  
If you've been wanting to learn Github, now's the perfect time! Github is seen as a big requirement by most employers these days and is very critical to business workflow. This Github tutorial will cover the basics of how to use Github and the command line.

GITHUB PULL REQUEST, Branching, Merging & Team Workflow

<https://www.youtube.com/watch?v=oFYyTZwMyAg>

Github Pull Requests are an integral part of Team Workflow. This video covers how to make a new Github feature branch, merge that branch, handle merge conflicts, submit a pull request, discuss the github pull request, and ultimately merge that pull into the master branch of your github repository.

How to use GitHub for Beginners

<https://www.youtube.com/watch?v=E8TXME3bzNs>

HOW TO DO THE GITHUB (The Absolute Basics)

<https://www.youtube.com/watch?v=EUvmCuPjHD4>

**Published on 2 Mar 2016**

This is also a good read if my video isn't your cup of tea: [https://guides.github.com/activities/...](https://guides.github.com/activities/hello-world/)  
The GUI application: <https://desktop.github.com/>  
Community Challenge Information: [https://www.reddit.com/r/TheHappieMak...](https://www.reddit.com/r/TheHappieMakers/)

How to Get Started with Github - Beginner Tutorial

<https://www.youtube.com/watch?v=73I5dRucCds>

Lecture style tutorial for beginners wanting to learn how to use Github with the Github bash

# What is GitHub?

<https://en.wikipedia.org/wiki/GitHub>

Git is a **free open-source Version Control System.**

**GitHub** is a web-based tool for version control ([source code management](https://en.wikipedia.org/wiki/Source_code_management" \o "Source code management)) and team collaboration for software development. It also offers [Internet hosting service](https://en.wikipedia.org/wiki/Internet_hosting_service). Github is mainly for free OpenSource projects but it also offers plans for private projects.

 GitHub is a [**distributed version control system**](http://en.wikipedia.org/wiki/Distributed_revision_control). This means that every user has a complete copy of the repository data stored locally on their machine. Therefore, users can work offline and there is no single point of failure.

It provides [access control](https://en.wikipedia.org/wiki/Access_control) and several collaboration features such as [bug tracking](https://en.wikipedia.org/wiki/Bug_tracking_system), [feature requests](https://en.wikipedia.org/wiki/Software_feature), [task management](https://en.wikipedia.org/wiki/Task_management), and [wikis](https://en.wikipedia.org/wiki/Wiki) for every project.

GitHub is mostly used for code. However, it also supports the following formats and features:

* Documentation
* Small website hosting: The URL format is GitHub pages is: http://*username*.github.io
* Issue tracking

Note: by default, it is public. Public repository is free. Do make sure you make your project private. There is small fees associated with private repository (less than $10 per month).

Git/GitHub Terms:

1. Repository : a project
2. Commit: submit a working version
3. Cloning vs

GitHub help: https://help.github.com/categories/bootcamp/

* [Set Up Git](https://help.github.com/articles/set-up-git)
* [Create A Repo](https://help.github.com/articles/create-a-repo)
* [Fork A Repo](https://help.github.com/articles/fork-a-repo)
* [Be Social](https://help.github.com/articles/be-social)

## Your GitHub workflow in one native app

What you need to do:

1. Create a GitHub account (free)

* Sign up an account here: go to GitGub website: https://github.com/

1. Set up Git (download and install GitHub): <https://help.github.com/articles/set-up-git/>

For Windows: <https://desktop.github.com/> OR <https://git-for-windows.github.io/>

For Mac:

Just follow the installer instructions.

1. Start a project/create a new repository (or copy/clone an existing one from GitHub directly)

* Use GUI (GitHub desktop): create a new project/folder (top-left corner +), then copy your existing files there, then Publish/push
* Use Git Shell: Using the command line to turn an existing folder into a Git repository:
  + cd path/to/project (example: cd H:\Documents\GitHub\test)
  + git init
  + git add .
  + git commit
* **Initialize** this directory to make it a GitGub repository
* **Add everything in it** – all files and subdirectories
* **Commit or store**, all current changes in the repository

You can use the GUI (if you are a Windows/Mac lover) or the command line (Git Bash) if you are a Linux fan.

I prefer the GUI because it is very easy to use- it is visual so you do not need to remember all the commands. If you want to be a real professional developer, you can use the command line. There are only 10 commands you use frequently.

GitHub Desktop is a seamless way to contribute to projects on [GitHub](https://github.com/)

Everyday workflow when using GitHub:

1. **Open a Terminal**.
2. Use the **“git pull”**command to get the latest changes in master repository.
3. Open the project in **a test editor** and make changes.
4. **Commit changes and add notes**. This only commits them locally and you can do this multiple times per day.
5. Use **“git push”** to push **the changes**to the master repository so that other team members can see them. You should do this at least once a day or after any major addition.

GitHub concepts: merging, squashing, rebasing

When merging pull requests, you can allow any combination of merge commits, squashing, or rebasing. At least one option must be enabled.

Allow merge commits 

Add all commits from the head branch to the base branch with a merge commit.

Allow squash merging 

Combine all commits from the head branch into a single commit in the base branch.

Allow rebase merging 

Add all commits from the head branch onto the base branch individually.

# Git commands:

pull, push, clone, add, commit, status,

$git pull

$git status

$git add

$git commit

Your Workflow when using GitHub:

1. Git pull -

**Use Git Shell – your favouriate Shell with Git enabled.**

CVS (GNU license), SVN (Apache license), Git, are all open source software versioning and revision control system for source code, web pages, and documentation.

## [2017 Version Control Software Comparison: SVN, Git, Mercurial - Biz 3.0](https://biz30.timedoctor.com/git-mecurial-and-cvs-comparison-of-svn-software/)

**https://biz30.timedoctor.com/git-mecurial-and-cvs-comparison-of-svn-software/**

**Other version control software:**

* CVS (Concurrent Versions System): free, open source, version control system.  Oldest/most mature version control software and it is fading out.
* SVN (Apache Subversion ): Newer system based on CVS (it is a newer technology that takes the best features of CVS and improves upon them and fixed some problems of CVS). Both CVS and SVN use client/server architecture (all repositories are on the central server). Many developers have switched to SVN. SVN is currently the king of server-based version control. TortoiseSVN is the GUI client for SVN.
* **Git: GitHub uses fully distributed, peer-to-peer architecture. Allows developers to work offline. Powerful and fast. Git is more difficult to learn than SVN/Mercurial.**
* Mercurial: uses **fully distributed, peer-to-peer architecture but it has interfaces similar to SVN so it is easier to learn. Good for python programmers as it was written in Python.**

All of these systems are fully functional. They’re also all free. They’ve all been used to create software, websites, and even operating systems that you’ve used and heard of.

+++++++++++++++++++++++

## Start a project

You’ll find all the projects you’re working on listed in the sidebar. If you’re starting a new project, use the repository drop down menu to create a new repository or clone an existing one directly from GitHub.com.

[**https://desktop.github.com/**](https://desktop.github.com/)

## Branch off

 Just select the current branch to switch branches or create a new one.