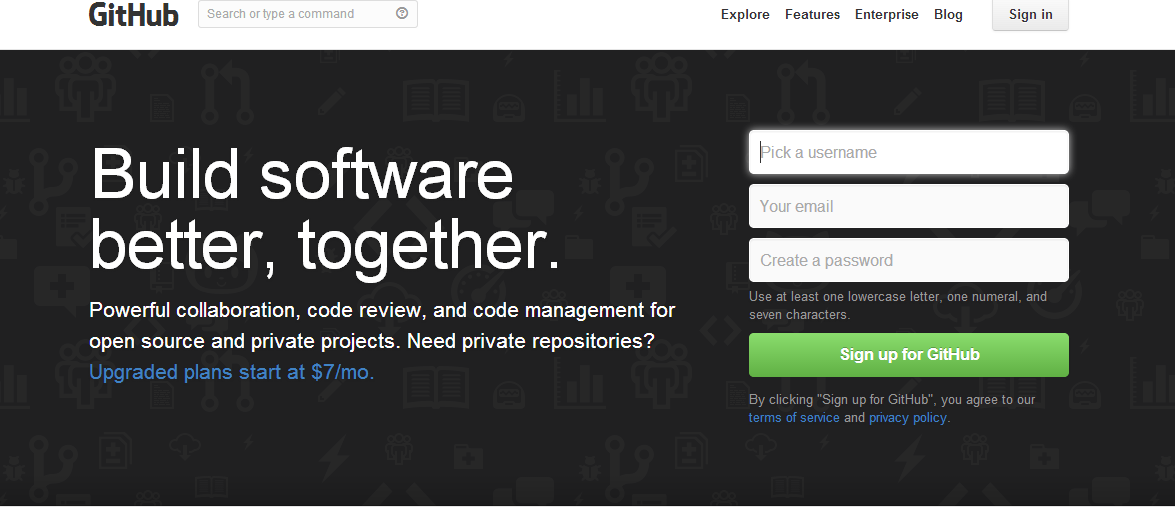
GIT HUB

[www.github.com](http://www.github.com) - Register to GIT HUB

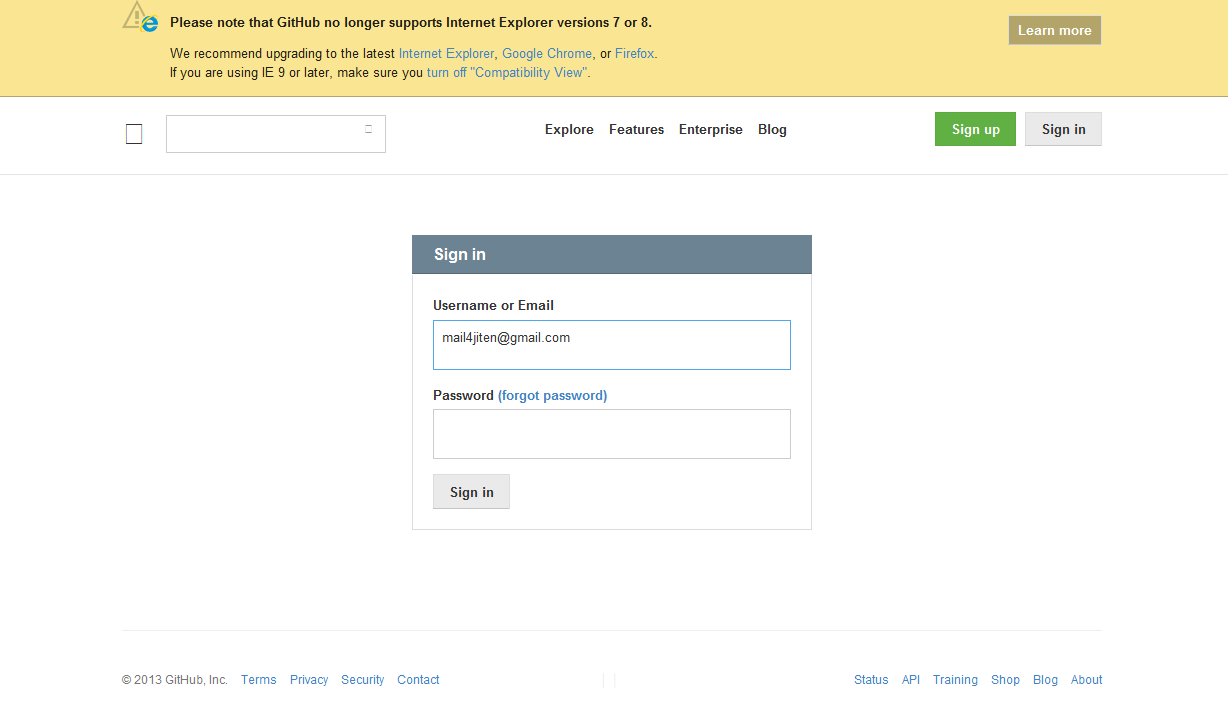


Installation Type 1 :

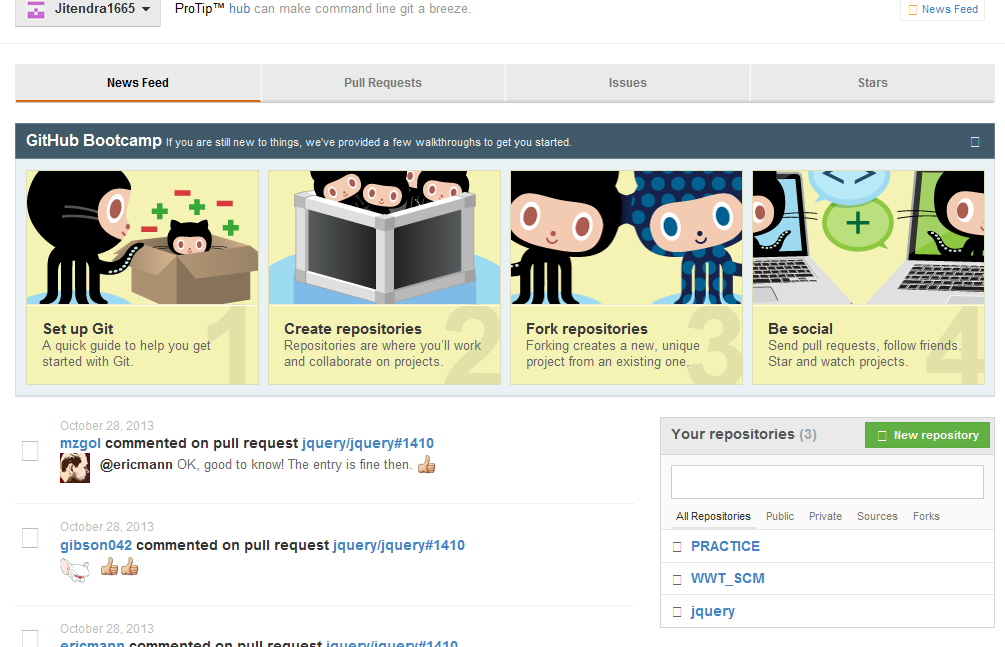
Steps 1 :

Follow the below mentioned steps or follows the Steps given under **– Local GIT Installation**

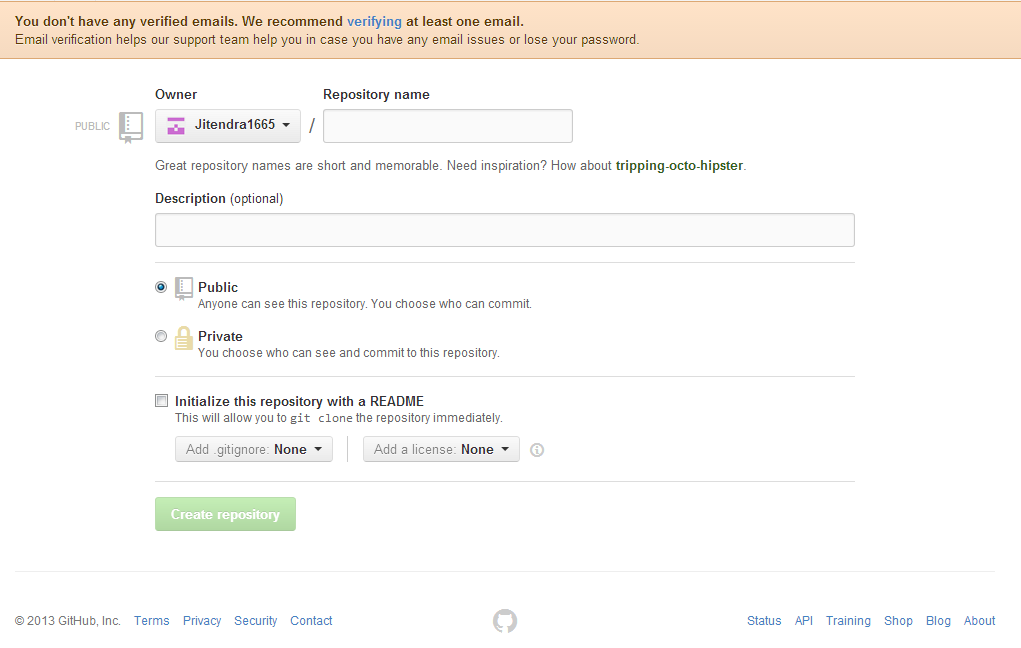
Sign In - user id/Password = [mail4jiten@gmail.com](mailto:mail4jiten@gmail.com)/dv0y2h@1



Click on New Repository



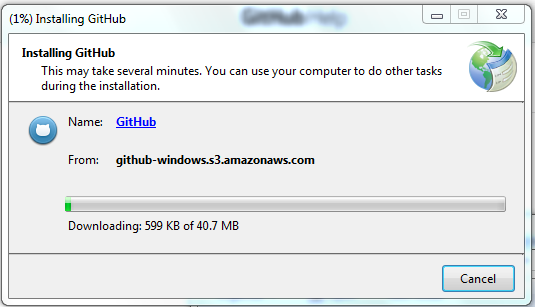
Select Public and give the new Repository Name

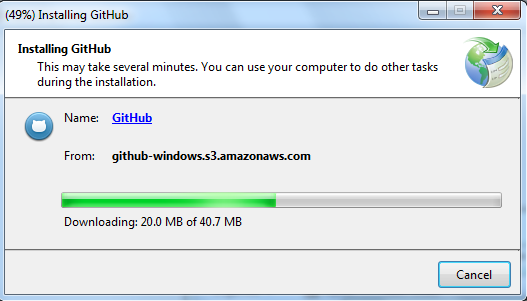


**Installation Type 2 -**

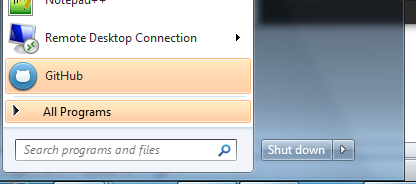
**Local GIT Installation -**

1. <https://help.github.com/articles/set-up-git> = Install the GITHUB Set Up for Windows
   1. Install – GIT HUB in the system.

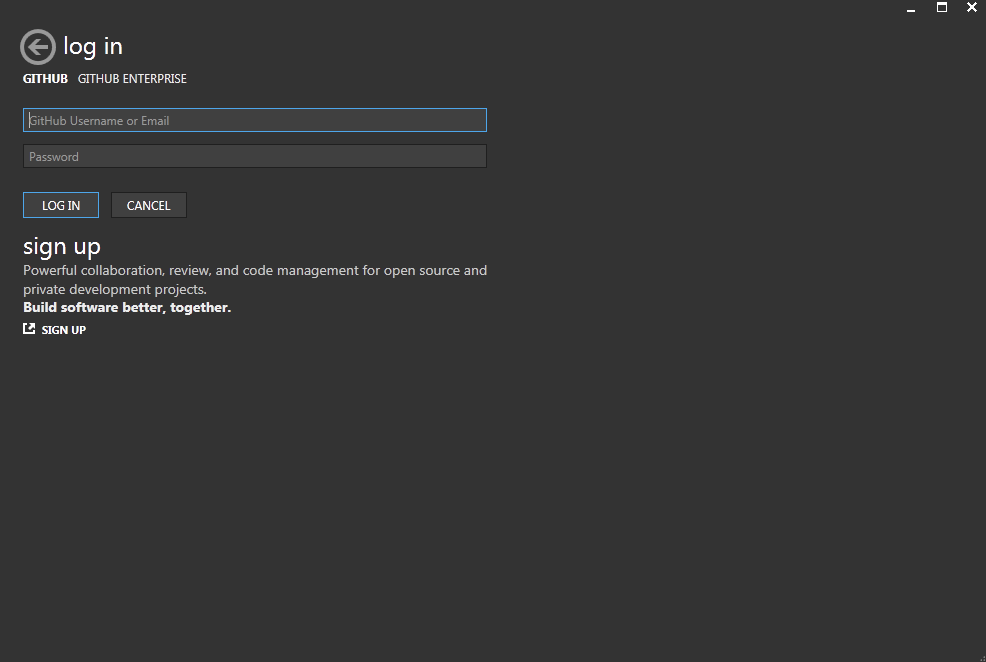




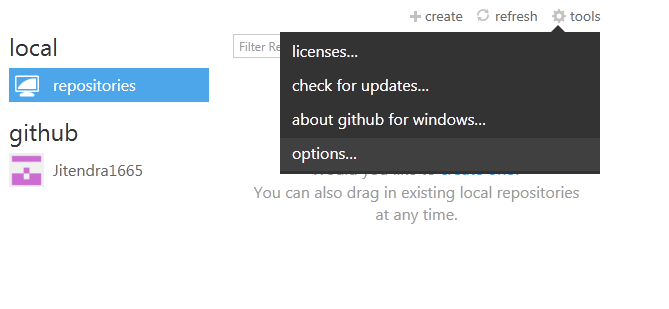
After TYPE 2 – Installation – you can find the GIT HUB in Window -> start as follows



Click on GitHub and login to the GIT HUB using the credentials created in step 1 of this document.

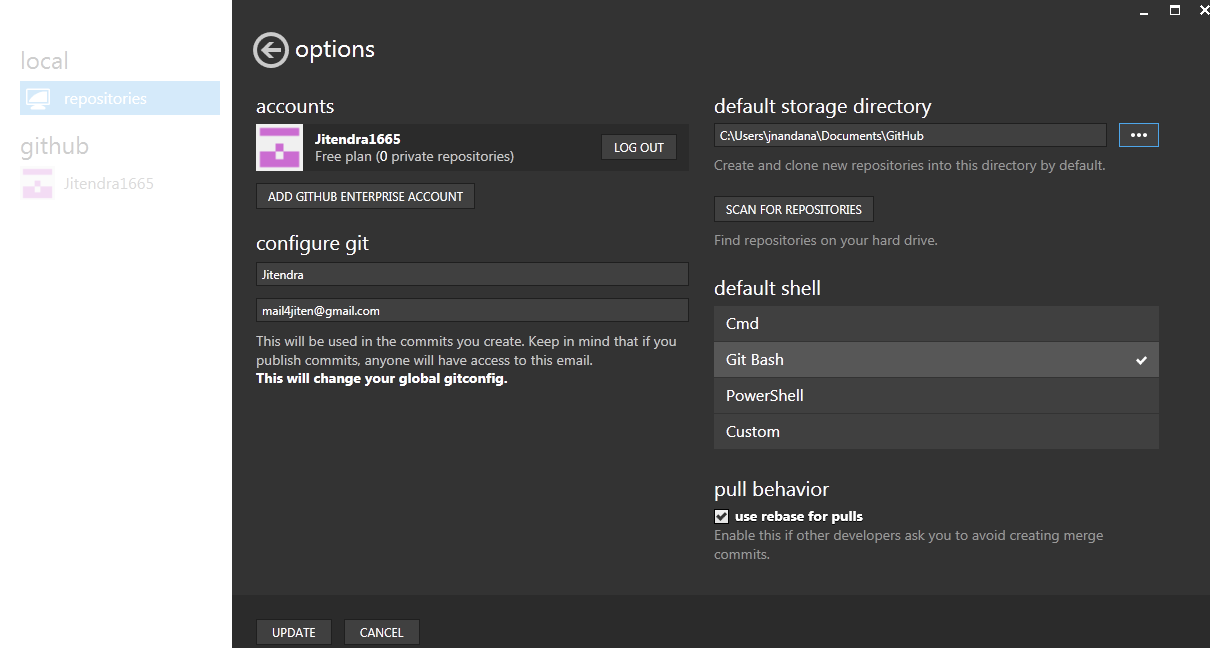


Navigate to tools -> Options



Note down the –

Default documents storage location - [file:///C:/Users/jnandana/Documents/GitHub](file:///C:\Users\jnandana\Documents\GitHub)



Install the GIT

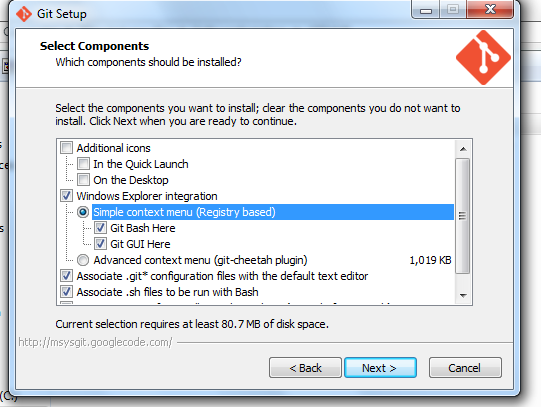
[Preview](http://git-scm.com/download/win) from

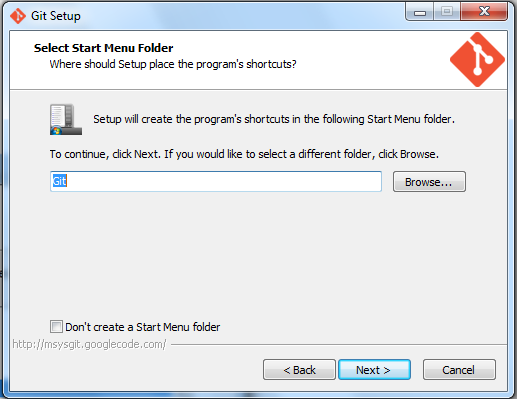
<http://git-scm.com/download/win> -

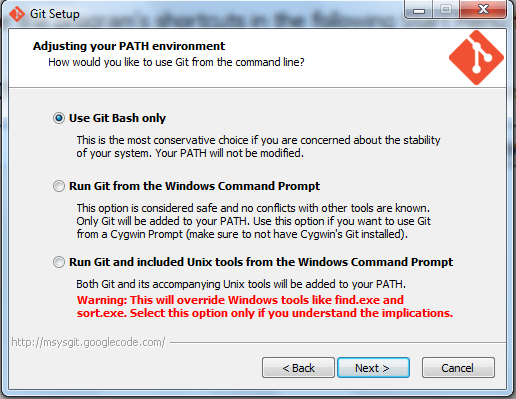
or

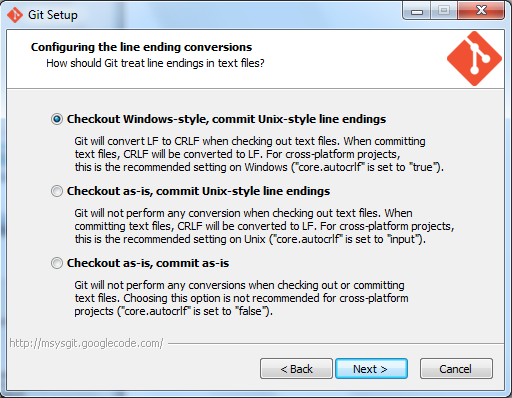
from google code site - <https://code.google.com/p/msysgit/downloads/detail?name=Git-1.8.4-preview20130916.exe&can=2&q=>

**While installing select the check box for – Windows Explorer Integration. (Simple Context menu(Registry based).**





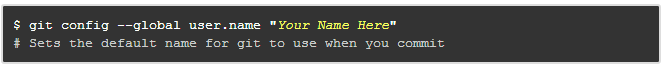




**Set Up Git**

Now that you have Git installed, it's time to configure your settings. To do this you need to open Git Bash (not the Windows command line).

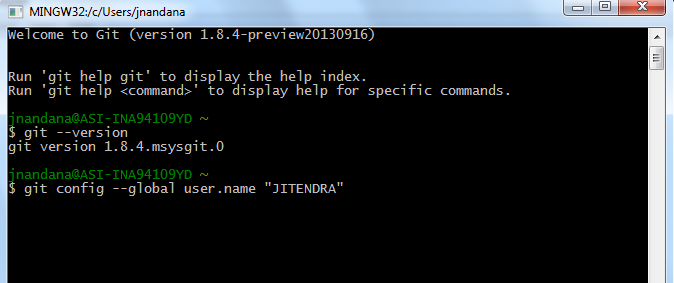
**Username**

First you need to tell git your name, so that it can properly label the commits you make.

**As follows :**

git config --global user.name "Your Name Here"

# Sets the default name for git to use when you commit



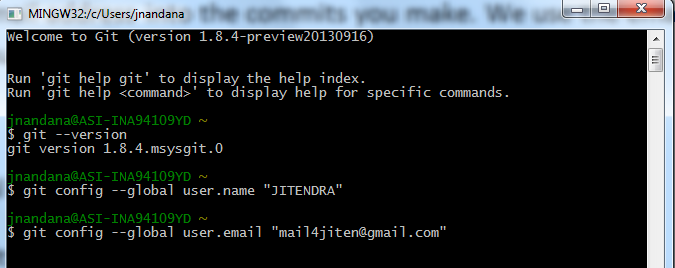
**Email**

Git saves your email address into the commits you make. We use the email address to associate your commits with your GitHub account.

**As Follows :**

git config --global user.email "your\_email@example.com"

# Sets the default email for git to use when you commit



Your email address for Git should be the same one associated with your GitHub account.

If it is not, see this guide for help adding additional emails to your GitHub account.

If you want to keep your email address hidden, this guide may be useful to you.

**Password caching**

The last option we need to set will tell git that you don't want to type your username and password every time you talk to a remote server.

Tip: You need git 1.7.10 or newer to use the credential helper

To use this option, you need install a credential helper.

GitHub for Windows includes this helper, and provides a git shell so you don't need to install and configure git manually.

If you don't want to use GitHub for Windows, you can download the helper for your OS here:

**Windows Vista, 7, & 8 (.NET 4.0 required)**

**Source**

Unzip the file and run the git-credential-winstore.exe program inside. This will start up the helper and update your git config to use it.

**Tip: The credential helper only works when you clone an HTTPS repository URL. If you use the SSH repository URL instead, SSH keys are used for authentication. This guide offers help generating and using an SSH key pair.**

**Celebrate**

Congratulations, you now have Git and GitHub all set up! What do you want to do next?

**Set Up Git**

Create A Repository

Fork A Repository

**Be Social**

Overriding settings in individual repos

**Fork A Repo**

If you've found yourself on this page, we're assuming you're brand new to Git and GitHub. This guide will walk you through the basics and explain a little bit about how everything works along the way.

Contributing to a project

==========================================================================================================================

At some point you may find yourself wanting to contribute to someone else's project, or would like to use someone's project as the starting point for your own. This is known as "forking". For this tutorial, we'll be using the Spoon-Knife project, hosted on GitHub.com.

===============

Step 1: Fork the "Spoon-Knife" repository

To fork this project, click the "Fork" button in the GitHub.com repository.

Click "Fork"

Step 2: Clone your fork

You've successfully forked the Spoon-Knife repository, but so far it only exists on GitHub. To be able to work on the project, you will need to clone it to your local machine.

Run the following code:

git clone https://github.com/username/Spoon-Knife.git

# Clones your fork of the repository into the current directory in terminal

Step 3: Configure remotes

When a repository is cloned, it has a default remote called origin that points to your fork on GitHub, not the original repository it was forked from. To keep track of the original repository, you need to add another remote named upstream:

More about remotes

cd Spoon-Knife

# Changes the active directory in the prompt to the newly cloned "Spoon-Knife" directory

git remote add upstream https://github.com/octocat/Spoon-Knife.git

# Assigns the original repository to a remote called "upstream"

git fetch upstream

# Pulls in changes not present in your local repository, without modifying your files

More Things You Can Do

You've successfully forked a repository, but get a load of these other cool things you can do:

Push commits

Once you've made some commits to a forked repository and want to push it to your forked project, you do it the same way you would with a regular repository:

More about commits

git push origin master

# Pushes commits to your remote repository stored on GitHub

Pull in upstream changes

If the original repository you forked your project from gets updated, you can add those updates to your fork by running the following code:

git fetch upstream

# Fetches any new changes from the original repository

git merge upstream/master

# Merges any changes fetched into your working files

What is the difference between fetch and pull?

Create branches

Branching allows you to build new features or test out ideas without putting your main project at risk. In git, branch is a sort of bookmark that references the last commit made in the branch. This makes branches very small and easy to work with.

How do I use branches?

Pull requests

If you are hoping to contribute back to the original fork, you can send the original author a pull request.

Unwatch the main repository

When you fork a particularly popular repository, you may find yourself with a lot of unwanted updates about it. To unsubscribe from updates to the main repository, click the "Unwatch" button on the main repository and select "Not Watching".

Click "Unwatch"

Delete your fork

At some point you may decide that you want to delete your fork. To delete a fork, just follow the same steps as you would to delete a regular repository.

Celebrate

You have now forked a repository. What do you want to do next?

Set Up Git

Create A Repository

Fork A Repository

Be Social