

# Git and GitHub-Intro

- **What is Git?**

So, what is Git in a nutshell? This is an important section to absorb, because if you understand what Git is and the fundamentals of how it works, then using Git effectively will probably be much easier for you. As you learn Git, try to clear your mind of the things you may know about other VCSs, such as CVS, Subversion or Perforce — doing so will help you avoid subtle confusion when using the tool. Even though Git's user interface is fairly similar to these other VCSs, Git stores and thinks about information in a very different way, and understanding these differences will help you avoid becoming confused while using it.

## **Snapshots, Not Differences**

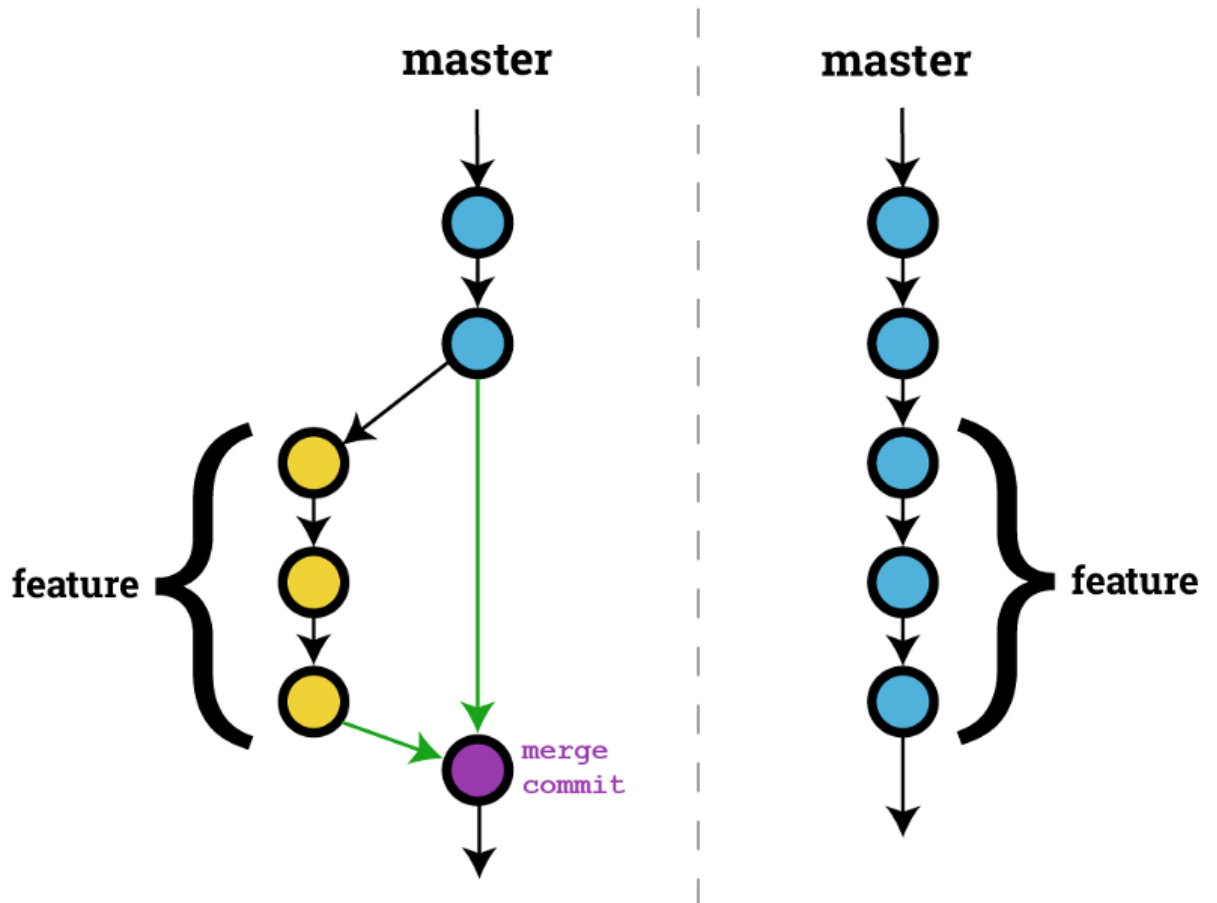
The major difference between Git and any other VCS (Subversion and friends included) is the way Git thinks about its data. Conceptually, most other systems store information as a list of file-based changes. These other systems (CVS, Subversion, Perforce, Bazaar, and so on) think of the information they store as a set of files and the changes made to each file over time (this is commonly described as **delta-based** version control).

- **Some key points**

1. Distributed Version Control.
2. Coordinate work between multiple developers.
3. Who made what changes and when.
4. Revert at any file.
5. Local and remote repository

- **Concept of git**

1. Keep track of code history.
2. Take **Snapshot** of your file.
3. You decide when to take a snapshot by making commit.
4. You can visit any snapshot at any time.
5. You can stage file before committing.



- **How to install git?**

1. Linux (Debian):

Run this command in terminal:

```
sudo apt-get install git
```

2. Linux (Fedora):

Run this command in Terminal:

```
sudo yum install git
```

3. Mac:

Visit:

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

4. Windows:

Visit:

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

- **Commands that we practiced**

git init	//initialize the git
git config --global user.name 'Username'	//configure the username
git config --global user.email 'email'	//configure the email
git add <filename>	//add file to index
git status	//check status
git commit	// commit changes in index
git push	// push to remote repository
git pull	//pull latest from remote repos
git clone	//clone repository into a new directory
git branch	//create or get name of branch
git checkout <branchname>	//checkout to another branch
git merge <branchname>	//merge the branch with base branch
git log	//to check the logs
git log --oneline	//oneline commits

- **How to write comments in vim editor**

1. First of all press 'I' in the keyboard to insert an comment (it will allow you to write something in the editor)
2. After that press 'Esc' to exit insert mode
3. Now write ':wq' to exit Vim editor.

# GITHUB

GitHub is a web-based version-control and collaboration platform for software developers.

GitHub facilitates [social coding](#) by providing a web interface to the Git [code repository](#) and management tools for collaboration. GitHub can be thought of as a serious [social networking](#) site for software developers. Members can follow each other, rate each other's work, receive updates for specific projects and communicate publicly or privately.

Three important terms used by developers in GitHub are fork, pull request and merge. A *fork*, also known as a *branch*, is simply a repository that has been copied from one member's account to another member's account. Forks and branches allow a developer to make modifications without affecting the original code. If the developer would like to share the modifications, she can send a *pull request* to the owner of the original repository. If, after reviewing the modifications, the original owner would like to pull the modifications into the repository, she can accept the modifications and *merge* them with the original repository. Commits are, by default, all retained and interleaved onto the master project, or can be combined into a simpler merge via commit squashing.

## [Setting up Git](#)

1. [Download and install the latest version of Git.](#)
2. [Set your username in Git.](#)
3. [Set your commit email address in Git.](#)

## [Next steps: Authenticating with GitHub from Git](#)

When you connect to a GitHub repository from Git, you'll need to authenticate with GitHub using either HTTPS or SSH.

### Connecting over HTTPS (recommended)

If you [clone with HTTPS](#), you can [cache your GitHub password in Git](#) using a credential helper.

### Connecting over SSH

If you [clone with SSH](#), you must [generate SSH keys](#) on each computer you use to push or pull from GitHub.

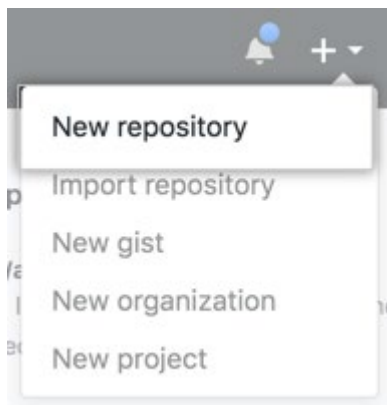
## Create a repo

---

To put your project up on GitHub, you'll need to create a repository for it to live in.

You can store a variety of projects in GitHub repositories, including open source projects. With [open source projects](#), you can share code to make better, more reliable software.

In the upper-right corner of any page, use the drop-down menu, and select **New repository**.



Type a short, memorable name for your repository. For example, "hello-world".

### Create a new repository

A repository contains all the files for your project, including the revision history.

Owner



Repository name

hello-world




Great repository names are short and memorable. Need inspiration? How about [potential-eureka](#).

Description (optional)

Optionally, add a description of your repository. For example, "My first repository on GitHub."

## Create a new repository

A repository contains all the files for your project, including the revision history.


Owner	Repository name
 octocat ▾	/ hello-world ✓

Great repository names are short and memorable. Need inspiration? How about **potential-eureka**.

### Description (optional)

My first repository on GitHub



Choose to make the repository either public or private. Public repositories are visible to the public, while private repositories are only accessible to you, and people you share them with. For more information, see "[Setting repository visibility](#)."

Owner	Repository name
 octocat ▾	/ hello-world ✓

Great repository names are short and memorable. Need inspiration? How about **potential-eureka**.

### Description (optional)

My first repository on GitHub

- ☒  **Public**  
Anyone can see this repository. You choose who can commit.
- ☐  **Private**  
You choose who can see and commit to this repository.

Click **Create repository**.

## Fork a repo

A *fork* is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project.

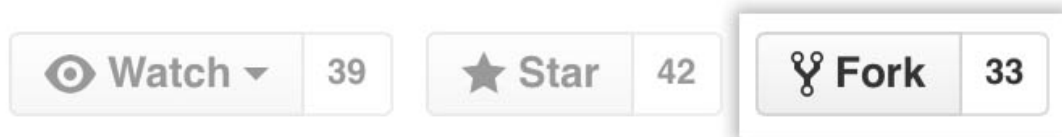
Most commonly, forks are used to either propose changes to someone else's project or to use someone else's project as a starting point for your own idea.

## Propose changes to someone else's project

A great example of using forks to propose changes is for bug fixes. Rather than logging an issue for a bug you've found, you can:


- Fork the repository.
- Make the fix.
- Submit a *pull request* to the project owner.

If the project owner likes your work, they might pull your fix into the original repository!



## ADDING IN YOUR NEW REPO

### Quick setup — if you've done this kind of thing before


 Set up in Desktop

 or 

HTTPS

SSH

https://github.com/seemantaggarwal/test.git



Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

### ...or create a new repository on the command line

```
echo "# test" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/seemantaggarwal/test.git
git push -u origin master
```



### ...or push an existing repository from the command line

```
git remote add origin https://github.com/seemantaggarwal/test.git
git push -u origin master
```



### ...or import code from another repository

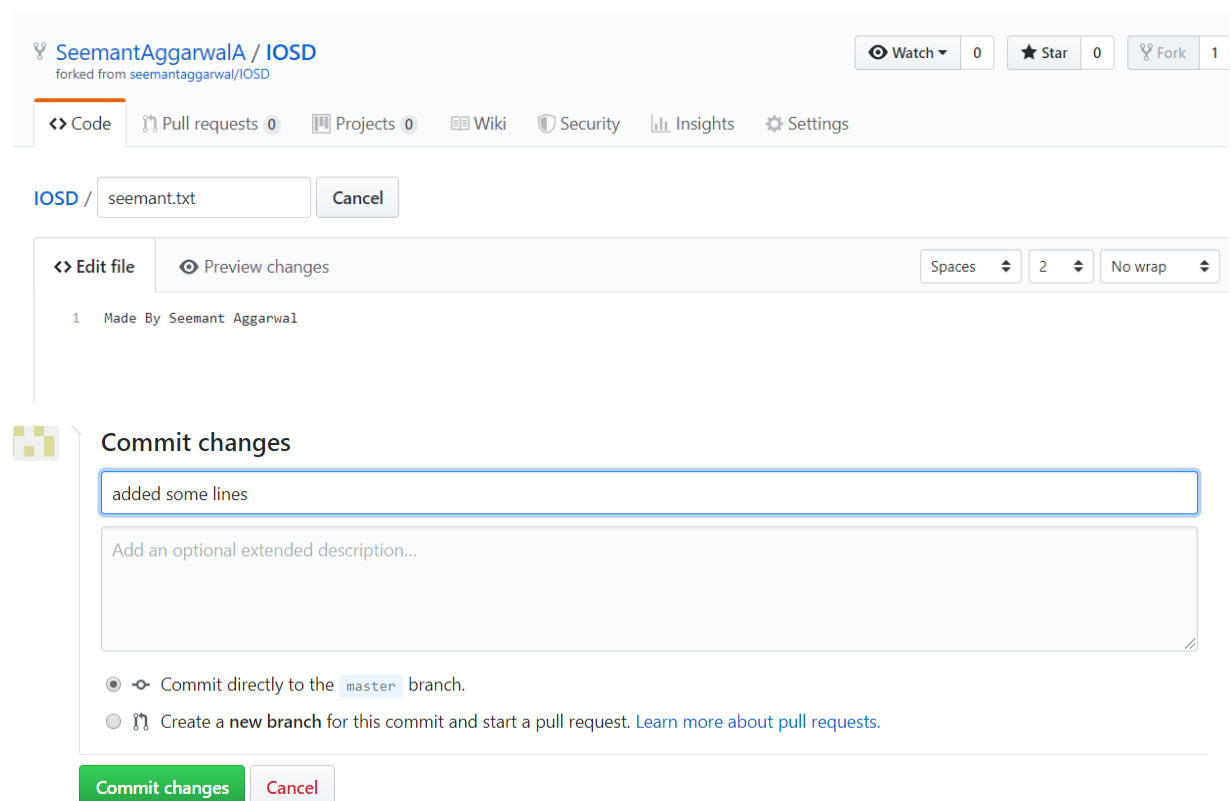
You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Import code

## About pull requests

Pull requests let you tell others about changes you've pushed to a branch in a repository on GitHub. Once a pull request is opened, you can discuss and review the potential changes with collaborators and add follow-up commits before your changes are merged into the base branch.

- 1) Fork the repo
- 2) Make changes
- 3) Commit changes



SeemantAggarwalA / IOSD  
forked from seemantaggarwal/IOSD

Watch 0 Star 0 Fork 1

Code Pull requests 0 Projects 0 Wiki Security Insights Settings

IOSD / seemant.txt Cancel

Edit file Preview changes Spaces 2 No wrap

1 Made By Seemant Aggarwal

**Commit changes**

added some lines

Add an optional extended description...

☒ Commit directly to the master branch.

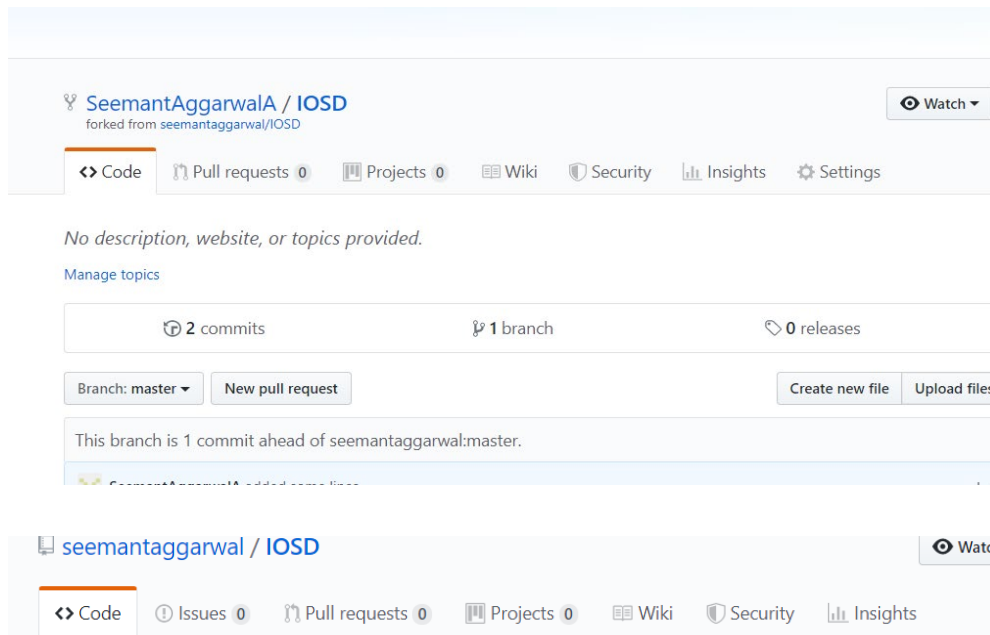
☐ Create a new branch for this commit and start a pull request. [Learn more about pull requests.](#)

Commit changes Cancel

- 4) Create a pull request



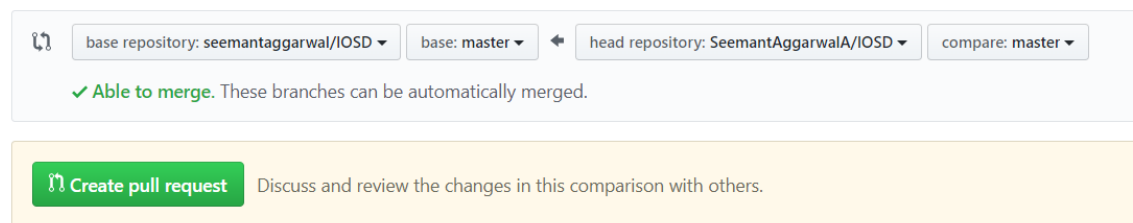
5)  
6)



7)  
8)

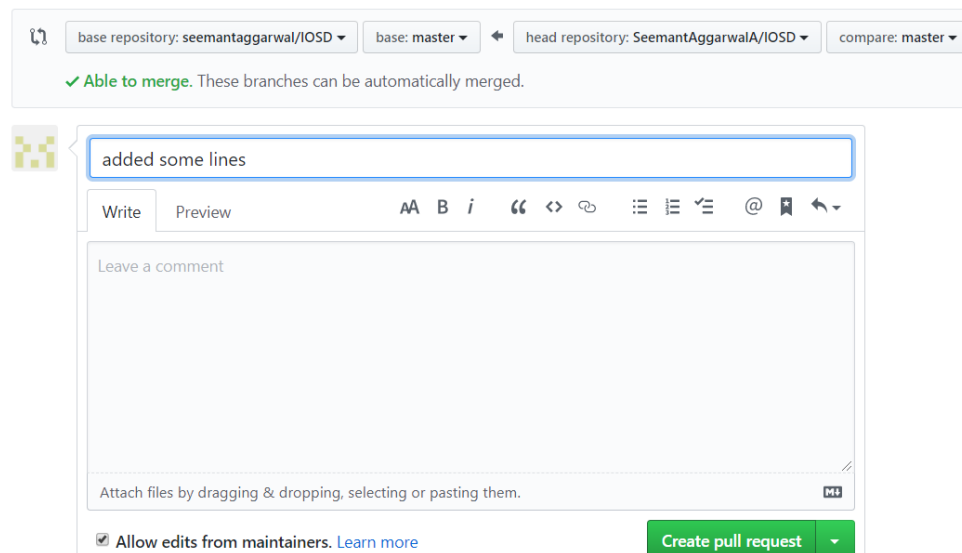
## Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#).



## Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).



9)