# **GIT – Version Control System**

## Registering on GITHUB site

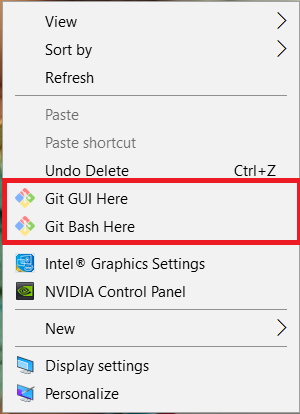
* Open git hub link: <https://github.com>
* Signup by entering your details
* Once Signup is done then you will get a mail from GitHub. Verify your email by clicking on “verify your mail id” link
* Now you can see the  button, click on **New repository** and create your Repository.

## Installation of Git

* Download the Git client from <https://git-scm.com/> and click on downloaded file.
* Click next on all the screens. no need to enter any details.

Verification:

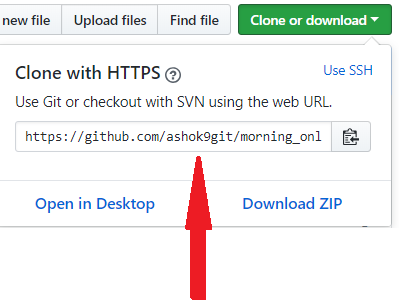
* Right on Desktop, Git Bash and Git GUI should be visible like below.



* Once you are done with Git installation then you can clone your code from github.
* Checkout command:

git clone <repositoryURL>

Note: get the repository URL from GITHUB by copying .. see below pic



* Now you can commit your changes to GitHub repository.

What is GIT?

Git is a **source code management system** and it is **Distributed/Decentralized version control system**.

Developed by **Linus Torvalds** for Linux kernel development.

We have two types of version control systems:

1) Centralized version control system. (CVCS)

2) Distributed/Decentralized version control system. (DVCS)

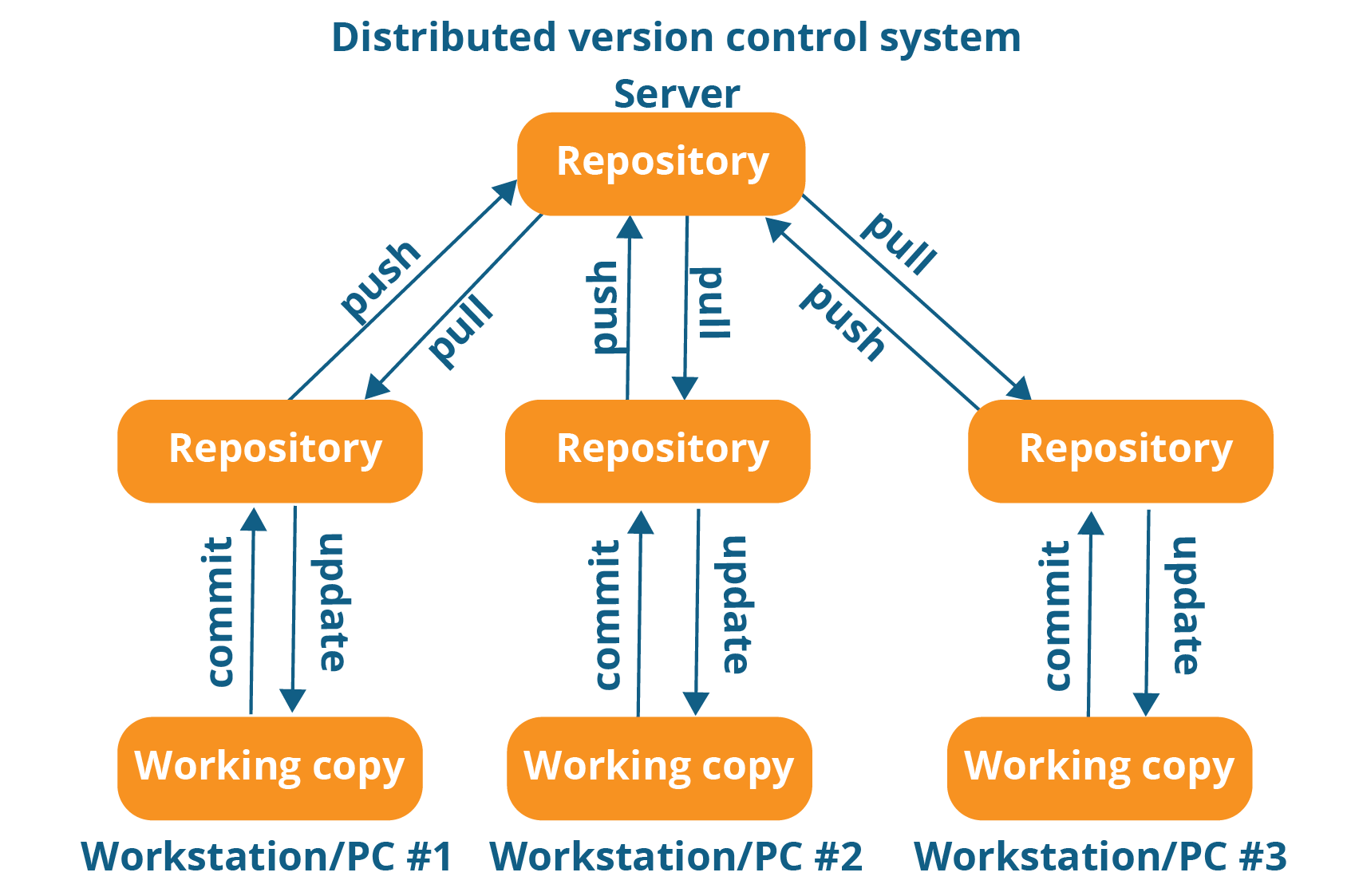
**Centralized version controlling system** holds all the information in the centralized server, so if you server is destroyed/crashed your data related to the repository will be lost. Which states that developers can’t collaborate between them for the code development.

Ex: SVN



**Distributed/ Decentralized version controlling system** as the capability to store the repository in the central server and provides the clients to make a master copy of the repository, so which helps to restore the repository in case of centralized server failure.

Ex: git



**Advantages of using git.**

1) Free and open source

2) Fast and small

3) Implicit backup

4) Security and No need of powerful hardware.

**Commits**

Commit holds the current state of the repository A commit is also named by SHA1 hash. You can consider a commit object as a node of the linked list. Every commit object has a pointer to the parent commit object. From a given commit, you can traverse back by looking at the parent pointer to view the history of the commit. If a commit has multiple parent commits, then that particular commit has been created by merging two branches.

+++

git commit -m "name this commit appropriately to identify the commit with a comment and that you need to show in the repository"

+++

**Branches**

Branches are used to create another line of development. By default, Git has a master branch, which is same as trunk in Subversion. Usually, a branch is created to work on a new feature. Once the feature is completed, it is merged back with the master branch and we delete the branch. Every branch is referenced by HEAD, which points to the latest commit in the branch. Whenever you make a commit, HEAD is updated with the latest commit.

**Tags**  
Tag assigns a meaningful name with a specific version in the repository. Tags are very similar to branches, but the difference is that tags are immutable. It means, tag is a branch, which nobody intends to modify. Once a tag is created for a particular commit, even if you create a new commit, it will not be updated. Usually, developers create tags for product releases.

**HEAD**

HEAD is a pointer, which always points to the latest commit in the branch. Whenever you make a commit, HEAD is updated with the latest commit. The heads of the branches are stored in .git/refs/heads/ directory.

**Revision**

Revision represents the version of the source code. Revisions in Git are represented by commits. These commits are identified by SHA1 secure hashes.

**URL**

URL represents the location of the Git repository. Git URL is stored in config file.

**Basic workflow of git to upload the code to the remote repository:**

----------------------------------------------------------------------------------------------

|  |
| --- |
| **Working Directory** |
| **Staging area/Index** |
| **Local git repository** |
| **Remote repository** |

1) You will modify/edit the files that are created/cloned in the directory.

--> git status (to check the modified files)

2) You add these files to the staging area.

--> git add . or git add \* or git add <filename/directory>

3) You perform the commit to move the files from staging area to the local git repository.

--> git commit -m "comment that you want to add"

4) Atlast you will permanently store the changes to the remote repository.

--> git push -u origin <branch\_name> (ex: git push -u origin master)

→ git push origin <branch\_name>