Git:

Distribution version control system. Designed to work together with team members at the same time.

Created by Linux Torvalds in 2005 to develop the Linux kernel.

# Features:

Open source

Scalability

Distributed.

* Everyone could download a clone copy to their local machine and every user could have their own repository that contains the entire the commit history of the project.

Security – Uses SHA1

Speed.

Supports non-liner development.

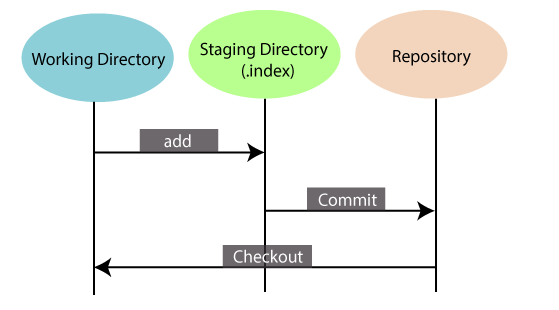
Branching and merging is which makes it different from other tools.

We can create a delete a branch for a new module of the project and commit/delete whenever we want.

Data assurance.

Stating area.

Preview for our next commit.



GitHub:

Git repository hosting service. Provides web-based GUI.

Features:

Collaboration.

Integrated issue and bug tracking.

Graphical representation of the branches.

Git repo hosting.

Project / team management.

Code hosting.

Track and assign tasks.

Conversations.

|  |  |
| --- | --- |
| **Git** | **GitHub** |
| Git is a distributed version control tool that can manage a programmer's source code history. | GitHub is a cloud-based tool developed around the Git tool. |
| A developer installs Git tool locally. | GitHub is an online service to store code and push from the computer running the Git tool. |
| Git focused on version control and code sharing. | GitHub focused on centralized source code hosting. |
| It is a command-line tool. | It is administered through the web. |
| It facilitates with a desktop interface called Git Gui. | It also facilitates with a desktop interface called GitHub Gui. |
| Git does not provide any user management feature. | GitHub has a built-in user management feature. |
| It has minimal tool configuration feature. | It has a market place for tool configuration. |

# Git Environment setup:

Jafer@LAPTOP /d/DevOps (main)

$ git config --global user.name "Jafer"

Jafer@LAPTOP /d/DevOps (main)

$ git config --global user.email "iamfaizaljafer@gmail.com"

Jafer@LAPTOP /d/DevOps (main)

$ git config --global core.editor Vim

Jafer@LAPTOP /d/DevOps (main)

$ git config --list

# Git configuration levels:

**--local**

It is the default level in Git. Git config will write to a local level if no configuration option is given. Local configuration values are stored in **.git/config** directory as a file.

**--global**

The global level configuration is user-specific configuration. User-specific means, it is applied to an individual operating system user. Global configuration values are stored in a user's home directory. **~ /.gitconfig** on UNIX systems and **C:\Users\\.gitconfig** on windows as a file format.

**--system**

The system-level configuration is applied across an entire system. The entire system means all users on an operating system and all repositories. The system-level configuration file stores in a **gitconfig** file off the system directory. **$(prefix)/etc/gitconfig** on UNIX systems and **C:\ProgramData\Git\config** on Windows.

The order of priority of the Git config is local, global, and system, respectively. It means when looking for a configuration value, Git will start at the local level and bubble up to the system level.

# Git terminology:

Branch

Version of the repository that diverges from the main working project.

Git can have more than one branch.

Operations: delete, rename, list, etc,

Checkout

**Git checkout** command is used to switch between branches and repositories.

Cherry-Picking

Apply some commit from one branch into another branch.

Clone

Copy a target repository to our local machine.

Fetch

Used to fetch branches and tags from one or more other repositories, along with the objects necessary to complete their histories.

Head

Representation of the last commit in the current checkout branch.

Index

Staging area between the working directory and the repository.

Master

Default branch.

Merge

Take the data created by a git branch and integrate them into a single branch.

Origin.

Refers to the remote repository from which the project was initially cloned.

Pull/ pull request

Pull is used to receive data from github.

Pull requests are a process for a developer to notify team members that they have completed a feature. Once their branch is ready, the developer files a pull request via their remote server account.

Pull request announces all the team members that they need to review the code and merge it into the master branch.

Push

Upload local repository content to the remote repository.

Rebase

Process of moving or combining a sequence of commits to a new base commit.

Remote

Shared repository that all the team members user to exchange their changes.

Stashing

Sometimes you want to switch the branches, but you are working on an incomplete part of your current project. You don’t want to make a commit of a half-done work. Git stashing allows you to switch branch without committing a current change.

Tag

Tags are used to mark a commit stage as important. Light weighted and Annotated.

Git revert

Revert some commit.

Git reset

Used to reset the changes. Soft, mixed and hard.

Git ignore

Used to specify the intentionally untracked files that git should ignore.

Git Diff

Multi use command. Used to show changes between commits, commit and working tree.

Git cheat sheet

Gives the list of quick summaries of references.

Git flow

Branching model for git.

Git squash

Used to group specific changes before forwarding them to others.

Git rm

Remove

Git fork

Fork is a rough copy of a repository. To resolve a issue for a bug you found you can fork the repository, make the fix and forward a pull request to the project owner.

Git Flow:

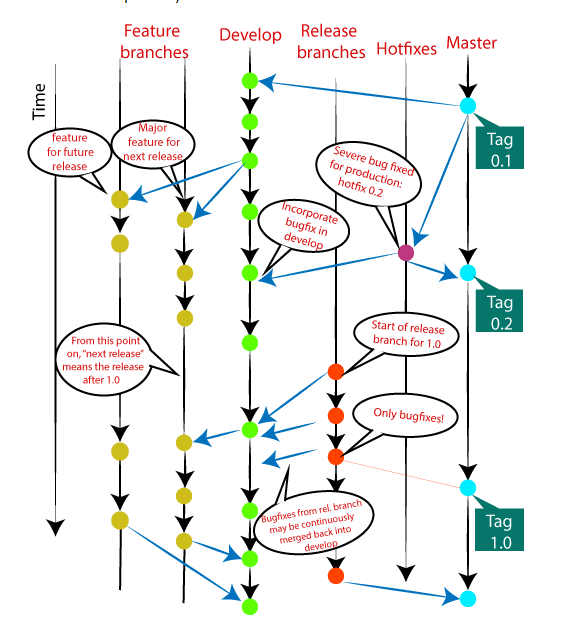
Master branch

Develop branch

Add new features to this branch and once everything is done push it to the master branch.

Supportive branches

* Feature release
* Release
* hotfix



# Release Branches

* Generally, senior developers will create a release branch.
* The release branch will contain the predetermined amount of the feature branch.
* The release branch should be deployed to a staging server for testing.
* Any bugs that need to be improved must be addressed at the release branch.
* The release branch must have to be merged back into developing as well as the master branch.
* After merging, the release branch with the develop branch must be tagged with a version number.

# Git Cheat-sheet:

<https://www.javatpoint.com/git-cheat-sheet>

# git add

git add -A (or) git add . -> all files

adding new or updated files only

git add –ignore-removal

adding all modified and deleted files

git add -u

undo add operation.

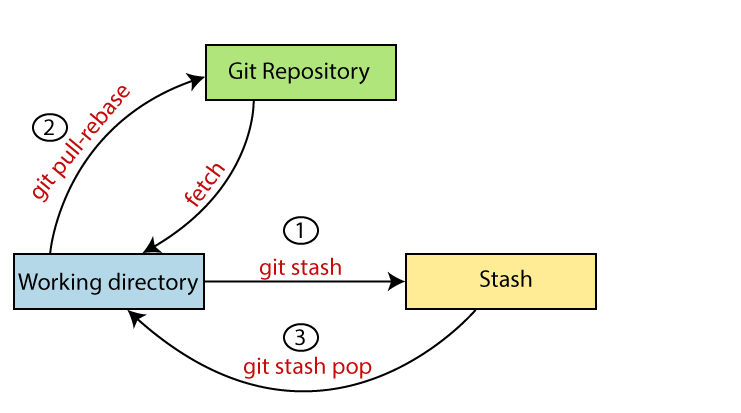
git reset

git commit -amend

git clone URL “directory we want it to go”

git clone -b branch\_name URL

# git stash - https://www.javatpoint.com/git-stash



Git pull command only can be used after adding a repository.