**Git version -- to get the git version installed in your system. ( win/linux)**

**SETUP/ git global configuration**

* **You may also specify options --system, --global, --local to read that value at a particular level.**

**export EDITOR=gedit**

**git config -global user.name <name>**

* Define author name to be used for all commits in current repo. Devs commonly use --global flag to set config options for current users.

To view the above modified

Git config --list

Or

Git config --global --list

Global git configurations are stored in a file called .gitconfig. You can modify the file directly to add the configurations.

**git config --global user.name "[firstname lastname]"**

* set a name that is identifiable for credit when reviewing version history. Define the author name to be used for all commits by the current user

**git config --global user.email '[valid-email]"**

* Define the author email to be used for all commits by the current user . set an email address that will be associated with each history marker

**git config --global color.ui autoset**

* automatic command line coloring for Git for easy reviewing

**git config --global alias. <alias-name> <git-command>**

* Create a shortcut for a Git command. E.g. alias.glog log --graph --oneline will set git glog equivalent to git log --graph --oneline.

**git config --system core.editor <editor>**

* Set text editor used by commands for all users on the machine. <editor> arg should be the command that launches the desired editor (e.g., vi).

**git config --global --edit**

* Open the global configuration file in a text editor for manual editing.

**SETUP & INIT**

**git init < directory >**

* This is to create a local empty repository on git, using git software. Standard is to use remote/online repository like github.com or bitbucket.com

**git clone < repo link >**

* Clone a repository located at <repo link > into a local machine.
* Original repo can be located on the local filesystem or on a remote repository machine via HTTP or SSH.
* Github and bitbucket are examples of remote online repositories.

**git add <file>**

* Stage the <file> for the next commit.

**git commit -m ''<message>"**

* Commit your staged content as a new commit snapshot. Commit the staged snapshot, but instead of launching a text editor, use <message> as the commit message.

git diff

git status commit

Undoing Changes git revert <commit>

git reset <file> overwriting any changes.

git clean -n clean

**Git Branches**

**git branch**

**git branch <branch>**

**git checkout -b <branch> git merge <branch> Remote Repositories**

Show un-staged changes between your index and working directory

List which files are staged, unstaged, and untracked. show modified files in working directory, staged for your next

Create new commit that undoes all of the changes made in <commit>, then apply it to the current branch. Remove <file> from the staging area, but leave the working directory unchanged. This unstages a file without

Shows which files would be removed from working directory. Use the -f flag in place of the -n flag to execute the

List all the branches in your repo. Add a <branch> argument to create a new branch with the name <branch>. List all the branches in your repo. Add a <branch> argument to create a new branch with the name <branch>. Create and check out a new branch named <branch>. Drop the -b flag to checkout an existing branch.

Merge <branch> into the current branch.

**git fetch <remote> <branch>**

* Fetches a specific <branch>, from the repo. Leave off <branch> to fetch all remote refs.

**git pull <remote>**

* Fetch the specified remote's copy of current branch and immediately merge it into the local copy.

**git push <remote> <branch>**

* Push the branch to <remote>, along with necessary commits and objects. Creates named branch in the remote repo if it doesn't exist.

**git log**

git statusList which files are staged, unstaged, and untracked. show modified files in working directory, staged for your next commit

git logDisplay the entire commit history using the default format. For customization see additional options

git log -<limit>Limit number of commits by <limit>. E.g. git log -5 will limit to 5 commits.

git log --onelineCondense each commit to a single line.

git log -pDisplay the full cliff of each commit.

git log --statInclude which files were altered and the relative number of lines that were added or deleted from each of them.

git log --author="<pattern>" Search for commits by a particular author.

git log --grep="<pattern>"Search for commits with a commit message that matches <pattern>.

git log <since>..<until>Show commits that occur between <since> and <until>. Args can be a commit ID, branch name, HEAD, or

any other kind of revision reference

git log -- <file>Only display commits that have the specified file.

git log --graph --decorate--graph flag draws a text based graph of commits on left side of commit msgs. --decorate adds names of branches

or tags of commits shown

**git diff**

git diffdiff of what is changed but not staged

git diff --stageddiff of what is staged but not yet committed

git diff HEADShow difference between working directory and last commit

git cliff --cachedShow difference between staged changes and last commit

git reset

git resetReset staging area to match most recent commit, but leave the working directory unchanged.

git reset --hardReset staging area and working directory to match most recent commit and overwrites all changes in the

working directory

git reset <commit>Move the current branch tip backward to <commit>, reset the staging area to match, but leave the working

directory alone.

git reset --hard <commit>Same as previous, but resets both the staging area & working directory to match. Deletes uncommitted changes,

and all commits after <commit>.

git rebase

git rebase -i <base>Interactively rebase current branch onto <base>. Launches editor to enter commands for how each commit will be

transferred to the new base.

git pull

git pullallow to see what others have contributed to the project

git pull --rebase <remote>Fetch the remote's copy of current branch and rebases it into the local copy. Uses git rebase instead of merge to

integrate the branches

**git push**

git pushpublish to remote repository

git push <remote> --forceForces the git push even if it results in a non-fast-forward merge. Do not use the --force flag unless you're absolutely

sure you know what you're doing

git push <remote> --all

* Push all of your local branches to the specified remote.

git push <remote> --tagsTags aren't automatically pushed when you push a branch or use the --all flag. The --tags flag sends all of your local

tags to the remote repo.

The branching is used also when a repository are made by only one developer, normally one branch correspond to a new functionality or a new update (also called patch) of the software this can help to organize the project. it also provide a better way to test new functionality of the software without touch the master code and if this functionality are no more requested you can simply delete the branch where are stored without too many efforts.

Rewriting Git History

git commit --amendReplace the last commit with the staged changes and last commit combined. Use with nothing staged to edit the

last commit's message.

git rebase <base>Rebase the current branch onto <base>. <base> can be a commit ID, a branch name, a tag, or a relative reference

to HEAD

git reflogShow a log of changes to the local repository's HEAD. Add --relative-date flag to show date info or --all to show all

refs.