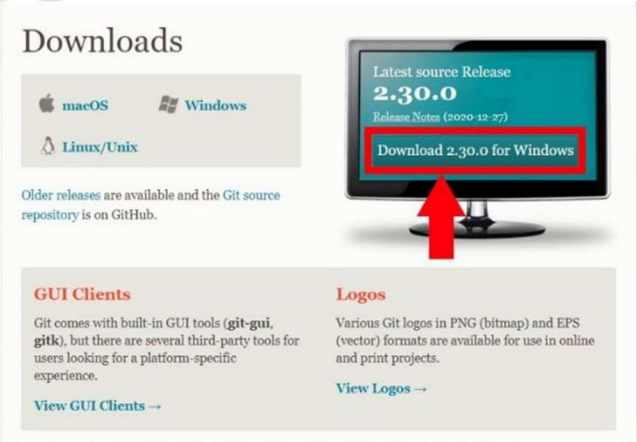
GITHUB CHEATSHEET

The Installation Process

Installing Git On Desktop :

Step 1 : Go to Git's official website: https://git-scm.com



Step 2 : Once you enter the website, click on *Download Git 2.30.0 version* which is the latest version available for Windows package.

Step 3 : After the download is completed, go to your Desktop's File Explorer and enter *Downloads* section where you'll find the latest download named as *Git-2.30.0*. Launch this installer which will begin the setup.

Step 4 : A dialogue box will pop up on the screen asking whether you want to allow this setup make changes to your computer. Click *yes*.

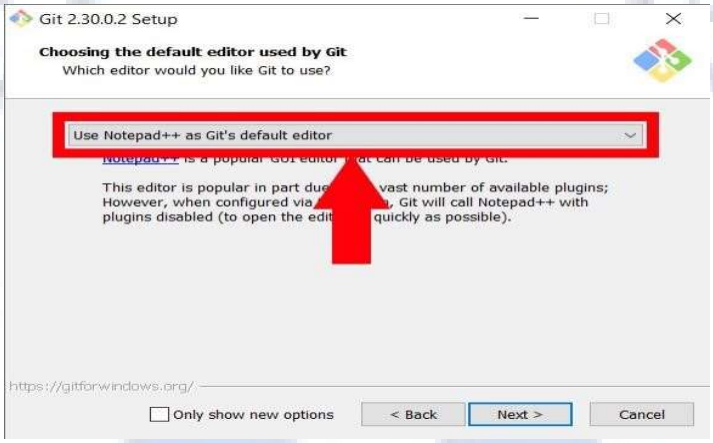
After each of the following steps are completed, click *NEXT*:

Step 5 : Continue the setup and select the default folder where you want the Git application to be installed. By default, the C drive of your computer will be selected and it is recommended to continue with the same.

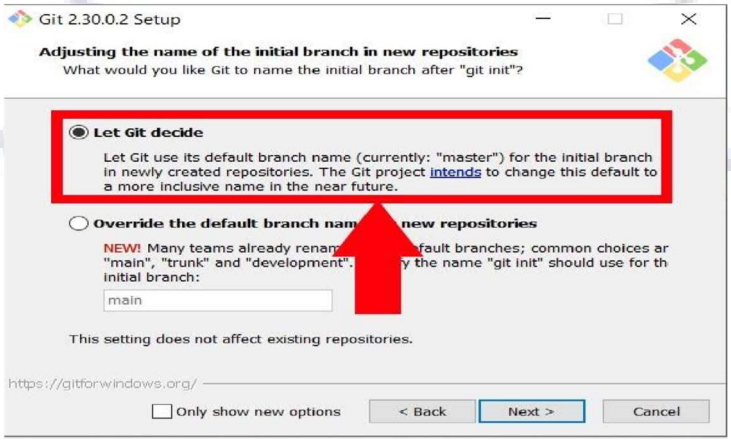
Step 6 : On the next step of *Select Components*, the required components would be automatically selected.

Step 7 : On the next step, you'll be told to select a *Start Menu Folder* which will create the program's shortcuts in it. The *Git* folder would already be selected by default.

Step 8 : On the Choosing Default Editor step, the Git editor would be selected by default but it is recommended to select Notepad/Notepad++ or any other editor which best suits the environment of your computer.



Step 9 : On this step, the setup will ask you to name the initial branch for whenever new repositories are created. Choose *Let Git Decide* option.



Step 10 : Choose the option with *Recommended* written in red while changing the Path environment.

Step 11 : Choose the Default OpenSSL Library option.

Step 12 : Choose the Default Window-style endings option.

Step 13 : Choose the default MinTTY emulator(recommended).

Step 14 : The setup will ask to Choose the default behaviour of git pull. Choose the default option of *fast forward or merge*.

Step 15 : On choosing Credential helper step, choose the default *Git credential Manager core* option.

Step 16 : On the next step, you'll be asked to select the features to enable for configuring extra option. Choose- *Enable File System Caching*.

Step 17 : Click the Install button to install the required components.

Step 18 : Once the installation is completed, click Finish.

Basic Terminologies

Repository : It is a collection of files and directories of different versions of a Project. It is considered as a folder containing the project and is often called a “repo”.

Branch : A branch is a version of the repository that diverges from the main Project. A repository can have multiple branches.

Master : It is the default development branch. After creating a repo, the "master" becomes the active branch. This is the default repository's branch.

Clone : It is used make your own local copy of a remote repository.

Working Tree : The current workspace being used to create / modify the files in the local repo.

Staging Area : The files added to the Staging Area are ready to be committed (saved).

Commit : A snapshot of your code. After making any changes to your local repo, a commit is used to save the changes made.

Remote : The term remote is used to refer to the repository which is stored in GitHub.

Origin : It is the default name of the remote repository from which it was initially cloned.

Push : It refers to the process of uploading changes/commits from local repository to a remote repository.

Fetch : Fetching is downloading code and commit history from remote to local repository.

Merge : Branches created can be merged into another.

Pull : A command by which we ‘fetch’ the changes made to the remote repo and ‘merge’ them with our local repository.

Fork : A Forked repo is a personal copy of an existing remote repository. It allows you to test, debug, or even make changes without affecting the original repository.

Pull Request : Pull requests on an existing repository, is to tell the developers contributing to that repository, about changes you've pushed to a branch in the remote repository.

GIT COMMANDS

1] Setting up your Git Environment :

Using command git config to set up configuration values on a global or local level.

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

git config --global user.email "Your Email-Id"

2] Creating a new Local Repository:

git init -> This command creates a new empty repository and initializes the current working directory as a Git Repository.

Note: It creates a ".git" subdirectory in the current working directory, which contains all the necessary Git metadata for the new repository.

3] Cloning a Repository (Creating a new Local repository from a remote repo):

git clone -> To make a copy of another repository that already exists.

4] Adding files to the Staging Area:

git add *file name* -> To add a file to the staging area (To make git track the file)

git add –p

Note: [-p] flag shows us the change that is about to be added and asks if we want to stage it or not.

5] Moving/Renaming A File In The Repository :

git mv *original file* *new file* -> To rename a file in the repository. (It can also be used for moving files)

Note: The change is “staged” and will be committed in the next commit.

6] Removing A File From The Repository :

git rm *file name* -> To stop the tracking of a file and remove it from the repository.

Note: The change is “staged” and will be committed in the next commit.

7] Commiting The File To The Repository :

git commit *file name* -> To get a file committed to a git directory (from the staging area)
git commit -m ‘Add periods to the end of the sentences.’
git commit -a -m “Call check_reboot from, exit with 1 on error”
git commit –amend

Note: Changes are made to a file in the “Staging Area”.
Note: The command opens an editor where we can enter a commit message.
Note: [-m] flag lets us provide a commit message without the need of having to open an editor.
Note: [-a] flag is a shortcut to stage any changes to “tracked files” and commit them in 1 step. Using this flag skips the staging area.
Note: [--amend] flag takes whatever is currently in the staging area and overwrites the previous commit. It allows us to modify and add changes to the most recent commit.
Note: [--amend] should be used to fix local commits.

8] Checking Status Of The Repository :

git status -> To get information about the current working tree and pending changes. It lists which files are staged or unstaged.

9] For Displaying Commit History :

git log -> It is used to display commit messages.
git log -p (More detailed)
git log -p -2 (Last 2 commits in the log)
git log --stat
git log --oneline (Summarized view of the commit history of a repository)

Note: [-p] flag stands for 'patch' because using this flag gives us the patch that was created. It shows actual lines that were changed in each commit. Press 'q' to exit the view.

Note: [--stat] flag shows some stats about the changes in the commit.

Note: [--oneline] flag is used to see only one line per commit.

10] For checking Changes (Unstaged / Staged files):

git diff -> To view the changes made to a file (shows "unstaged" changes by default)
git diff --staged (show staged changes)

Note: New additions are denoted by green-colored text and a '+' sign at the start of the line.
Note: Any replacements/removal are denoted by red-colored text and a '-' sign at the start of the line.

Note: [--staged] shows us changes that are staged but not yet committed.

11] Checking Branches in Repository :

git branch -> It is used to manage / list all of the branches in the repository. Add a <branch> argument to create a new branch with the name <branch>.

git branch dev (Creating new branch "dev")
git branch -d new-feature (Deleting a branch)
git branch -D new-feature (Forcefully deleting)
git branch -r (Lists remote repos)

Note: The current branch is shown with an asterisk in a different color

Note: The 'git commit' command is used to switch to a branch in the repository.

Note: [-d] flag is used to delete a branch in our repo

Note: [-D] flag is used to delete a branch in our repository even if it has unmerged changes

12] Merging Branches in Repository :

git merge <branch> -> Merges <branch> into the current branch.
git merge --abort

Note: In case of a "Merge Conflict", open the editor and resolve the issues, or abort the merge.

13] Creating A Remote Branch And Pushing To The Repository :

git remote add <name> <url> : Create a new connection to a remote repository. After adding a remote, you can use <name> as a shortcut for <url> in other commands.

git push <remote> <branch> : Pushes the branch to <remote>, along with necessary commits and objects. Creates named branch in the remote repository if it doesn't exist.

14] Pulling changes from a remote repository :

git pull -> When others are working on the same remote repo, and we want to get the updated files from the latest version of the remote repo. We use this command which looks at the original repository that we cloned from, fetches all the changes made since the cloning, and merges the changes into the current repository.

15] Rollbacks (Undoing changes):

git reset HEAD *file name* -> To unstage a Staged file.

git reset *commit id* -> To rollback to a previous commit and start work from there (Use this command wisely as any commits made after the specified commit would not be accessible later).

git checkout *file name* -> If the file is Unstaged, we can undo the changes done to that file and bring back the last commited version in the working tree.

git checkout *branch name* -> To switch HEAD from one branch to another.

git checkout HEAD~2 -> To go back 2 commits from the current position of HEAD.

git revert *commit id* -> This command lets us "undo" the changes made during a previous commit, followed by creating a new commit to document the changes.