Silver plan: Limit 20 repositories already hit

Find way to add more repos

Change subscription. Or see if the new news can help to by pass/

28-06-2012: 25$ plan with 10 repos

30-5-2017: 50$ plan with 20 repos

Option 1:

Make some of the private repos as public. Will lead to more maintenance work later.

Option 2:

**Installation (for Windows)**

Git version 2.26.2 latest -20th April 2020

Git installation installs GUI and command line interface

The command line interface can use Linux commands.

Installation from <https://git-scm.com>

**Documentation**

Documentation at <https://git-scm.com/doc>

Useful book on Git <https://git-scm.com/book/en/v2>

**Git benefits:**

Allows to create many branches

It is fast as it is doesn’t use centralized server as operations are performed locally

It is distributed and every clone has complete information with the history (Perforce, CVS, SVN were centralized systems) so we don’t risk losing all the data. Also, many workflows can be created by collaborating with the non-central users.

Git stores snapshots and not differences like other VCS

It has staging area (index) not present in other versioning systems along with a working directory or working tree (after checkout) and the compressed repository to which commits are done. The three states of a file are committed, modified or staged.

**Configuration files in increasing priority order**

1. Location of gitconfig file for storing username and email for Git activities throughout the system for all users

C:\Program Files\Git\etc\gitconfig

1. Location of .gitconfig file for storing username and email for the user logged in

C:\Users\$USER\.gitconfig

1. Location of git config for the particular repo

.git\config

**Gitignore file**

.gitignore file keeps a record of those files which should not be tracked by Git in the working directory. Useful for binaries, libraries, etc. generated during builds which should not be committed to the repo. Please note that it is possible to have multiple .gitignore files for a project

A list of .gitignore files for various requirements is listed below at  
<https://github.com/github/gitignore>

**Some definitions**

*Author*: one who makes a patch

*Committer*: one who applies a patch

|  |  |
| --- | --- |
| **Configuration** | **Description** |
| git config –-system user.name “Ritesh Udhani”  git config –-global user.name “Ritesh Udhani”  git config –-local user.name “Ritesh Udhani” | Sets username used for all commits.  Updates the file #1, #2 and #3 above. |
| git config –-system user.email “\*\*\*\*@gmail.com”  git config –-global user.email “\*\*\*\*@gmail.com”  git config –-local user.email “\*\*\*\*@gmail.com” | To set email address for all commits.  Updates the file #1, #2 and #3 above. |
| git config --list --show-origin | To see where the configurations are stored. Listed in the order system, global and local with increasing priority |
| git help <command>  git <command> --help | To get help on a command |
| **Commands** | **Description** |
| git init <project name>  git clone <project path> | To create a local repo afresh or from a remote repository. The .git folder is the repository and the files are a part of working directory (working tree). |
| git add <filename> | It means “add precisely this content to the next commit”. This command is used to add a file to start tracking, staging and resolve merge conflicts |
| git restore –-staged <filename>  git reset HEAD <filename> | To unstage a file in the staging area (index) |
| git restore <filename> | To discard modifications done in a file |
| git status | To check the status of each file whether it is modified, staged or non-modified. The output has two regions, staged area and non-staged area |
| git status -s | For a simplified output |
| git diff | To know the exact changes made but not staged. Compares staging area with working tree changes |
| git diff –-staged  git diff --cached | Also, to know the exact changes one is about to commit from the staging area of each file |
| git difftool | To use a different tool for comparison like vimdiff, emerge etc |
| git commit –m “message” | Commit changes to the local repo |
| git commit | The commit message is to be entered through the default text editor for git |
| git commit –a –m “message” | To commit directly from working tree to the repository by skipping the staging area. Use carefully as it can be risky |
| git rm <filename> | To remove and existing file from the repository. It also deletes file from the working tree and get the file deletion to the staging area for commiting. |
| git mv <filename1> <filename2> | To rename a file in git |
| git log | To see the history of commits in reverse chronological order. This lists commit id, author name, email, date of commit and commit messages |
| Git log –-pretty=oneline | Shows commit information (not file information) in one line. Shows a better readable formatting when many commits are present. |
| Git log –-pretty=short | Shows commit information (not file information) in short. Same as git log, but no commit date |
| Git log –-pretty=full | Shows committer and author information both in commit information (not file details) |
| Git log –-pretty=fuller | Committer and author information and the corresponding dates in commit information (not file details) |
| Git log –-pretty=format:”%h %ar” | To get the commit id, author, committer information, date, commit message in a specific format. Shows commit information (not file information) |
| Git log --relative-date | The time relative to now when the commit was made instead of the absolute date (shows commit information and not file information) |
| Git log --stat | In addition to commit information it displays file information like filename, no of insertions and deletions |
| Git log --shortstat | In addition to commit information it displays file information with stats summary line for file changes |
| Git log –-name-only | In addition to commit information it displays file information with only file name in stat |
| Git log –-name-status | In addition to commit information it displays file information with filename and the action done on that file, like additions, modification or deletion. |
| git log --graph | To view branches and merges information |
| git log –p -2 | Filter out commits. In addition to commit information and file information it displays modification information to show the patch details of the last two commits |
| git log –p -1 <commitid> | Filter out commits. In addition to commit information and file information it displays modification information to show patch details of a specific patch with commit id |
| git log –-author=”<part of author name>” | Filter out commits. To see commits by a specific author |
| git log –-committer=”<part of committer name>” | Filter out commits. To see commits by a specific comitter |
| Git log –since=”2 weeks” (“2 years, 2 minutes, 2 hours or specify a specific date "2008-01-15" etc.) | Filter out commits. By time boundation. |
| Git log –until=”2 weeks” (“2 years, 2 minutes, 2 hours or specify a specific date "2008-01-15" etc.) | Filter out commits. By time boundation. |
| Git log –grep “message” | Filter out commits. Greps message in commit messages and filters the commits |
| Git log <filename/directory name> | Filter out commits. Show only those commits which are related to a specific file or directory |
| Git log --no-merges | Filter out commits. Show only those commits which are not merge commits |
| Git commit --amend | To amend previous commit by additionally committing staging area contents. It can also help to modify the commit message. The previous commit is deleted and replaced by a new commit. |

If a file is modified, staged and modified again, the commit will be done only for the changes which have been staged and not the latest ones in the working directory