Links to install the Git

<https://git-scm.com/book/ru/v2>

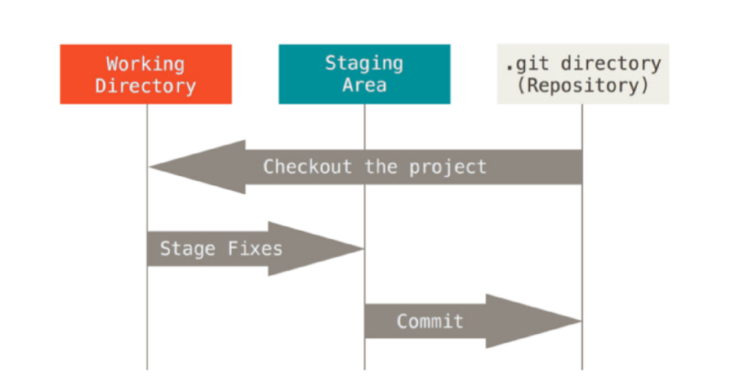
<https://git-scm.com/>

Git – Verson control system, hosting service for git repository

GitHub – is a hosting service for git repository. Git is the tool, while the gitHub is the service to use git.

1. Open terminal git bash
2. Crate folder and open git bash in that folder
3. $ git config –global user.name rajesh
4. $ git config –global user.email [rajeshrewa37@gmail.com](mailto:rajeshrewa37@gmail.com)
5. Git init (Initialized repository) Initialize the git, .git hidden folder is created, **git init** <pathToFolder>
6. git -lart git status

**Recording changes to the git repository**

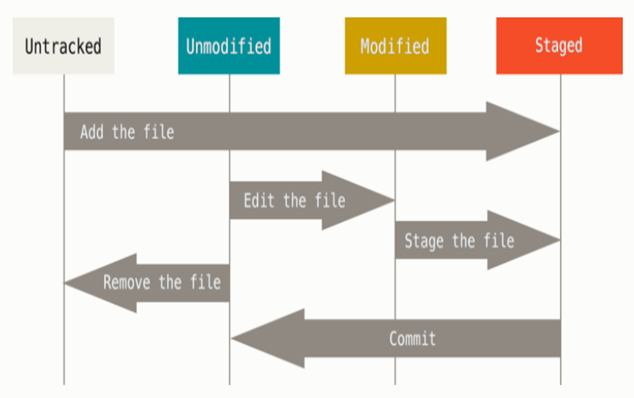
****

* **Crete new file or edit existing file or files in the Working Directory**
* **Stage changes (add to index) staged area**
  + git add . - add all new, changes files
  + git add <pathToFile> - add file
  + git add -- <pathToFile> - add file
  + git add -i - interactive add (can split changes to the hunks)
* **Commit (only added to index changes)**
  + git commit -m “commitMsg”

**You can’t commit changes that are not staged**

**Recording changes to the git repository**

* Each file in your working directory can be in one of two states: tracked or untracked
* Tracked files can be unmodified, modified
* git status - show change’s status
* git diff - show the changes you made relative to the index (staging area for the next commit)
* git diff --cached - to view the changes you staged for the next commit relative to the named <commit>. --staged is synonym for --cached  
  By default <commit> is HEAD   
  (check git diff --cached before committing to see difference)



**.gitignore**

* **.**gitignore file specified intentionally untracked files that git has to ignore
* Add all generated files to the gitignore
* If tracked file was added to the gitignore - git won’t ignore it,   
  If this file could be removed from the server - use   
  git rm --cached <pathToFile> to remove such file from the server and add file to the .gitignore
* **git log**
  + -n - number of commits
  + --pretty - log output format, default is short, to see information about committer, committed date use fuller
  + --graph - display commits graph, useful when we have merge commit
  + -S string - show only those commits that changed the number of occurrences of that string
  + --grep - show commits with a commit message containing the string
  + --since, --after - limit the commits to those made after the specified date
  + --until, --before - limit the commits to those made before the specified date
  + --author - show commits in which the author entry matches
  + --committer - show commits in which the committer entry matches

**Add a file**

Git add index.html created in added in staged area

Git add –A added all file to staged area, tracked all files at once

Git status show the file

Git commit initial commit

Touch about.html create new blank file

Git checkout contact.html recover modified file

Git checkout -f all files recovered once

Git log or git log –p -5 Display last 5 commit out of 100

Git diff Compare working area with staging area

Push added local repository to remote repository

Escape :wq to quit from VM editor

Git rm remove file from working area & staging area

Git rm -- cached remove file from staging area only remain in hard disk, put in untracked file

rm -rf .git // remove git repository

git diff //compare working direcory and staging area

git diff --staged stagging area with previous commit

git rm third.txt remove file

git mv first.txt fi.txt rename the file

if we want to ignore any file just add it in gitignore but when you modify the file it show the modify status of that file why??????

first you have to untraced the file by command

gir rm --cached file name // now file become untraced

git log -p what are changes in commit

git log --stat

git log --pretty=oneline display log in one line

git log --pretty=short display log in one line

**Working with remote**

You can access and write data in repositories on GitHub.com using SSH (Secure Shell Protocol). When you connect via SSH, you authenticate using a private key file on your local machine.

When you generate an SSH key, you can add a passphrase to further secure the key. Whenever you use the key, you must enter the passphrase. If your key has a passphrase and you don't want to enter the passphrase every time you use the key, you can add your key to the SSH agent. The SSH agent manages your SSH keys and remembers your passphrase.

ssh-keygen -t ed25519 -C “[rajeshrewa37@gmail.com.com](mailto:rajeshrewa37@gmail.com.com)”

OR

ssh-keygen -t rsa -b 4096 -C "rajeshrewa37@gmail.com"

When you're prompted to "Enter a file in which to save the key", you can press **Enter** to accept the default file location. Please note that if you created SSH keys previously, ssh-keygen may ask you to rewrite another key, in which case we recommend creating a custom-named SSH key. To do so, type the default file location and replace id\_ssh\_keyname with your custom key name.

**passphrase** - With SSH keys, if someone gains access to your computer, the attacker can gain access to every system that uses that key. To add an extra layer of security, you can add a passphrase to your SSH key. To avoid entering the passphrase every time you connect, you can securely save your passphrase in the SSH agent.

**Generated**

$ ssh-keygen -t ed25519 -C "rajeshrewa37@gmail.com"

Generating public/private ed25519 key pair.

Enter file in which to save the key (/c/Users/RAJESH/.ssh/id\_ed25519):

Enter passphrase (empty for no passphrase):

Enter same passphrase again:

Your identification has been saved in /c/Users/RAJESH/.ssh/id\_ed25519

Your public key has been saved in /c/Users/RAJESH/.ssh/id\_ed25519.pub

The key fingerprint is:

SHA256:SpgOW0IdqkgUbu5BPN1qocirnuLarob4xGanqcYVHXY rajeshrewa37@gmail.com

The key's randomart image is:

+--[ED25519 256]--+

| o. . |

|+ .o.+ E |

| Boo+.o |

|Ooo.o+ |

|+=oo= . S |

|..+B . . |

|+oB o . |

|=O.+ |

|%O\* |

+----[SHA256]-----+

**Ensure the ssh-agent is running. You can use the "Auto-launching the ssh-agent" instructions**

eval "$(ssh-agent -s)"

**Add your SSH private key to the ssh-agent**

ssh-add ~/.ssh/id\_ed25519

**Adding a new SSH key to your account**

Tail ~/.ssh/id\_ed25519.pub

The ssh key is generated add this key

**Now push**

Git push –u origin master

**Branching**

Use a branch to isolate development work without affecting other branches in the repository. Each repository has one default branch, and can have multiple other branches. You can merge a branch into another branch using a pull request.

git branch hello-world-images or git checkout –b newbranch

git branch

hello-world-images

\* master

We can see the new branch with the name "hello-world-images", but the \* beside master specifies that we are currently on that branch.

checkout is the command used to check out a branch. Moving us from the current branch, to the one specified at the end of the command:

git checkout hello-world-images

We have made changes to a file and added a new file in the working directory.

git add –all

Using --all instead of individual filenames will **Stage** all changed (new, modified, and deleted) files.

git commit -m "Added image to Hello World" //Now we have a new branch, that is different from the master branch.

Now, let's see what happens when we change branch to master

git checkout master //The new image is not a part of this branch. List the files in the current directory again:

**Merging**

We have the emergency fix ready, and so let's merge the master and emergency-fix branches.

git checkout master

git merge emergency-fix

git branch --merge // show only the merged branches

git branch –no-merge // show only the non-merged branches

Since the emergency-fix branch came directly from master, and no other changes had been made to master while we were working, Git sees this as a continuation of master. So it can "Fast-forward", just pointing both master and emergency-fix to the same commit.

As master and emergency-fix are essentially the same now, we can delete emergency-fix, as it is no longer needed:

git branch -d emergency-fix //Deleted branch emergency-fix (was dfa79db).

**Conflicts**

Suppose one branches b1 is created with master. Then idex.html in master is updated at some location. Now index.html in b1 also updated at the same place.

When you merge both than it will show a conflict whose updation is important, select it than merge.

