GIT:

- There is a strong History in command lines.
- New features
- Online Help.

Source control:

1. It is a type of Back up it is difficult to store
2. It allows to undo changes.
3. used for comparing

who needs Source control?

Software developers / engineers / programmers
- Source code (Java, C++, objective-c, Ruby, etc)
- Models (UML, ERB)
- SQL, configuration, text files

- Freelancers
- Web designers
  - Graphic artists
    Ex: original art, vector graphics, photoshop files
- Share code / open source

Source control options:

Two main types:

- Centralized.
- Decentralized / Distributed.

Centralized:

- Free: subversion, CVS
- Commercial: clearCase, perforce, Team Foundation server (TFS)
- Requires connection to central server for most operations.

Distributed:

- mecurial (Hg)
- Git.

What is Git?

1. Distributed source control system.
   - Not required to be decentralized.
- opensource.
- Developed for Linux project requirements.
- most operations are local
- very fast
- most popular DVCS, VCS.
- Active community

Key concepts:

- Repository contains files, history, config. managed by Git.
- Three states of Git
  - Working Directory
  - Staging area - pre-commit holding area
  - Commit - Git Repository (history) (.git folder)

- Remote repository (GitHub)

- Master branch.

Branches in Git:

Master

$\Rightarrow$

Time

```
$ git config --global user.name "kumar"
$ git config --global user.email "a.madankumar128@gmail.
                                   Com"
$ git config --global --list.
```

cloning a repository

```
$ git clone url.
```

$ git status.

Creating a text file.

$ echo $ "Test . Git . Quick start demo" >> Start.txt.

$. cat Start . txt.

$ git status.

$ git add start.txt.

$ git status.

$ git . commit - m . "Adding Start text file"

$.git push origin master. → default and only branch in repository
↓ origin refers to the
GitHub copy of our repository

notepad . .bash_profile

---

alias npp='notepad++.exe . -multiInst -nosession'

$ cat . ~/.gitconfig.

$ git config . --global core.editor "notepad ++.exe
-multiInst -nosession"

```
$ git config --global --list

$ git config --global -e
```

Git basics overview:-

- Starting a project
    - Fresh (no source yet)
    - Existing source (locally)
    - Git hub project (Fork & clone)

- Basic workflow (add, comit, push & pull)
- working with Files (rename, move & delete)
- History and Aliases
- Ignoring unwanted files.

```
$ git init fresh-project

$ ls -al.

$ cd .git/.

$ git status
$ make ·hipster. txt·

$ git. add hipster. txt·

$ git status·

$ git commit ⟶ Adding new file·with hipster ipsum
                This was done with Text make 2

$unzip · ~/Downloads/ initializr-verekia-4.0.zip

$. mv initializr · web-project·

$ · ls

$ cd web-project/ .
```

```
$ git init

$ ls -ak

$ git status·

$ git ·add · .

$ git status.

$ git commit -m " my first commit; inline"
```

$ rm -rf .git

Sudo snap install notepad-plus-plus

---

how to add new files

$ git add Start.txt

$ git Commit -m "my first commit".

$ git status

$ git ~~pull~~ origin main. { before push we need to use pull to add changes.

$ git push origin main.

~$ mate ~/.gitconfig → for mac

~$ npp ~/.gitconfig for windows

$ git commit -am "Adding more ipsum text"

Track file: → git is tracking.

git ls-files

Editing files:

```
$ git commit -m " Adding new files"

$ git add hipster.txt
```

Recursive addi:

```
$ mkdir -p level1/level2/level3/level4.

$ cd level1.

$ npp level-file.txt.

cd. level2

$ npp level2-file.txt

$ cd level3

$ npp level3-file.txt -

$ gitt add .

$ git.commit -m " Adding sevelal files recursively.
```

Backing out changes

```
$ git reset HEAD level.txt

$ git checkout -- level.txt
```

Renaming and moving files

```
$ git mv levels-file.txt level3.txt

$ git commit -m "Renaming level3"

$ mv level2-file.txt level2.txt

$ git status

$ git add -A

$ git status

$ git commit -m "rename level2"

$ git mv level2.txt 2.txt

$ git status

$ git mv 2.txt level2.txt
```

$ git mv level.txt level3.

$ git commit -m" moving file from level 2 to level3"

$ mv level2.txt ..
↳ moving to previous directory

$ git add -A → recursively adding

renaming file out side of git

To add individually

$ git add level.txt }
$ git add -u } → updating value

$ git commit -m " file.txt changes" level.txt

Deleting files:

$ npp file.txt

$ git rm file.txt → not possible

$ rm ~~do~~ file.txt → file deleted.

```
$ git rm newfile.txt.

$ git status.
```

2

deletion is permanent

```
$ git com -m "Deleting newfile",

$ git status'

$ git ls-files.
```

→ git is tracking this file

```
$ git rm hipster.txt.

$ git status.

$ git reset HEAD hipster.txt

$ git status

$ git checkout -- hipster.txt.

$ git status'
```

---

deleting files using bash command

```
$ ls.

$ rm hipster.txt

$ ls.
$ git status.

$ git add -A.

$ git status.

$ got commit -m "Deleted hipsleu.txt".


$ rm -rf. level

$ ls.

$ git status.

$ git add -A.

$git status.

$git commit -m "deleting level and all children"

$git Status
```

$ git help.log. ⟶ To get History

$git log

↳ Commit log

```
$ git log --abbrev-commit.

$ git log --oneline --graph --decorate.

$ git log --since = "3 days ago"
```

$\downarrow$

Commit in last 3 days

specific history in individual file

```
$ git log - hipster.txt.

$ git log --follow. -- level0/ level1/ level2.txt

$ git show
```

Git Aliases :-

```
$ git status.

$ git config --global alias.hist ". log - all.
      -- graph -- decorate. --oneline".

$. git hist
```

Ignoring unwanted files and folders.

```
~$ cd ls

$ git status

$ ls -al

$ npp .gitignore

.DS_.store

$ ls .-al.

$ git add . .gitignore

$ git commit -m " Adding gitignore file ".

$ git push origin main.

$ git ecommit -am "Excluding log directory"

$ git pul origin main.

$ git push origin main --) Branch.
                 L) name of remote repository
```

Compare/mergeTool:

- windows
  - p4 merge for windows
  - Git configuration

```
$ p4merge

$ git config. --global mergetool. p4merge

$ git config --global mergetool.p4merge.path
        "c://programFiles/perforce
        /p4merge.exe".

$ git config --global mergetool. prompt false

$ git config --global - diff. tool p4merge

$ git config --global diftool. p4merge.path
        " c:// program Files/perforce/p4merge.
        exe"

$ git config --global diff.tool. prompt=
        false.
```

```
$ npp. README.md.

#| start web project

## introduction.

## purpose.

## How to contribute.

$ git add README.md.

$ git commit -m "Adding README.md"

$ git push origin scrvcl.
```

Comparing working directory and staging area.

```
$ git diff.

$ git difftool.
```

In working directory. tell us the modified files but doesn not content

```
$ git status.
```

$ git diff HEAD

compares diff b/w. working directory and last commit.

$ git difftool HEAD.

Comparing Staging area and git repository:

$ git diff --staged HEAD
↓
last commit

$ git difftool --staged HEAD.

Limiting comparisons to one File:

comparing b/w commits:-

$ git log --oneline.

$ git diff ac6f872 HEAD..

$ git diff HEAD HEAD^

compares Head and Head-1

```
$ git difftool HEAD HEAD^
```
2

```
$ git diff 22c289a b196780
```
&

```
$ git difftool 22c289a b196780
```

Comparing b/w. Local and Remote master branches:

```
$ git diff master origin/master

$ git difftool master origin/master

$.
```

Branching and merging:

```
$ git branch -a
```

↓ represents all master. Both local & remote branches

* master ↓ current active branch

```
$ git branch my new branch

$ git branch -a.
$ git checkout mynewbranch

$ git branch -a

$ git log --oneline --decorate

$ git branch -m mynewbranch newbranch
```

$\downarrow$

move

```
$ git branch -d newbranch
```

$\downarrow$

delete

```
$ git branch -a
```

Happy path : Fast Forward:

```
$ git checkout -b little-change
```

$\downarrow$

before checkout