

Git Hub

SLA :- Service level agreement

1. **srs** : software requirement specification
or known as urs : user requirement specification.
2. Taxation, Invoicing
3. Terms and conditions
4. security ---> followed by security Audit and verification.

Software Ready

Security Audit ---> Task ---> Resolve----> developer (30 days)

salary : 30000/- 10 days => 10000/-

this problem, is not only related to, security audit, but during coding and designing cycle.

Introduction to git:-

1. git unix|linux based tool for scm (source code management)
or vcs(version control system)

version : next type or upgraded type version

here means changes in source code.

git vs github

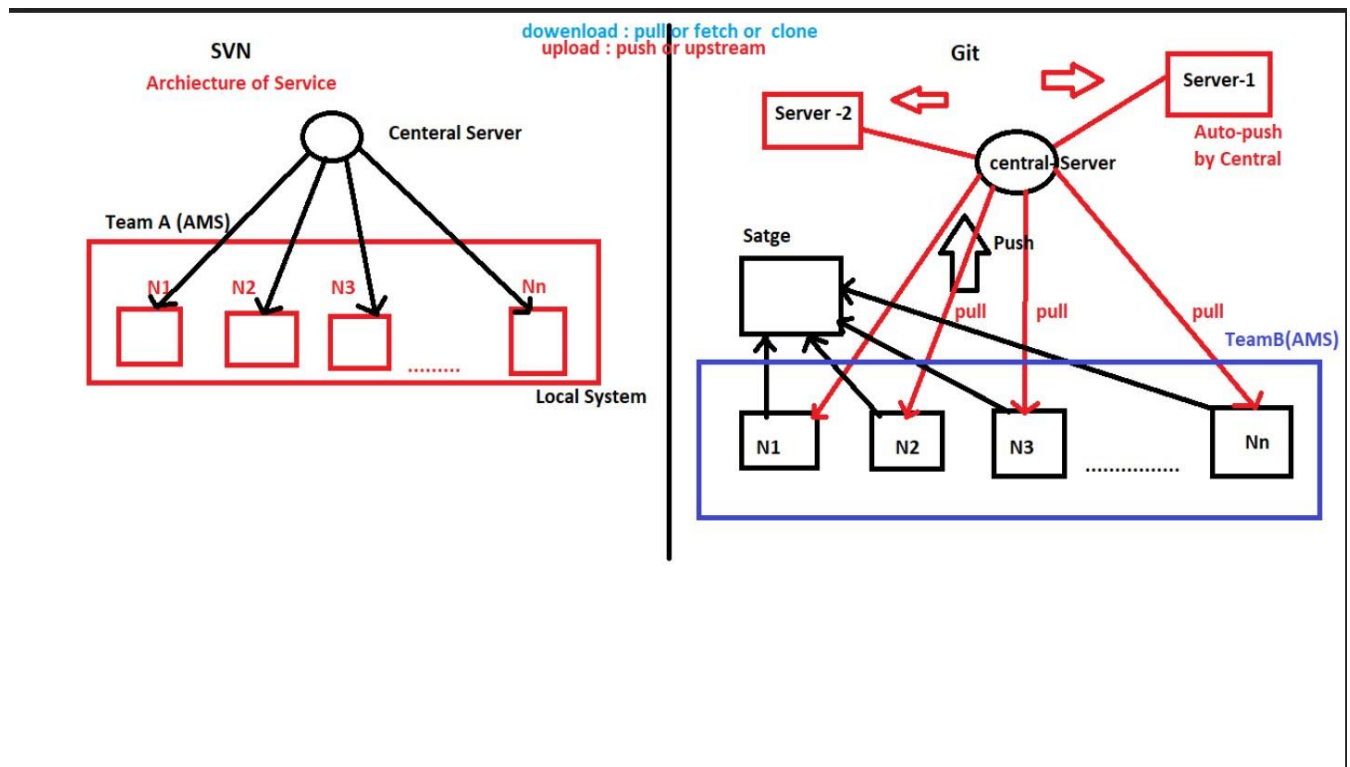
- | | | |
|--------------------------|--|------------------------|
| 1. command line tool | | cloud |
| 2. local system | | server |
| 3. code manage | | code upload, code host |
| 4. Linux Torvald (Linux) | | microsoft (Bill Gates) |

Tools other than git

svn or tortoise (sub versioning) : centralised

git is distributed.

Cmd:- <tool-name> <command>



Tools other than git hub

other than git hub, we have gitlab, bitbucket, bitkeeper

Important Terms:-

download => pull,fetch,clone

upload => push,upstream

Note :: In git, pull,fetch,clone are all used to download but in different cases.

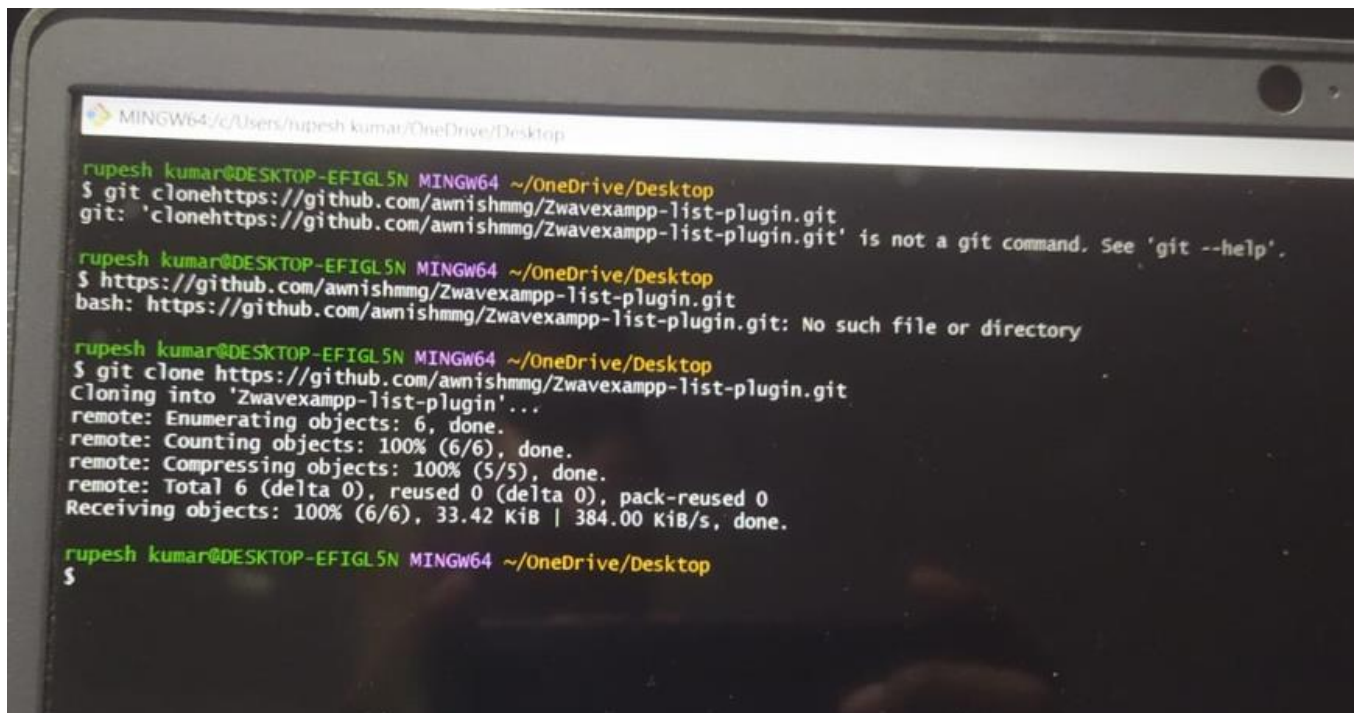
All are same as donwload but at different case

git clone <project>

git pull <project-part>

git fetch -all

[git clone (shift+insert)]

A photograph of a laptop screen displaying a terminal window. The terminal shows a series of commands and their outputs. The first command is 'git clone https://github.com/awnishmmg/Zwavexampp-list-plugin.git', which results in an error: 'git: 'clonehttps://github.com/awnishmmg/Zwavexampp-list-plugin.git' is not a git command. See 'git --help'.'. The second command is 'https://github.com/awnishmmg/Zwavexampp-list-plugin.git', which results in an error: 'bash: https://github.com/awnishmmg/Zwavexampp-list-plugin.git: No such file or directory'. The third command is 'git clone https://github.com/awnishmmg/Zwavexampp-list-plugin.git', which successfully clones the repository. The output shows progress bars for enumerating, counting, and compressing objects, and a final message indicating that the objects have been received successfully.

```
MINGW64/c/Users/rupeesh.kumar/OneDrive/Desktop
rupesh kumar@DESKTOP-EFIGL5N MINGW64 ~/OneDrive/Desktop
$ git clonehttps://github.com/awnishmmg/Zwavexampp-list-plugin.git
git: 'clonehttps://github.com/awnishmmg/Zwavexampp-list-plugin.git' is not a git command. See 'git --help'.

rupesh kumar@DESKTOP-EFIGL5N MINGW64 ~/OneDrive/Desktop
$ https://github.com/awnishmmg/Zwavexampp-list-plugin.git
bash: https://github.com/awnishmmg/Zwavexampp-list-plugin.git: No such file or directory

rupesh kumar@DESKTOP-EFIGL5N MINGW64 ~/OneDrive/Desktop
$ git clone https://github.com/awnishmmg/Zwavexampp-list-plugin.git
Cloning into 'Zwavexampp-list-plugin'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (6/6), 33.42 KiB | 384.00 KiB/s, done.

rupesh kumar@DESKTOP-EFIGL5N MINGW64 ~/OneDrive/Desktop
$
```

git flavours:-

flavours means git is available in other forms :-

1. Cli (command line interface | cmd black and white screen) : comamnds
2. GUI (graphical User Interface) | GUI no coding or command Require

fastest : cli

slow : gui

secure | cli

insecure | gui

beginner | gui

expert | cli

dev|designer|coder |dba|tester |bde => Technical line => CLI

Repository => project folder

31-oct-2022

profile Url or Account Url or workspace url or Profile Url

<https://github.com/username/project-name> : Access public

<https://github.com/username/project-name> : If private ----> owner

-----> Login

Key Points :-

1. *default branch*: master or main changes on August, 2021.
- 2.) If public repo, add valid License if you have if not, Let it be None, or Unlicensed.
- 3.) Public Repository (Repo) can be forked (pull) by anyone (pull).
- 4.) Url, it is visible to Everyone in the Universe.
- 5.) Your Every Action is recorded as Activity and, Added in Watcher list .

Ques:

Repo xyz owner vibhu

Url :- <https://github.com/vibhu/xyz>

Requirement Analysis

Design Analysis

Developement

(Design | coding | Database)

Testing

Deployment (Release)

monitoring

feedback

Hence, from above discussion it is very clear that git provide complete support from Requirement gathering to feedback cycle. where memeber can contribute there, part.

1-nov-2022

How to Add members to the Repository

Members working together in a team for a Repository is called ,
collabarators .

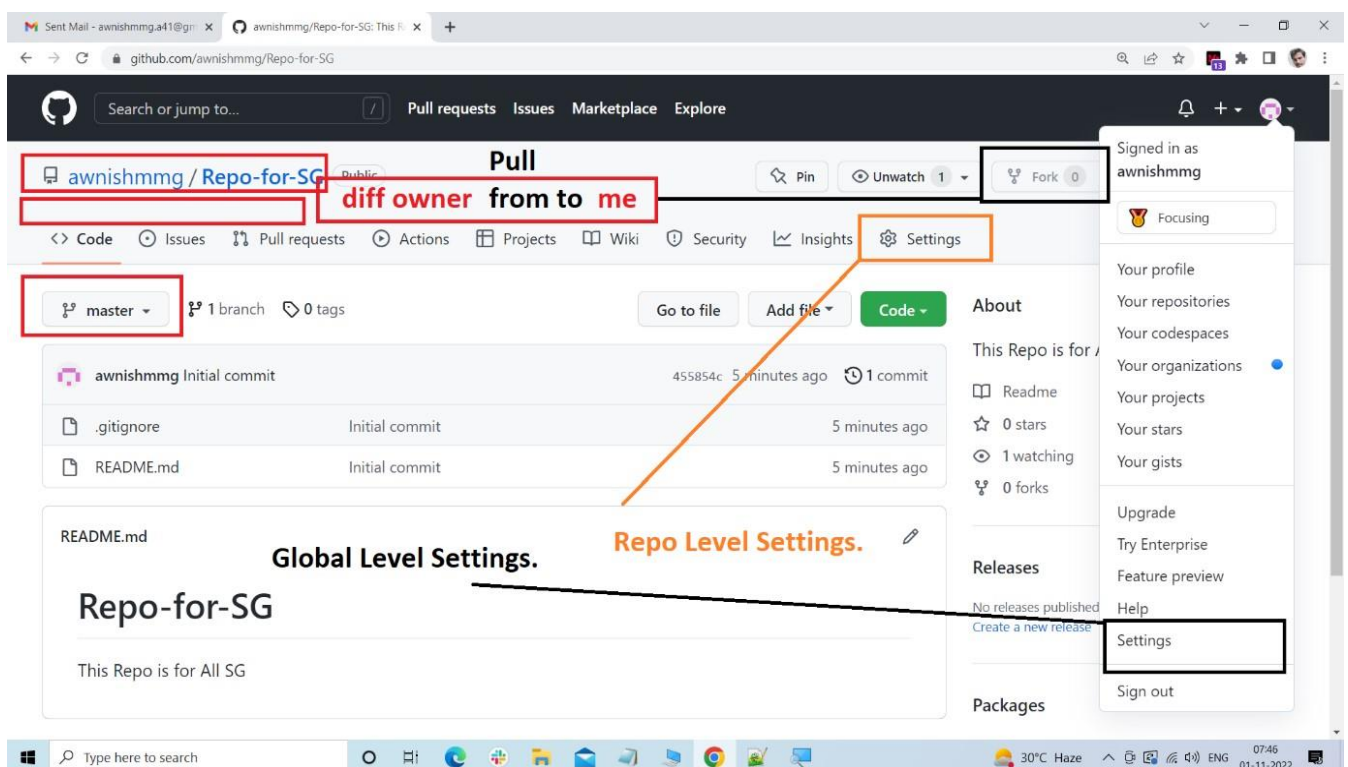
Types of Settings :- Github two types --

.....

1.) Global level setting : - It is locate on profile menu or Account
menu All setting applied here , will be applied on the entire project .

2.) Repo level setting : - Repo setting is located on the right hand
side of the , each private or public repo , everything you create a repo ,
you need to edit these setting .

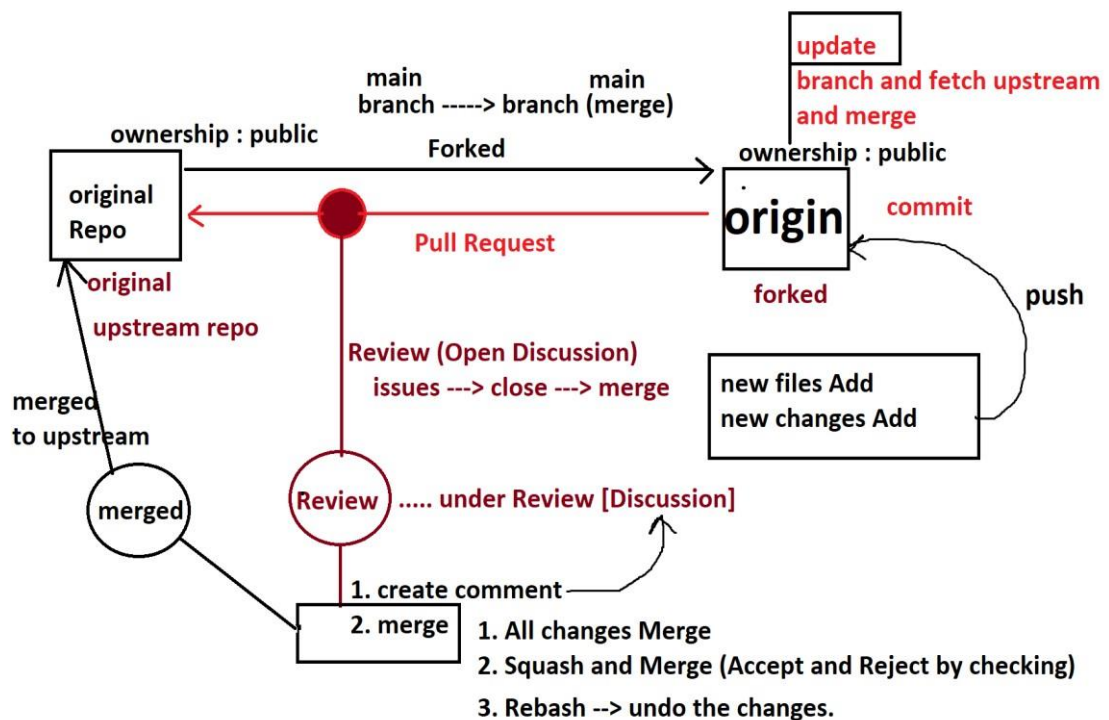
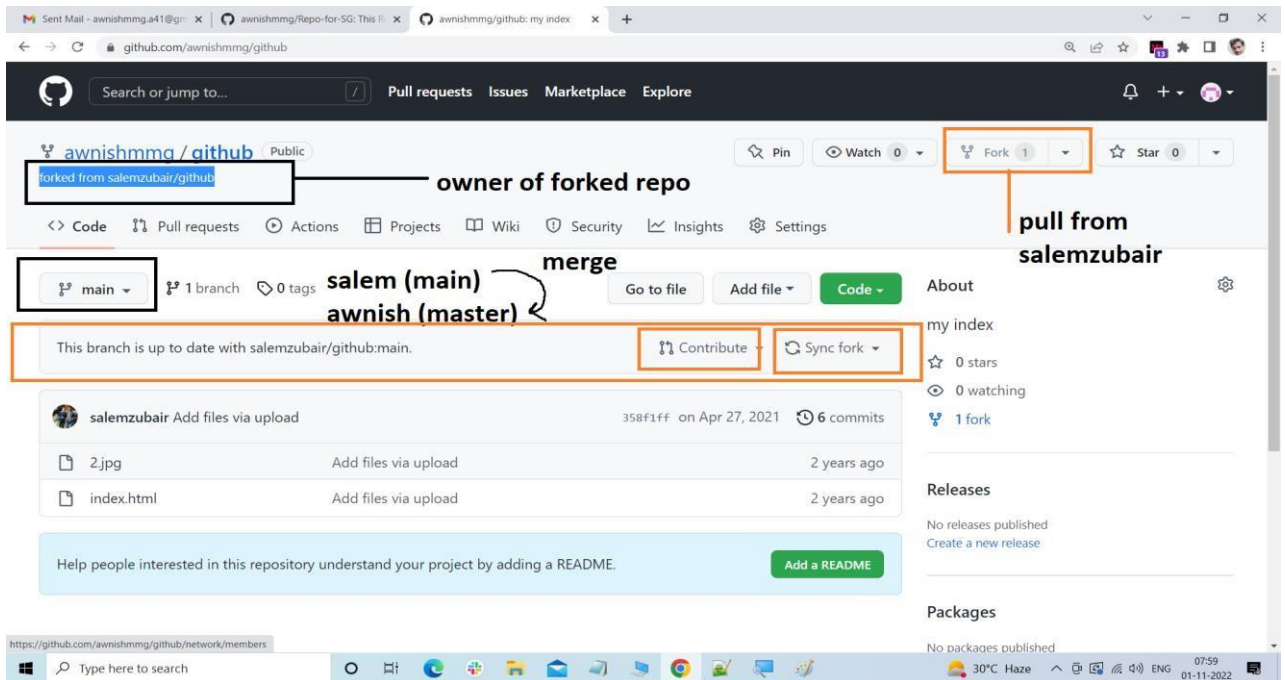
forked –



What is fork How is it important:

Fork create copy project from different to owner to my account it can be applied on public repo.

merging-from-origin –



2-Nov-2022

Linux

working with git in local system

REPL :- Read Evaluate print loop : symbol of the terminal
related to any language

```
>>> print('hi')
```

```
hi
```

```
> print('hi');
```

```
      syntax error
```

```
> console.log('hi')
```

```
cmd:\> echo hello
```

```
hello
```

\$: unix or linux

Important command of Unix or Linux

1. **whoami** : tell the owner name or pc name
2. **clear** : clear the console
3. **echo** : print the output or expression
4. **dir** : used to list of folder and files <cmd B/W>
5. **ls** : used to list down files and folders <unix terminal colorful>
6. **touch** : used to create empty files with or without extension.
7. **mkdir** : used to create folder(directory)
8. **ls -lart**

Note :- ./ or . current Directory

../ or .. parent Directory

9. **cd** : to change the path

Eg : \$ cd <path-name>

Types of File and Folder

1. **Anonymous File** : with Extension but no name
2. **Named File** : with name and extension or with name without extension.

extension .name => extension

without . name => firstname

file fullname => firstname + extension

Types of folder:-

1. *hidden folder* : .foldername
2. *without folder* : readable folder

ls and ls -lart

ls : normal files and folder show.

ls -lart : all hidden files and folder.

ls : no count of files and folder.

ls -lart : total no of files and folder.

ls : files and folders horizontally display.

ls -lart : vertically files and folders.

ls : No files permission and meta data.

ls -lart : shows all permission, of read, write, execute, security, sharing. author, when created, who created, data and time information (meta-data).

rm will delete file

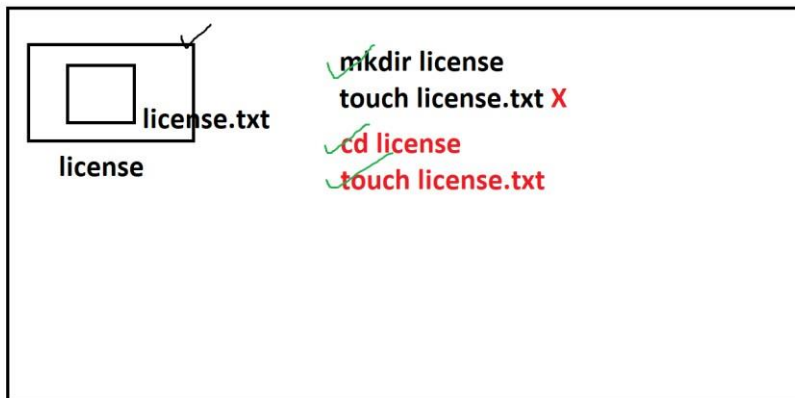
rm -r <folder-name> : delete folder (shift + delete)

Task 1:-

`mkdir xyz`

`| --cd--> abc.txt (touch)`

&& : multiple statement Run at a single line



How to write data inside a file

echo any statement you want to write `> abc.txt`

How to write data inside a file

`cat <filename>`

eg:

`cat abc.txt`

content at

Task Write the Two lines inside abc.txt file

1. echo html is front end language > abc.txt
2. echo python is backend language > abc.txt

Shell file write two mode

1. override mode >
2. append mode >>

How to set Range in command

\$ command anyname{start..end}

eg:

make 300 folders

mkdir test{1..300}

delete 300 folder

rm -r test{1..300}

pwd :- print working directory

display current full path

Before we finish with git, we must be know basic command of linux

| unix.

that we already know.

Basic Terminology in Git

1. *Versioning or version :-*

maintianing the different changes of code is called as versioning or version.

2. *Work-space / working Directory*

That main folder where we are working , and all our important files and folder are located .

D:

|-----> summerTraning

|-----> winterTRaning

|-----> Apprenticship

|

|-----

1.html

2.css/js/php/Android/

3.projects (work-space/working)

1.calculator

2.static-web sites

3. crud.project

1. Repository:

Type 1. *public Repository* : Everyone access

2. *private Repo* : No body can access only access can be done who has the access.

Access : owership (master | main)

Team :

forked : contributor -----> for contributor -----> fork life cycle.

we know that github cloud, centralized server connected data share.

Local System

local system

1. Private repo.

Scope Transfer or Visibility Transfer :-

local Repo visibility : private

private : share own -----> private in github

private : share Team -----> private in github

private : share with Everyone -----> public in github

Different phases of Files/Floder in the Git

1. untracked |
2. tracked |
3. stage |
4. commit |
5. origin |
6. upstream |

Towards Downwards

we have three repository

i.) local system

|
push
|

ii.) your github repo

|
fork/pr (pull req.)
|

iii.) others github repo

|
pull or clone or fetch
|
|

iv.) others local system

Repository :-

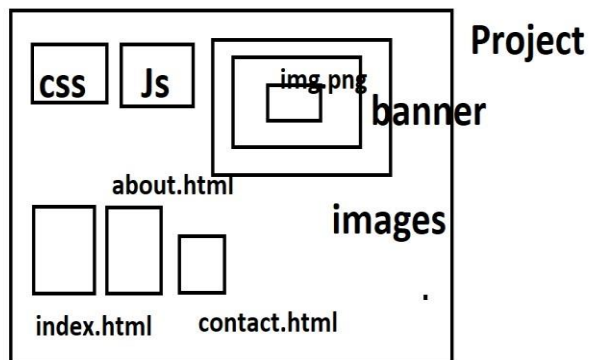
Sauhar(Pati) -----> Abbu (papa)

min 1 child

Parent : workspace / working directory

Workspace –

workspace :: Parent Folder with all important files and folders



+-----+

| Folder 1 |

| Folder 2 | -----> Repository

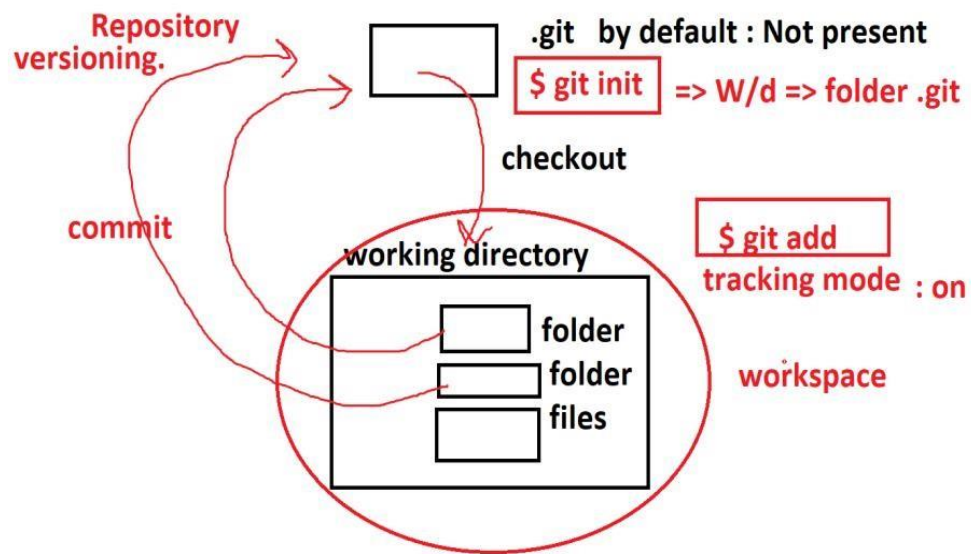
| Folder 3 | special folder (.git)

| Folder 4 | atleast/minimun one commit (initial commit).

| Folder-n |

+-----+

Commit –



4. **commit** :- means save ,add the file to the .git repository
5. **checkout** :- taking back the files from .git to the original workspace or working directory is called as , checkout.

4-Nov-2022

Git Architecture :-

If you look at SVN architecture, then every commit and checkout.

will take from central repository or server(cloud).

Note :: In svn not data is locally saved.

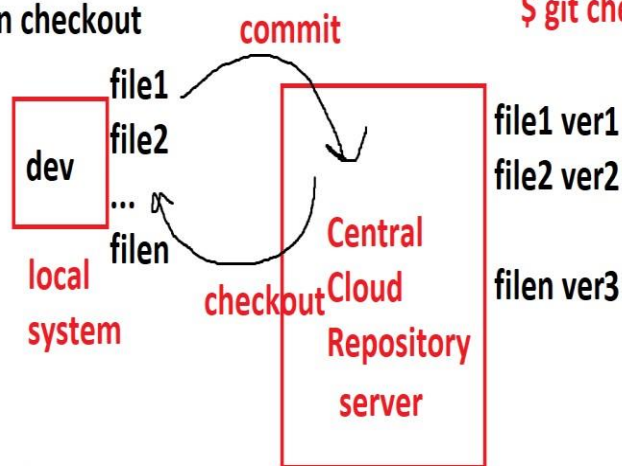
but in Git data is saved locally as well as in, cloud.

svn-vs-git-cc-cycle –

SVN commit and checkout

✓ \$ svn commit

\$ svn checkout



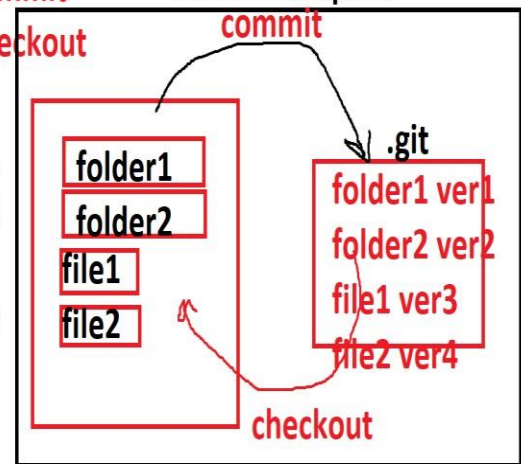
c/c : cycle or commit checkout cycle
local to server : scope
centralise.

this is made within

same workspace

\$ git commit

\$ git checkout



c/c: cycle or commit checkout
local to local : scope
distributed

important command in git:-

\$ git status : status check for file .

How to initialise .git repo

\$ ls -lart .git folder not available

\$ git init

\$ ls -lart .git/

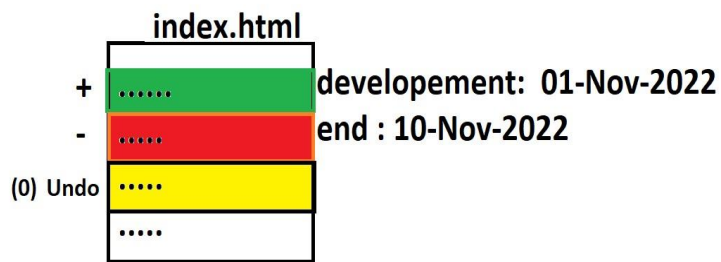
How to add files to stage (add to cache)

\$ git add index.html

Versioning –

Role : web-developement

SRS :- 01 to 08 Nov



Code Versioning : we need to generate PR(Pull Request) -----> code Review ()

Date	Red	Green	
01-Nov		line1	
02-Nov	line 1	line2	line1
03-Nov	line1	line2	
04-Nov		line3	
...		line4	
....			
10-Nov		line10	

Code Versioning : All Changes of code

How to add files to un-stage file (remove from cache)

\$ git rm --cached index.html

How to commit the changes in the file

Note :: before commit your git should know you information or
Identity

This identity is saved in config

How to get author information to config:-

Note :: One system has one owner

local config
global config

single user : global

multiple user : local

How to see local and global config

\$ git config --local

\$ git config --global

Author Information:-

git store author two details :

1. user.name:

2. user.email:

local Author data see

\$ git config --local user.name

\$ git config --global user.name

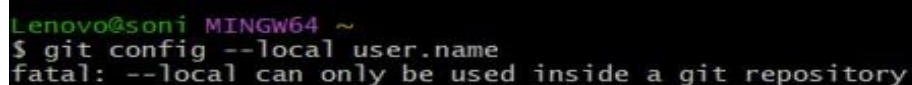
\$ git config --local user.email

\$ git config --global user.email

How to set author information to config:-

\$ git config --local user.name "soni nishad"

\$ git config --local user.email "soninishad7860@gmail.com"

A terminal window with a black background and white text. The prompt is 'Lenovo@soni MINGW64 ~'. The command entered is '\$ git config --local user.name'. The output is 'fatal: --local can only be used inside a git repository'.

\$ git config --global user.name "soni nishad"

\$ git config --global user.email "soninishad7860@gmail.com"

To see :-

\$ git config --global user.name

\$ git config --global user.email

```
Lenovo@soni MINGW64 ~
$ git config --global user.name Soni

Lenovo@soni MINGW64 ~
$ git config --global user.name
Soni

Lenovo@soni MINGW64 ~
$ git config --global user.email "soninishad7860@gmail.com"

Lenovo@soni MINGW64 ~
$ git config --global user.email
soninishad7860@gmail.com

Lenovo@soni MINGW64 ~
$
```

How to commit the changes:-

\$ git commit -m "I have added index.html and about.html and
some code"

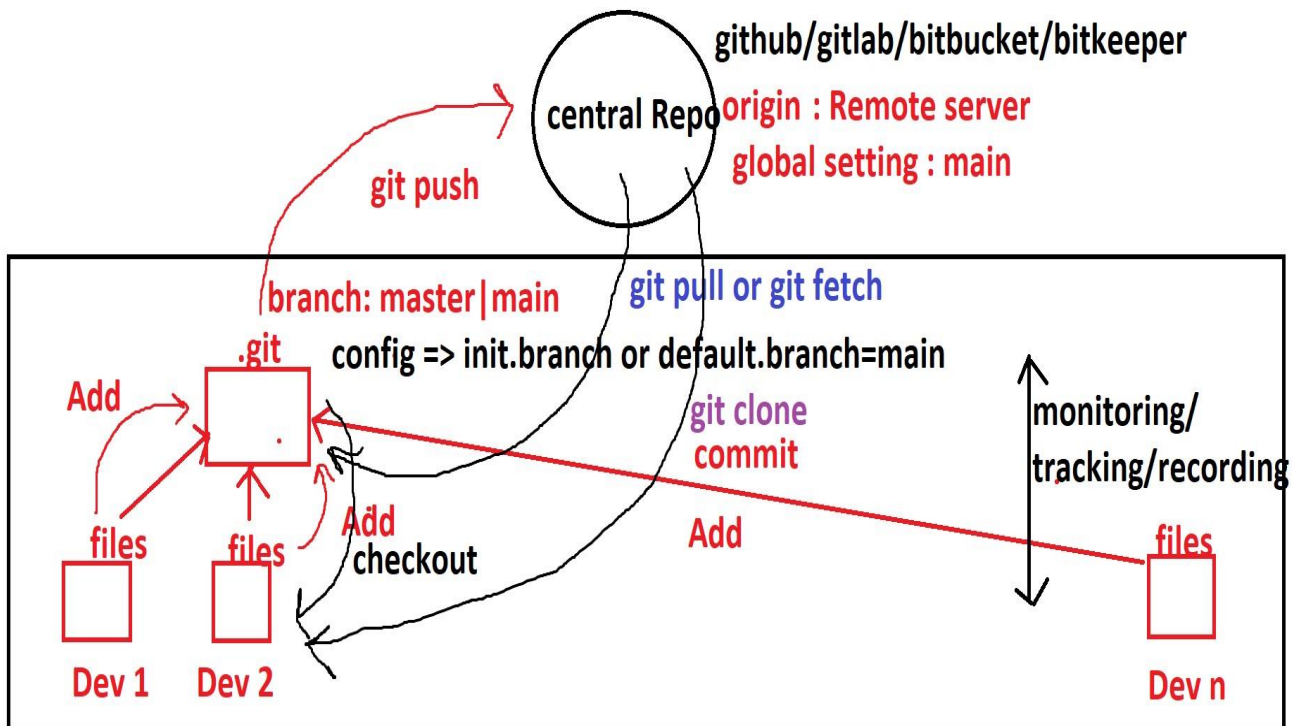
```
Lenovo@soni MINGW64 ~
$ git commit
fatal: not a git repository (or any of the parent directories): .git

Lenovo@soni MINGW64 ~
$ git commit -m "added "
fatal: not a git repository (or any of the parent directories): .git

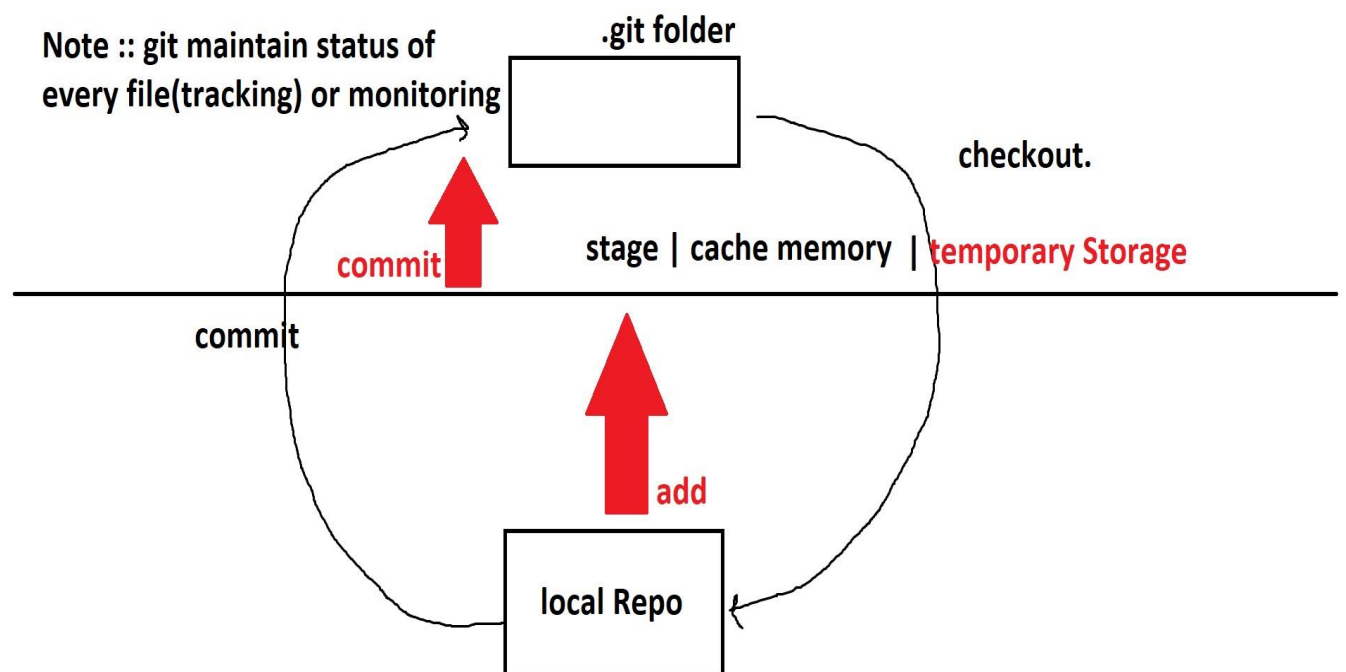
Lenovo@soni MINGW64 ~
$ |
```

Note :: mode will changed : and All staged file ---> commit
files will again untracked mode and will unstaged

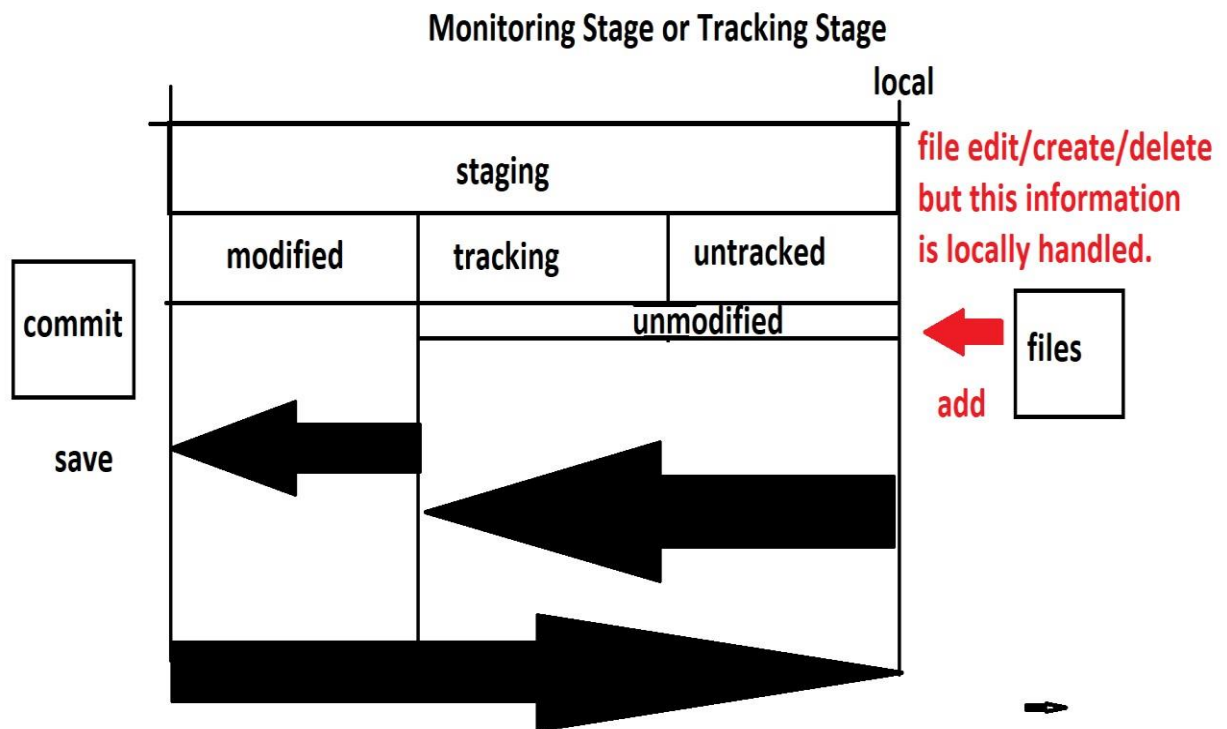
git-Architecture-1 –



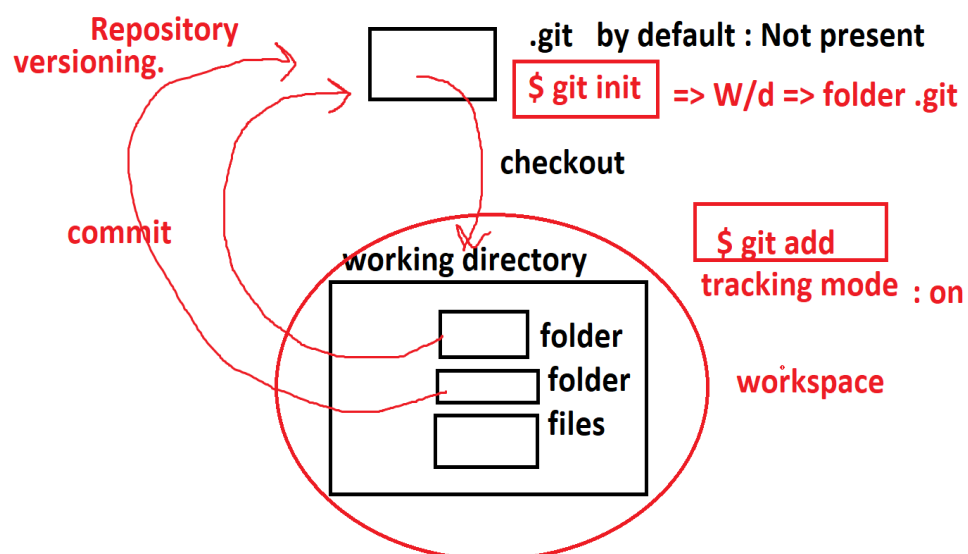
git-Architecture-2 –



Architecture-3 –



Commit –



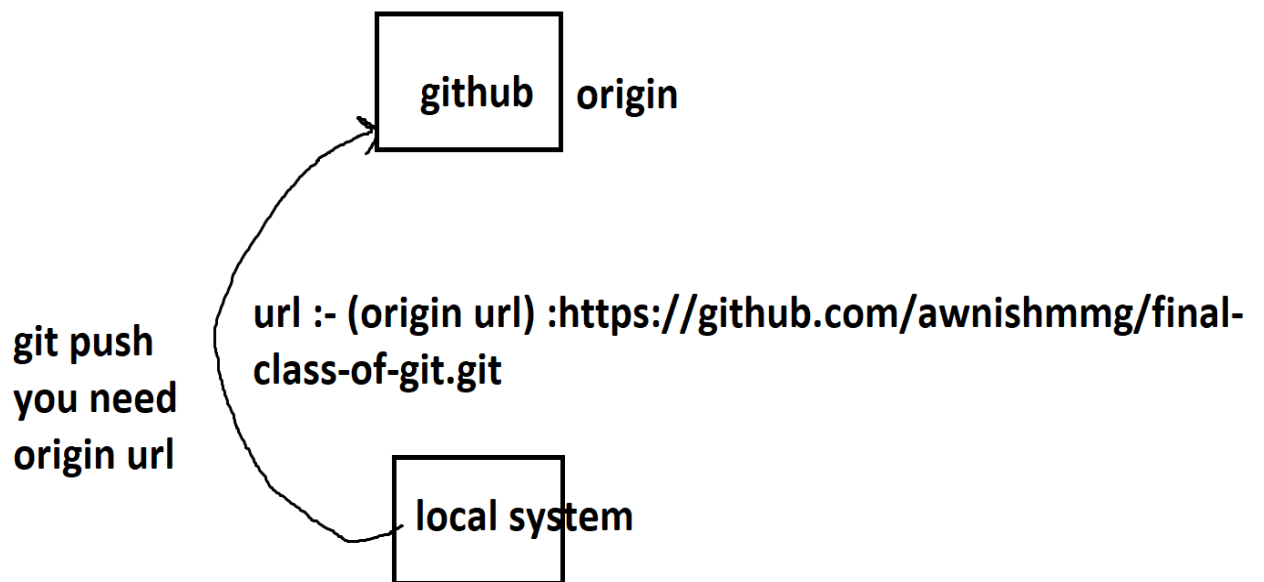
How to checkout:-

\$ git checkout <file-name>

\$ git checkout .

git branches :-

<https://learngitbranching.js.org/>



Q1. Which of the following is default branch of github

- 1.master (Correct)
- 2.test
- 3.awnish
- 4.production
- 5.None of the Above.

Q2. Which of the following is default branch of github

- 1.master
- 2.main (Correct)
- 3.awnish
- 4.production
- 5.None of the Above.

One Important Tools very very Popular (TCS,wipro,....infosis,...)

PMS : is used follow the Organisation workflow in SDLC.

JIRA paid software (PMS | 1800 Software)

OpenProject open source (PMS | 1 Software)

Software => SDLC (Software Developement Life Cycle) => Organisation

S/W Company

1. service based.
2. product based.

Brief Introduction About Agile and SDLC

2 weeks => 14 Days => sprint : (time period of project in 1 Sprint)

10 Days => scrum

1 week => 5 days scrum call.

Process of Sprint

1. Sprint Planning or grooming
2. 1-Week-Scrum-Review Meeting on Daily Basis or weekly Basis (most of the cases)

1-day Daily Stand Up Scrum (planning| Internal-Demo| Retro) | DSR(Daily Status Report)

2-day Daily Stand Up Scrum

3-day Daily Stand Up Scrum

4-day Daily Stand Up Scrum

5-day Daily Stand Up Scrum Review Meeting fix

3. Monday-Demo Internal or Restrospection (Retro meeting)
 1. stop doing
 2. keep doing
 3. start doing
4. Friday Release or Client Demo

For More Information Visit :-

<https://hygger.io/blog/what-is-scrum-lifecycle/>

Converting Scrum to Git:-

Scrum 1

Daily Stand Up

\$ git clone or git pull 07-Nov-2022

DSR or Review Daily

\$ git commit git push : code Review

merge master

Scrum 1

Connect with JIRa

Task No : TN -1 Kaif

Project XYZ

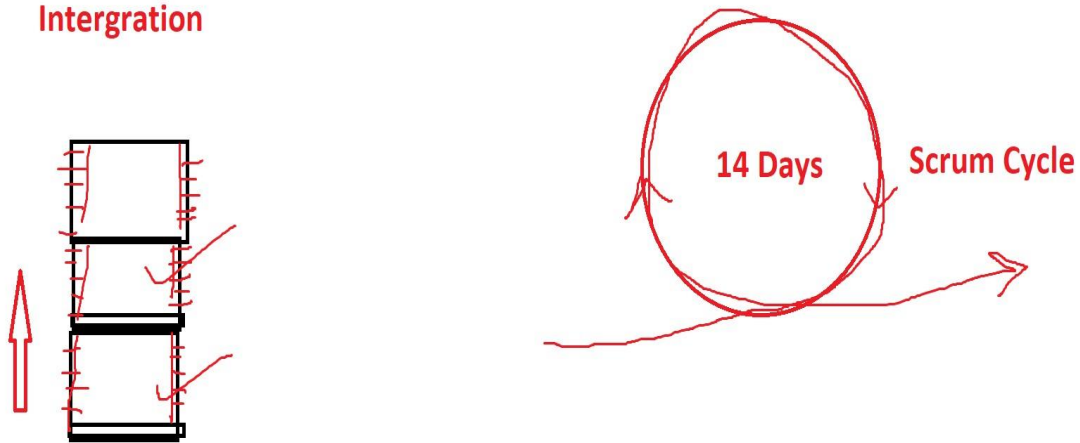
1. TN-0 master : License
2. TN-1(Ekarsh) : Create Index.html
3. TN-2(Akash): Create About.html
- 4 TN-3(Sheshank): Create Contact.html
- 5 TN-4(Satyam): Created Login.html
- 6 TN-5(vijayshekhar): Created Dashboard.html
7. TN-6 (Harsh) : Create Test Report

Agile-scrum-cycle –

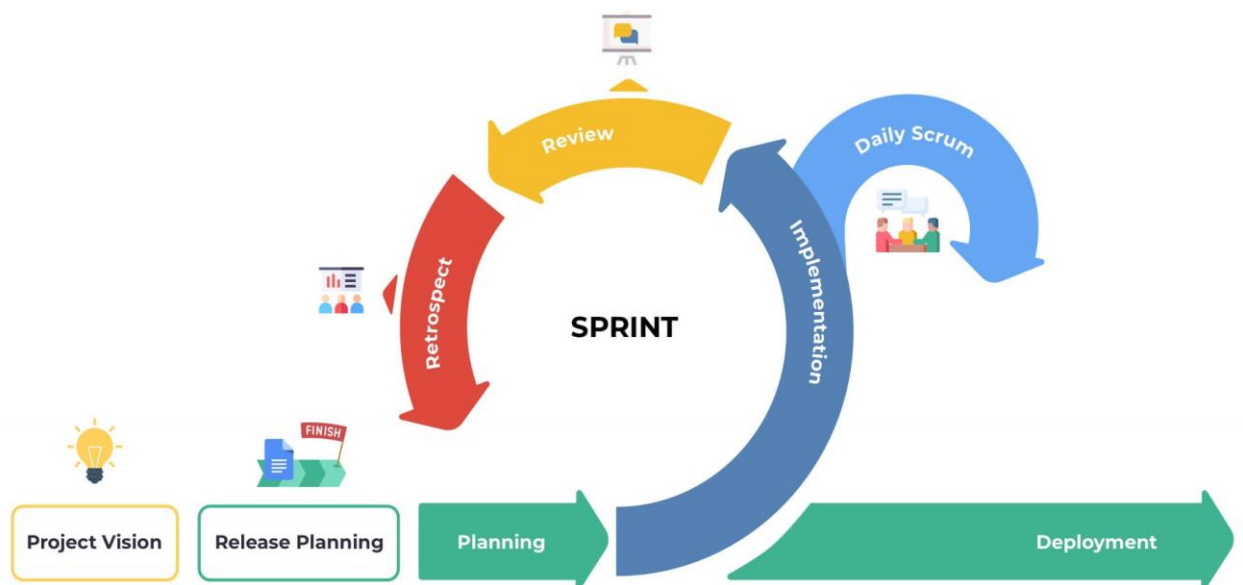
Agile model : sprint basis => min 14 Days.

Agile : fast mover

Agile : Continous developement/Deployment and continous Intergration

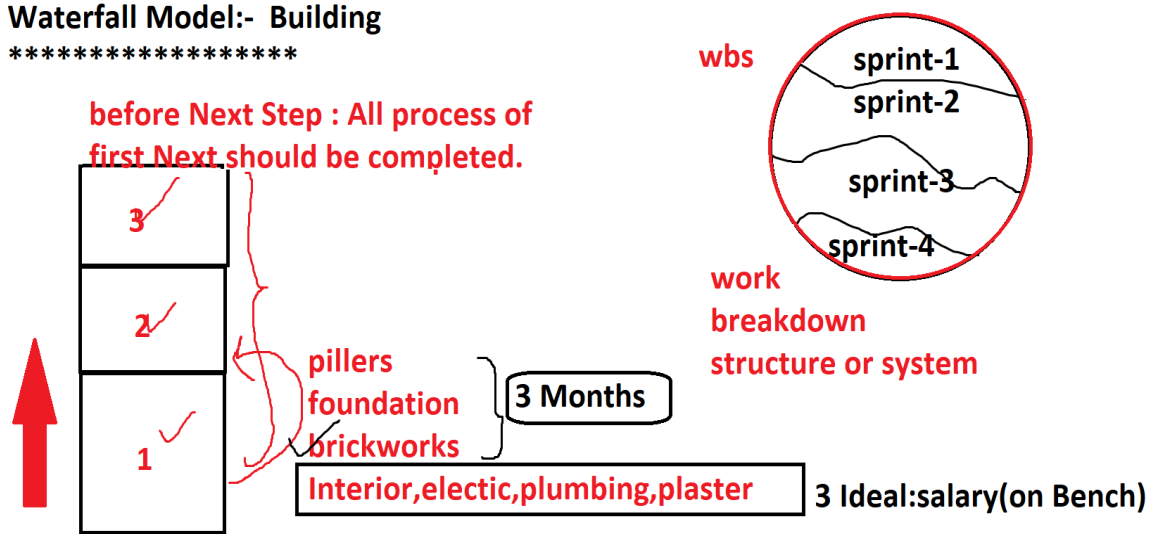


scrum-diagram –



Water-fall-model –

Waterfall Model:- Building



Jira-tool –

start.atlassian.com

Mohd Kaif

Avatar



switch To

Notification
↓
settings

Frequent

Recents

git : username and useremail(Author) : Jira Ticket Assigned Name.

commit : Date and time

Date and time ----> scrum date

Jira 8 Hours

pull Request and Push

How to see branch:-

git branch

*master | current branch

How to get Commit Information:-

\$ git log

How to create a new branch:-

\$ git branch <any-new-branch>

How to change the branch:-

\$ git checkout <branch-name>

Note :: Merging can take place only by two ways

1.checkout from master and merge from master (easy) to any branch

2. checkout with latest commit on the child branch detach the head and merge with child branch (difficult and confusing).

Note :: Never forget to finally commit.

```

Lenovo@soni MINGW64 ~/Desktop
$ mkdir gitpractical

Lenovo@soni MINGW64 ~/Desktop
$ ls
'Adobe Dreamweaver 2021.lnk'*  boot_demo2/
'All notes'/'                  colorpicker.exe*
Apprenticeship/               dBMS_NOTES.docx
Birthday/                     dbms/
'CLG DOCUMENT'/'              demo/
Connect_4/                    demo-project/
DxDiag.txt                    demo_boot/
'Git Hub.docx'                desktop.ini
'Oracle software'/'          files-in-js/
Postman.lnk*                  git/
Practice/                    git-demo/
'Python 3.10 (64-bit).lnk'*   gitpractical/
'SD TASKS'/'                  'javascript Appre'/'
Scholarship/                  'rapid fire Ques'/'
'Visual Studio Code.lnk'*     sadia/
'Where are my files.lnk'*     'web war'/'
'all websites'/'

Lenovo@soni MINGW64 ~/Desktop
$ touch index.html

Lenovo@soni MINGW64 ~/Desktop
$ cd gitpractical

Lenovo@soni MINGW64 ~/Desktop/gitpractical
$ touch index.html

Lenovo@soni MINGW64 ~/Desktop/gitpractical
$ touch about.html

Lenovo@soni MINGW64 ~/Desktop/gitpractical
$ ls
about.html  index.html

Lenovo@soni MINGW64 ~/Desktop/gitpractical
$ git add index.html
fatal: not a git repository (or any of the parent directories): .git

Lenovo@soni MINGW64 ~/Desktop/gitpractical
$ ls-lart
bash: ls-lart: command not found

```

MINGW64/c:/Users/pc/Desktop/GitHub/Ekta

```

pc@VIJAY-SAXENA MINGW64 ~/Desktop/GitHub/Ekta (master)
$ git init
Initialized empty Git repository in C:/Users/pc/Desktop/GitHub/Ekta/.git/

pc@VIJAY-SAXENA MINGW64 ~/Desktop/GitHub/Ekta (master)
$

```

```
MINGW64/c/Users/pc/Desktop/GitHub/Ekta
pc@VIJAY-SAXENA MINGW64 ~/Desktop/GitHub/Ekta (master)
$ git init
Initialized empty Git repository in C:/Users/pc/Desktop/GitHub/Ekta/.git/

pc@VIJAY-SAXENA MINGW64 ~/Desktop/GitHub/Ekta (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    index.html

nothing added to commit but untracked files present (use "git add" to track)

pc@VIJAY-SAXENA MINGW64 ~/Desktop/GitHub/Ekta (master)
$ git add index.html
warning: in the working copy of 'index.html', LF will be replaced by CRLF the next time Git touches it

pc@VIJAY-SAXENA MINGW64 ~/Desktop/GitHub/Ekta (master)
$ git status
```

```
pc@VIJAY-SAXENA MINGW64 ~/Desktop/GitHub/Ekta (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   index.html

pc@VIJAY-SAXENA MINGW64 ~/Desktop/GitHub/Ekta (master)
$ touch index2.html
```

```
pc@VIJAY-SAXENA MINGW64 ~/Desktop/GitHub/Ekta (master)
$ touch index2.html

pc@VIJAY-SAXENA MINGW64 ~/Desktop/GitHub/Ekta (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   index.html

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    index2.html
```



```
pc@VIJAY-SAXENA MINGW64 ~/Desktop/GitHub/Ekta (master)
$ git config --global user.name
Vijay Shekhar

pc@VIJAY-SAXENA MINGW64 ~/Desktop/GitHub/Ekta (master)
$ git config --global user.email
vijaysaxena222319@gmail.com

pc@VIJAY-SAXENA MINGW64 ~/Desktop/GitHub/Ekta (master)
$ |
```

```
MINGW64/c/Users/pc/Desktop/GitHub/Ekta

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch master
#
# Initial commit
#
# Changes to be committed:
#   new file:   index.html
#
# Untracked files:
#   index2.html
#
~
```

```
MINGW64/c/Users/pc/Desktop/GitHub/Ekta

pc@VIJAY-SAXENA MINGW64 ~/Desktop/GitHub (master)
$ cd Ekta

pc@VIJAY-SAXENA MINGW64 ~/Desktop/GitHub/Ekta (master)
$ ls
index.html  index2.html

pc@VIJAY-SAXENA MINGW64 ~/Desktop/GitHub/Ekta (master)
$ ls|
```

```
pc@VIJAY-SAXENA MINGW64 ~/Desktop/GitHub/Ekta (master)
$ git commit -m "I have added the two new files"
[master (root-commit) fba55a3] I have added the two new files
1 file changed, 3 insertions(+)
create mode 100644 index.html
```

```
MINGW64/c/Users/pc/Desktop/GitHub/Ekta
$ ls
index.html  index2.html

pc@VIJAY-SAXENA MINGW64 ~/Desktop/GitHub/Ekta (master)
$ git commit -m "I have added the two new files"
[master (root-commit) fba55a3] I have added the two new files
1 file changed, 3 insertions(+)
create mode 100644 index.html

pc@VIJAY-SAXENA MINGW64 ~/Desktop/GitHub/Ekta (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        index2.html

nothing added to commit but untracked files present (use "git add"
to track)
```

```
MINGW64/c/Users/pc/Desktop/GitHub/Ekta

pc@VIJAY-SAXENA MINGW64 ~/Desktop/GitHub/Ekta (master)
$ ls
index2.html

pc@VIJAY-SAXENA MINGW64 ~/Desktop/GitHub/Ekta (master)
$ git checkout .
Updated 1 path from the index

pc@VIJAY-SAXENA MINGW64 ~/Desktop/GitHub/Ekta (master)
$ ls
index.html  index2.html
```

```
pc@VIJAY-SAXENA MINGW64 ~/Desktop/GitHub/Ekta (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        index2.html

nothing added to commit but untracked files present (use "git add"
to track)

pc@VIJAY-SAXENA MINGW64 ~/Desktop/GitHub/Ekta (master)
$ rm -r ../100}
```

```
pc@VIJAY-SAXENA MINGW64 ~/Desktop/GitHub/Ekta (master)
$ ls
```

```
pc@VIJAY-SAXENA MINGW64 ~/Desktop/GitHub/Ekta (master)
$ mkdir demo{1..100}
```

```
pc@VIJAY-SAXENA MINGW64 ~/Desktop/GitHub/Ekta (master)
$ echo "<h3 style='color:red;'" Hello SG ji "</h3>" >> index.html
```

```
pc@VIJAY-SAXENA MINGW64 ~/Desktop/GitHub/Ekta (master)
$ echo "<h3 style='color:red;'"
```

```
pc@VIJAY-SAXENA MINGW64 ~/Desktop/GitHub/Ekta (master)
$ echo "<h2>" Hello Sarvesh "</h2>" >> index.html
```

```
pc@VIJAY-SAXENA MINGW64 ~/Desktop/GitHub/Ekta (master)
$ cat index.html
<h1> welcome ji </h1>
<h2> Hello Sarvesh </h2>
<h3 style='color:red;'" Hello SG ji </h3>

pc@VIJAY-SAXENA MINGW64 ~/Desktop/GitHub/Ekta (master)
$ |
```

```
Lenovo@soni MINGW64 ~/Desktop/gitpractical
$ git init
Initialized empty Git repository in C:/Users/Lenovo/Desktop/gitpractical/.git/
```

```
Lenovo@soni MINGW64 ~/Desktop/gitpractical (master)
$ ls -lart .git/
total 15
drwxr-xr-x 1 Lenovo 197121  0 Nov  5 08:48 ../
-rw-r--r-- 1 Lenovo 197121 73 Nov  5 08:48 description
drwxr-xr-x 1 Lenovo 197121  0 Nov  5 08:48 hooks/
drwxr-xr-x 1 Lenovo 197121  0 Nov  5 08:48 info/
drwxr-xr-x 1 Lenovo 197121  0 Nov  5 08:48 refs/
-rw-r--r-- 1 Lenovo 197121 23 Nov  5 08:48 HEAD
-rw-r--r-- 1 Lenovo 197121 130 Nov  5 08:48 config
drwxr-xr-x 1 Lenovo 197121  0 Nov  5 08:48 objects/
drwxr-xr-x 1 Lenovo 197121  0 Nov  5 08:48 ./
```

```
Lenovo@soni MINGW64 ~/Desktop/gitpractical (master)
$ git add index.html
```

```
Lenovo@soni MINGW64 ~/Desktop/gitpractical (master)
$ git rm --cached index.html
rm 'index.html'
```



```
Lenovo@soni MINGW64 ~/Desktop/gitpractical (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        about.html
        index.html

nothing added to commit but untracked files present (use "git add" to track)
```

```
Lenovo@soni MINGW64 ~/Desktop/gitpractical (master)
$ git commit
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        about.html
        index.html

nothing added to commit but untracked files present (use "git add" to track)
```

```
Lenovo@soni MINGW64 ~/Desktop/gitpractical (master)
$ git add index.html

Lenovo@soni MINGW64 ~/Desktop/gitpractical (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   index.html

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        about.html
```

```
Lenovo@soni MINGW64 ~/Desktop/gitpractical (master)
$ git commit
On branch master

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        about.html

nothing added to commit but untracked files present (use "git add" to track)
```

```

Lenovo@soni MINGW64 ~/Desktop/gitpractical (master)
$ git commit -m "Hello everybody "
[master (root-commit) a475b01] Hello everybody
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 index.html

Lenovo@soni MINGW64 ~/Desktop/gitpractical (master)
$ ls
about.html  index.html

Lenovo@soni MINGW64 ~/Desktop/gitpractical (master)
$ ls -lart
total 16
-rw-r--r-- 1 Lenovo 197121 0 Nov  4 09:39 index.html
-rw-r--r-- 1 Lenovo 197121 0 Nov  4 09:39 about.html
drwxr-xr-x 1 Lenovo 197121 0 Nov  5 08:48 ./
drwxr-xr-x 1 Lenovo 197121 0 Nov  5 13:41 ../
drwxr-xr-x 1 Lenovo 197121 0 Nov  5 22:30 .git/

```

How to merge the code from one branch to another:-

steps are :-

1. checkout to master branch
2. check for the head using git log (Head-> master)

Note:: Head is recent commit in a branch

3. start merge using command

\$ git merge <branch-name> at master branch

4. After that Auto-merge will happen

check for the merge conflicts

5. once merge conflicts is solved put all the files to the stage from master branch.

6. and finally commit, Now

Head (Recent commit CommitId of Another branch) will be merged for Head (Recent commit commitId of Master Branch)

How to merge :-

\$ git checkout master

\$ git log

\$ press q to exit from git log is done

\$ git merge <branch-name>

\$ if not Error Auto-merge will happen

\$ git add .

\$ git commit -m "Finally Merged with any branch-name "

How stop/kill/pause merge :-

\$ git merge --abort

How to Resolve the Conflicts :-

merge conflicts arises when, two coders code at branches but write the different code on same line number,
then git cannot decide which line to keep and which to delete,
hence there arises a conflict.

hence it will organise the code in order of commit with is

Ahead another commit

when you open a conflict file:- (index.html or about.html)

<<<<<<<<<<<<<<< HEAD

This is first line code

=====

This is second line code

<<<<<<<<<<< Prashant

hence in order to resolve the conflicts, we need to solve or remove these lines.

after cleaning

1| This is first line code

2| This is second line code

\$ git add .

\$ git commit -m "code cleaned and Now ready to merge"

[new commitId] merged with <new-commitId>

difference B/w git branch and git branch --all

1. **git branch** : show all local branches

2. **Remote(cloud => github) git branch --all** : local + cloud server branches.

Help of git:-

\$ git help <topic-name>

Eg :-

\$ git help branch

\$ git branch -d <branch-name>

How to push, code to Github

1. working with .gitignore

2. touch .gitignore

Add the file-name and folder-names you dont want to sent to server.

3. Decide which branch code you want to send

4. Update 2021, Older time, git login from system

username and password

latest time :- username

personalised token :- use token instead of password.

How to generate token

1. login github

2. global settings

3. developer settings

4. peronalise Token

simple Token(classic) => Title => generate Token => Copy

save the token.

Customise Token

5. Open Credential Manager

1. window credentials

2. git remove all password and username.

6. create a Repo private/public

How to see origin url

\$ git remote -v

1. **fetch version** : url must exist then only push possible.

2. **push version** : url must exist then only push possible.

How to Add url :-

\$ git remote add origin <url-copy-paste>

for eg:-

\$ git remote add origin https://github.com/awnishmmg/finalclass-of-git.git

\$ git remote -v

(fetch) : <https://github.com/awnishmmg/final-class-of-git.git>

(push) : <https://github.com/awnishmmg/final-class-of-git.git>

Finally push the code

\$ git push origin <branch-name>

or

\$ git push -f origin <branch-name>

branch :

master

dev

tester

\$ git push -f origin master

\$ git push -f origin dev

\$ git push -f origin tester

[SELF LEARNING]

Git Repository cmd :--

1. mkdir foldername -----> create folder
 2. cd -----> change directory
 3. git touch filename -----> create new file (index.html ,
about.html)
 4. git init -----> for file initialization
 5. ls ----->
 6. ls -lart -----> for check all directory files
 7. git add filename -----> file in tracking mode
-
-

git branch

git branch soni

git checkout soni -----> working /switching account

git log

git add. -----> add all file current directory

cat filename

cat index.html -----> show all code

git checkout master

git merge

git --abort -----> stop the merging

git commit -m "....."

Two Types

1. Ansi Standard

1. Natural

- i.) Equi-join
- ii.) Non-Join
- iii.) Normal Join (inner Join)

2. Non Ansi Standard

i.) Outer Joins

- a) Left
- b) right
- c) Full Outer Joins

ii.) self joins

iii.) cartesian Product (cross Join)

Insert into tablename values("","","");

Insert into values("","","");

tablename : college

Insert into college values("","","");

insert college values("","","");

A : Atomicity

C : consistency

I : Isolated

D : Durability

Atoms consist of isolated Durability.

JIRA

Jira :-

1. Pms Project management
2. it is product of Atlassian (Australian Company)
3. 60,000 Teams and world Wide use
4. it is Running on Cloud Server
5. Jira
 1. user => for every new user -> sub-domain create
 2. Team(Organisation | group) -> sub-domain create

Ex:-

[https://\[shankar\].atlassian.net\(com\)](https://[shankar].atlassian.net(com))

|
username/organisation-name

Organisation name : infosis

[https://infosis.atlassian.net\(com\)](https://infosis.atlassian.net(com))

6. Jira is a paid Software
7. Jira was launched in 2002
8. Jira is a Product

Marketplace of Jira :-

Marketplace : product sell but demo is possible for free

Jira has 1000+ paid and free software

Software Categories:-

free : always free

paid : always paid

proprietary s/w : paid some time free or demo (15 days or 30 days)

products of Jira:-

1. Jira Confluence

2. JiraOps (Administration) : opsgenie

3. Jira Service Management : Others like Finance, legal, salary, Support etc Jira Service Management.

Jira Confluence : All documents related to SDLC like SRS, licensing Audit, security Audit

JiraOps : Administration level work, user create, user delete, user invite, user block.(Opsgenie)

Integration Jira <-----Connect-----> GitHub (marketplace)

Important links to Jira:-

id.atlassian.com : Jira Login Mode => Already Login => Dashboard

Jira Starting Mode

start.atlassian.com