**WHAT IS VERSION CONTROL SYSTEM?**

A Version Control System is a tool that helps you track changes to files over time — especially useful when you're writing code.

**🛠️ Types of VCS**

|  |  |  |
| --- | --- | --- |
| **Type** | **Description** | **Example** |
| **Local** | Tracks changes on your own computer only | RCS |
| **Centralized** | All users pull from and push to one server | SVN, CVS |
| **Distributed**  **(2005) - Linus Torvalds** | Everyone has their own full copy of the project | **Git**, Mercurial |
|  |  |  |

**SETTING UP GIT ON YOUR MACHINE**

1. **Install git :-** [**Git Download**](https://www.google.com/search?q=git-scm.com&rlz=1C1CHBF_enIN1034IN1034&oq=git+&gs_lcrp=EgZjaHJvbWUqBggCEEUYOzIHCAAQABiPAjIPCAEQLhgKGMcBGNEDGIAEMgYIAhBFGDsyBggDEEUYPDIGCAQQRRg8MgYIBRBFGDwyBggGEEUYPDIGCAcQRRg80gEIMzQyMmowajeoAgCwAgA&sourceid=chrome&ie=UTF-8)
2. **Configure username and email**

**Cheat Sheet - Git Commands Used in This Section**

In this section, you installed and configured Git on your machine. To configure Git, you learned about different commands.

You used the following Git commands

1. **Show the version of your Git installation**: git -v
2. **Show the configurations**: **git config -l**
3. **Configure your username and email:** git config --global user.name "Amardeep Rana" , git config --global user.email "amardeeprana2905@gmail.com"

The **username and email are used by Git for every commit** that you will create. Without this configuration, **you are not able to commit to a repository**.

To show the origin of the configurations, you learned about the **--show-origin parameter**. It shows you in which file a configuration is located:

git config -l --show-origin

This command above allows you to find the . gitconfig file that is in your user directory. That *.gitconfig* file contains now the username and the email that you configured.

**Now, Git is set up and configured**. This means you're ready for some work! :-)

In the next section, you will learn how to work with local repositories.

**WORKING WITH LOCAL REPOSITORIES**

**CREATE YOUR LOCAL REPOSITORY**

**CMD LINE**

1. **mkdir Myproject** :- create
2. **cd Myproject** :- change directory, visit/ open that folder
3. **explorer .**  :- open file
4. **git init**  :- To create new git repository on this folder (May be hidden so unhide)
5. Note : open other folder use **cd ..\foldername**

**THREE MAIN AREA**

**Staging area**

**3.Commit**

**.git folder**

1. **checkout**

**Repository**

**(version database)**

**Working Directory**

**2.Staging**

**Project folder**

**Any changes** , **edit in working area it goes staging area then repository**.

**CREATE YOUR FIRST REPOSITORY**

1. **git status** :- Check on branch main / no commit
2. create **readme.txt file** on Myproject (Welcome to MyProjecty !)
3. **git add .**  :- to identify current file / area
4. git status :- check
5. **git commit -m “create readme file”** :- to commit new repository

**IF ANY CHANGES IN THE SOURCE CODE, AFTER THEN USE THIS STEPS**

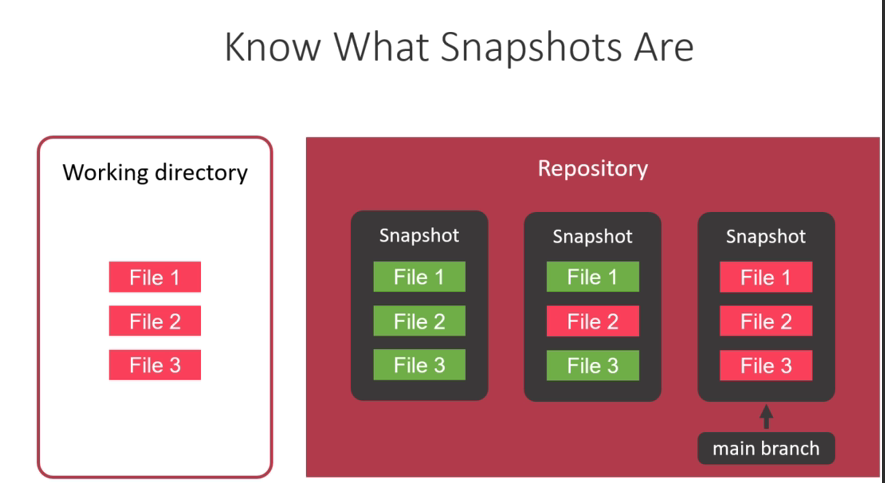
1. Let assume you modified
2. **Git status** :- check for status in this show “changes not staged for commit :”

Modified : show in red

1. **Git add .** :- which you modified inside the file it will be added.
2. **Git status**  :- check for status in this show “Changes to be committed::”

Modified : show in green

1. (optional) **git diff** (work file name) :- If you want to **check what is different before and after in modified**
2. (optional) **git diff –staged** :- its show **what you change that file after add again check**
3. (optional) **git commit** :- it goes to that editor which you specified installation time .
4. **git commit -m “add author to readme”** :- add author to readme
5. **git status** :- **On branch main, nothing to commit, working tree clean**



**Commit is a Snapshot**

Git create for every snapshot / commit a hash vale

3f9a5a6e9c2b7d3efdd9c6b9eabf7a2c8df0f2a0

* Identify a snapshot/commit
* Can be used to check out a specific commit

**Look at the History of your code**

* **git log / git log –pretty=oneline**

A screenshot of a computer program

AI-generated content may be incorrect.**Ignore files in the working directory**

* **create** a **“.gitignore “** file , open in notepad write “**compiledoutput.txt**”
* create compiledoutput.txt file
* git status :- Untracked files: .gitignore
* git add .
* git status :- new file: .gitignore
* **git commit -m "Add .gitignore file"** :- After adding a file to the staging area, you want to commit it.
* git status :- On branch main, nothing to commit, working tree clean

**Create & merge on Branch**

You used the following Git commands

1. List the local branches: **git branch**
2. Create a new branch with the name "feature/AddTwitterHandle": **git branch feature/AddTwitterHandle**
3. Checkout the new branch or switch to branch : **git checkout feature/AddTwitterHandle**
4. Checkout the main branch: **git checkout main**

**Merge changes** from the feature branch into the main branch. This command assumes that you have the main branch checked out: **git merge feature/AddTwitterHandle**

Show a graph when showing the history: **git log --pretty=oneline --graph**

Now you know how to branch and merge your code, which can be helpful when you want to work on different features in parallel.

**Examples**

**Create Branch**

1. Open the Terminal and type **cd Myproject** open the folder on terminal.
2. **gits status** : on main branch
3. **git branch feature/AddTwitterhandle** : Create branch use (git branch feature/”name of that” ).
4. **git branch** : List of the branches
5. **git status** : on main branch
6. git checkout feature/AddTwitterHandle : For switching the branch / switched to branch .
7. **git status** : Now, on branch feature/AddTwitterHandle means branch changed
8. Now you can add commit on **readme.txt** file like Twitter : @amardeep29.
9. **git status** : not commit
10. **git add .**
11. **git commit -m “feature/AddTwitterHandle”**
12. **git status** : Committed show modified

**Merge Branch**

1. **git branch** : for check on which branch
2. **git merge feature/AddTwitterHandle** : Now merged
3. **git status**
4. **git branch -d feature/AddTwitterHandle : deleting that branch**
5. **git branch** : now you are in main branch

**Handle Merge conflict**

1. Create branch :

* git branch feature/lastname
* git branch feature/middlename

1. git branch : check
2. git checkout feature/lastname
3. Now you can add Author of last name on **readme.txt** file
4. git status : not commit
5. git add .
6. git commit -m “feature/lastname”
7. git status : committed
8. git checkout main : switchted to main branch for merging
9. git merge feature/lastname : merged
10. git branch : list show
11. git branch -d feature/lastname : deleted
12. git branch : show only , feature/middlename and main branch

**Now for feature/middlename**

1. same follow steps 2 – 9 as above
2. git merge feature/middlename : its Show Conflicts
3. Check **readme.txt file**  and  **edit it by removing and correcting right information.**
4. **git status : show conflicts**
5. **git add .**
6. git status
7. git branch
8. **git commit -m “merge branch feature/middlename” : merged**
9. git status
10. git branch
11. **git branch -d feature/middlename : deleted feature/middlename**
12. git branch : Now on only main branch
13. **git log –pretty==oneline** : Show the history of works

**Create Repository on Github**

You know the way.

**Push your Code to the Remote Repository**

**create a new repository on the command line**

1. echo "# git-for-beginners-my-project" >> README.md
2. git init
3. git add README.md
4. git commit -m "first commit"
5. git branch -M main
6. git remote add origin https://github.com/amardeeprana29/git-for-beginners-my-project.git
7. git push -u origin main

**push an existing repository from the command line**

1. git remote add origin <https://github.com/amardeeprana29/git-for-beginners-my-project.git>
2. git remote : origin
3. git remote -v : Showing details
4. git remote show origin : show fetch, push,head branch
5. git branch -M main
6. git push -u origin main : branch 'main' set up to track 'origin/main'.

**Clone the Repository**

1. copy the link from Github Repository code
2. open terminal
3. mkdir myproject\_copy
4. cd myproject\_copy
5. git clone <https://github.com/amardeeprana29/git-for-beginners-my-project.git> .
6. explorer.
7. Now add new commit process means edit or add text in **readme.txt**
8. git remote : origin
9. git remote -v : Showing details
10. git remote show origin : show fetch, push,head branch and remote branch main tracked
11. git status & git add .
12. git commit -m “Add website to readme”
13. git push

**Pull changes from the remote**

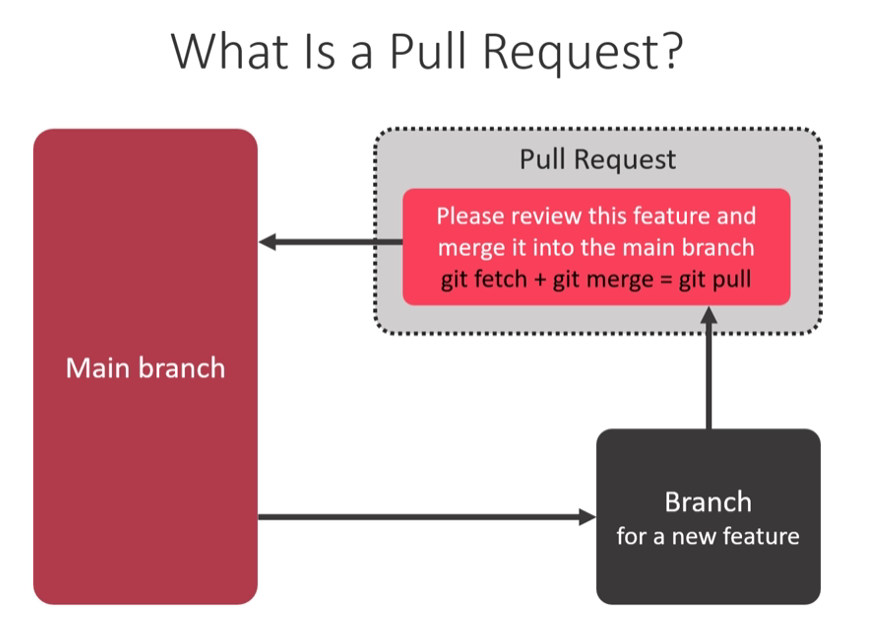
**git pull**

git pull = git fetch + git merge

Fetch changes from the remote repository without merging them into your local branch:

**git fetch**

**git merge origin/main**



**Create and merge a Pull Request**

1. **git pull :** Already up to date
2. **notepad readme.txt**
3. **git branch** : main
4. **git branch feature/AddGithub**
5. **git checkout feature/AddGithub**
6. **notepad readme.txt :** add Github : paste link
7. **git status**
8. **git add .**
9. **git commit -m “Add Github to readme”**
10. **git status**
11. **git push origin feature/AddGithub**
12. **git branch**
13. **git checkout main**
14. **git branch**
15. **git merge feature/AddGithub :** You see fast – forward readme.txt
16. **git status**
17. **git push**
18. **git branch -d feature/AddGithub**
19. **git branch**

**Fork a repository on Github**

* Copy link of user GitHub profile 🡪 sitting 🡪 Collaborators 🡪 Add people
* Copy link of repository from code 🡪 Fork 🡪 create new fork 🡪 do if you want to change 🡪 Commit change
* You request Pull Request

**Merge the pull request from people**

1. **git pull** : already up to date
2. **git branch -a**
3. **git branch person-main** : People 🡪 Name of that person
4. **git checkout person-main**
5. **git branch : now in person-main**
6. **copy of the repository link**
7. **git pull paste link main** : Enter now fast – forward readme.mn & readme.txt
8. **explorer .**
9. **git branch**
10. **git checkout main**
11. **git merge person-main** : Enter now fast – forward readme.mn & readme.txt
12. **git push**