YouTube

Credit : @code.xam (Follow : instagram)
@gitxam (Follow : instagram)
@javaxam (Follow : instagram)

Founder: Subham

A Short History Of A Git

As with many great things in life, Git began with a bit of creative destruction and fiery controversy.

GitXam

The Linux kernel is an open source software project of fairly large scope. During the early years of the Linux kernel maintenance (1991–2002), changes to the software were passed around as patches and archived files. In 2002, the Linux kernel project began using a proprietary DVCS called BitKeeper.

In 2005, the relationship between the community that developed the Linux kernel and the commercial company that developed BitKeeper broke down, and the tool's free-of-charge status was revoked. This prompted the Linux development community (and in particular Linus Torvalds, the creator of Linux) to develop their own tool based on some of the lessons they learned while using BitKeeper.

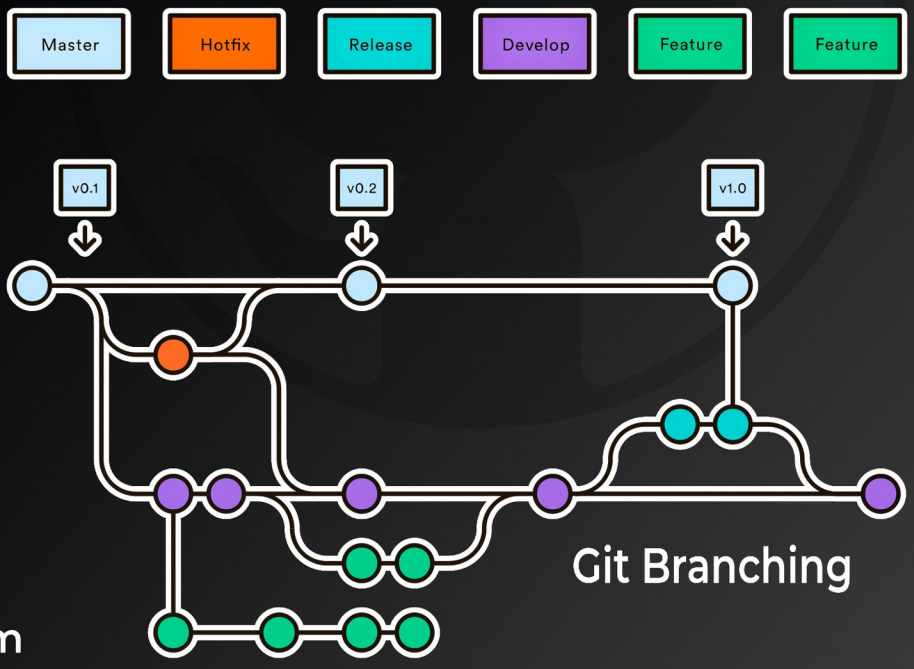
GitXam

Some of the goals of the new system were as follows:

- Speed
- Simple design
- Strong support for non-linear development (thousands of parallel branches)
- Fully distributed
- Able to handle large projects like ‘ the Linux kernel efficiently (speed and data size)

GitXam

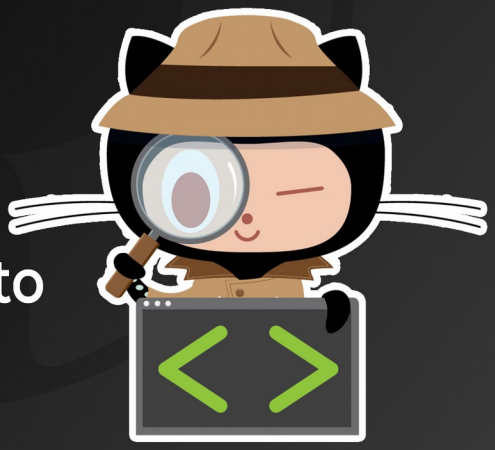
Since its birth in 2005, Git has evolved and matured to be easy to use and yet retain these initial qualities. It's amazingly fast, it's very efficient with large projects, and it has an incredible branching system for non-linear development.



Basic Question Answer

What is GitHub and why it is used?

GitHub is a web-based interface that uses Git, the open source version control software that lets multiple people make separate changes to web pages at the same time.
As Carpenter notes, because it allows for real-time collaboration, GitHub encourages teams to work together to build and edit their site content.



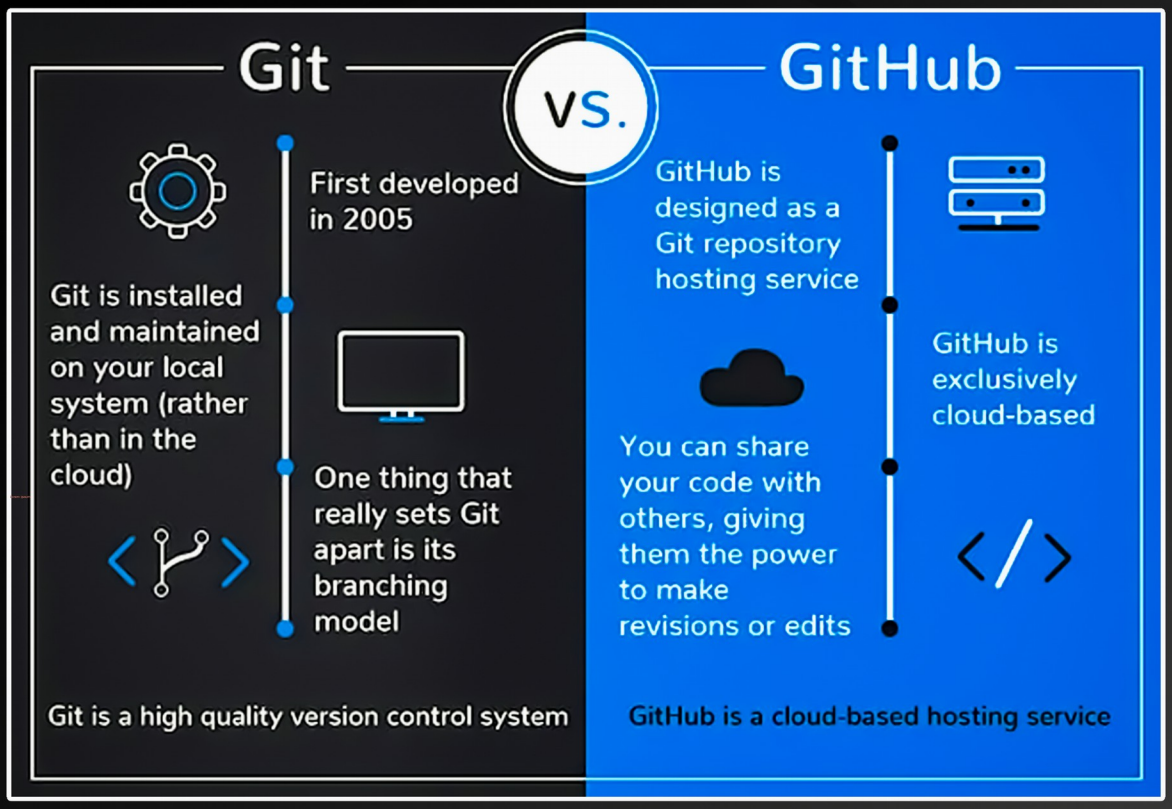
GitXam

What's the difference between Git and GitHub?

Git is a version control system that lets you manage and keep track of your source code history.

GitHub is a cloud-based hosting service that lets you manage Git repositories. If you have open-source projects that use Git, then GitHubis designed to help you better manage them.

GitXam


Git
VS.
GitHub
First developed in 2005
Git is installed and maintained on your local system (rather than in the cloud)
One thing that really sets Git apart is its branching model
GitHub is designed as a Git repository hosting service
GitHub is exclusively cloud-based
You can share your code with others, giving them the power to make revisions or edits
Git is a high quality version control system
GitHub is a cloud-based hosting service

GitXam

Is GitHub account free?

You can use organizations for free, with GitHub Free, which includes unlimited collaborators on unlimited public repositories with full features, and unlimited private repositories with limited features.

GitHub offers free private repositories for unlimited collaborators

GitXam

Is GitHub necessary?

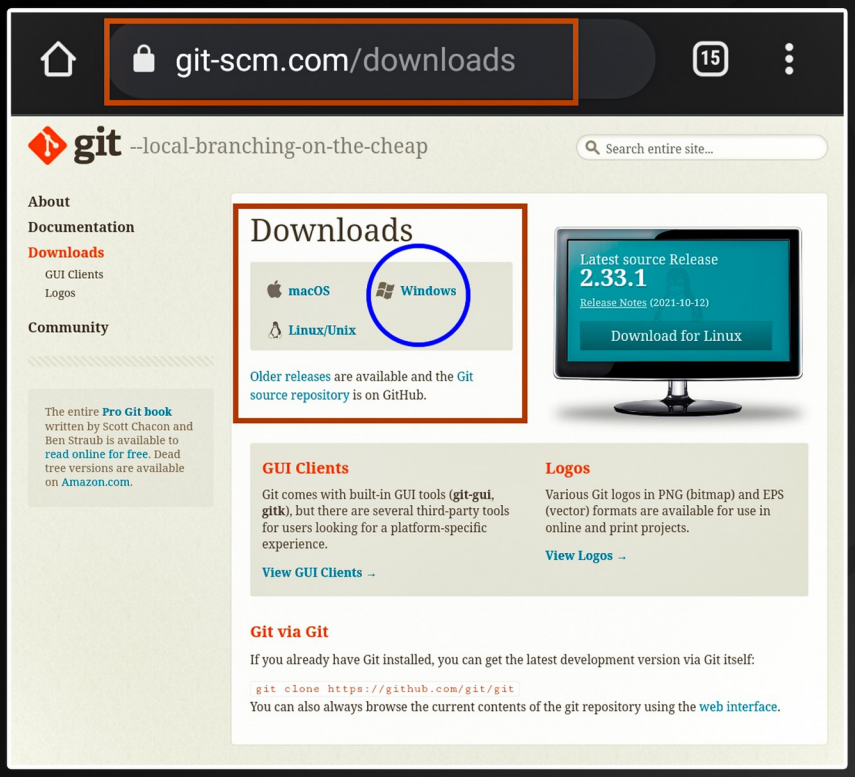
If you only want to keep track of your code locally, you don't need to use GitHub. But if you want to work with a team, you can use GitHub to collaboratively modify the project's code. When you're done filling out the information, press the 'Create repository' button to make your new repo.

GitXam

Installing Git

[Git - Downloads (git-scm.com)]

There are few ways to install Git on Windows. The most official build is available for download on the Git website. Just go to git-scm.com/download



GitXam

The Command Lines

Experiment
Fail
Learn
Repeat

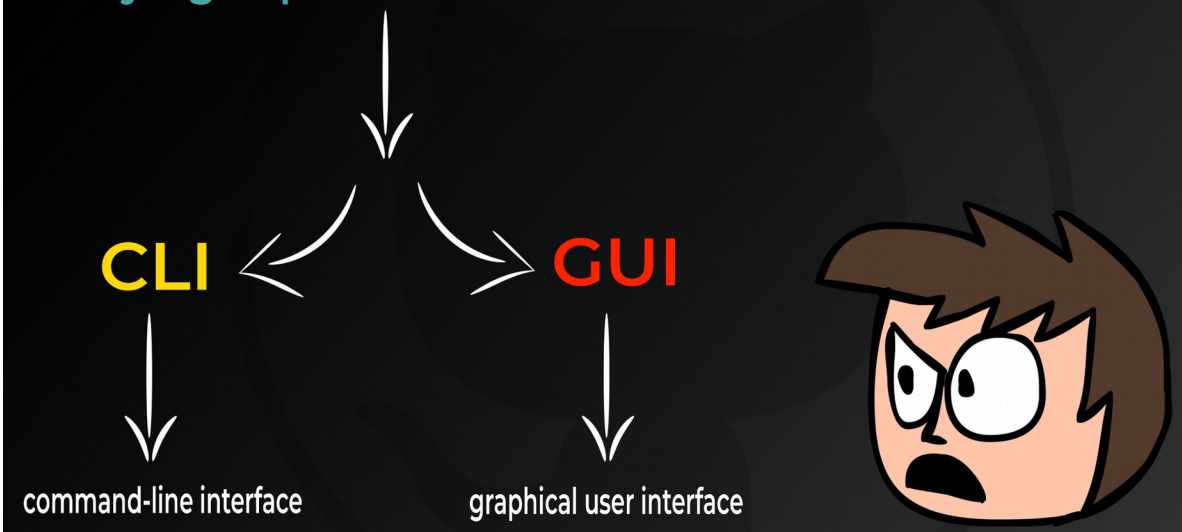
Topic:

The Command Line

GitXam

There are a lot of different ways to use Git.

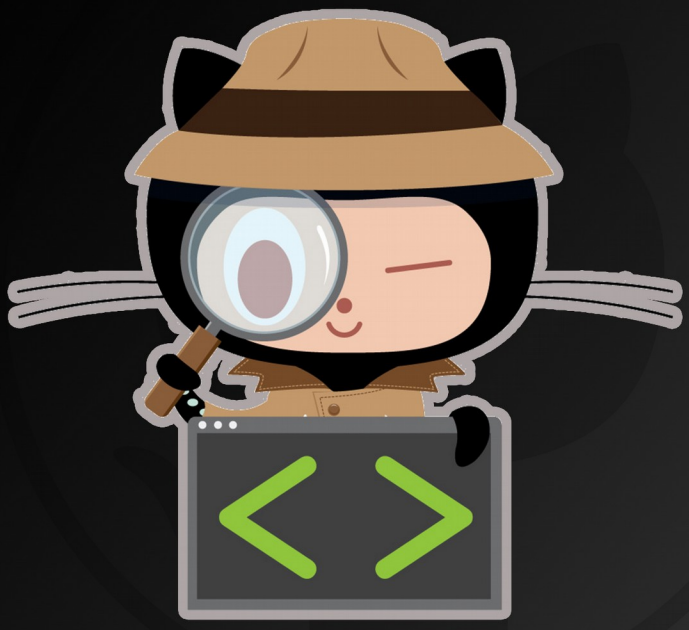
There are the original command-line tools , and there are many graphical user interfaces of varying capabilities



What should we use?

GitXam

We will be using Git on the command line. (CLI)



But Why?

GitXam

The command line is the only place you can run all Git commands

most of the GUIs implement only a partial subset of Git functionality for simplicity

Any benefits ?

GitXam

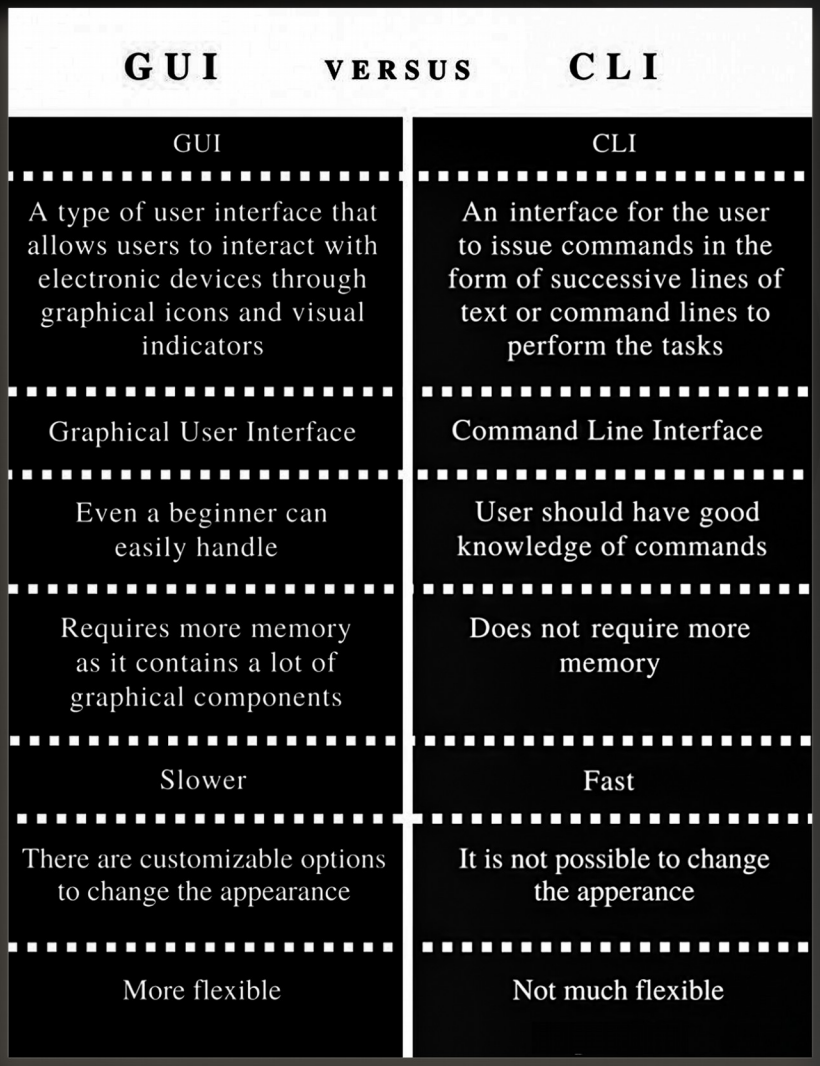
Let's Compare...

If you know how to run the command-line version, you can probably also figure out how to run the GUI version, while the opposite is not necessarily true. Also, while your choice of graphical client is a matter of personal taste, all users will have the command-line tools installed and available.

Bro I'm not so genius tell me with some details

GitXam

Detail Compare...



GitXam

So we will expect you to know how to open Terminal in macOS or Command Prompt or PowerShell in Windows. If you don't know what we're talking about here.

you may need to stop and research that quickly so that you can follow the rest of the examples

Please Help me . I'm your brother/sister. I'm beginner.

GitXam

Initial Setup 1

Experiment
Fail
Learn
Repeat

Topic:

Initial Setup 1

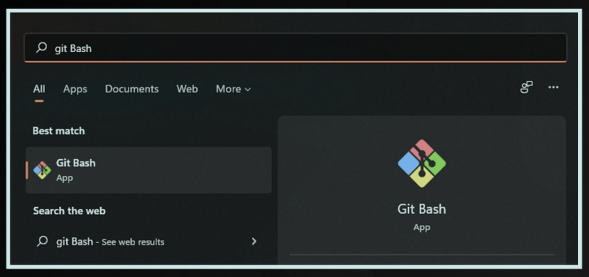
GitXam

Now that you have Git on your system.

You wanna see ?

Open any Terminal (PowerShell / CMD / Git Bash)

1. Here I'm using Git Bash (It's also a Terminal)



2. Now Type git

But how ?

GitXam

MINGW64:/c

```
maity@Subham MINGW64 ~
$ git
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           [--super-prefix=<path>] [--config-env=<name>=<envvar>]
           <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
   clone             Clone a repository into a new directory
   init              Create an empty Git repository or reinitialize an existing
one

work on the current change (see also: git help everyday)
   add               Add file contents to the index
   mv                Move or rename a file, a directory, or a symlink
   restore           Restore working tree files
   rm                Remove files from the working tree and from the index
   sparse-checkout   Initialize and modify the sparse-checkout

examine the history and state (see also: git help revisions)
   bisect            Use binary search to find the commit that introduced a bug
   diff              Show changes between commits, commit and working tree, etc
   grep              Print lines matching a pattern
   log               Show commit logs
   show              Show various types of objects
   status            Show the working tree status

grow, mark and tweak your common history
   branch            List, create, or delete branches
   commit            Record changes to the repository
   merge             Join two or more development histories together
   rebase            Reapply commits on top of another base tip
   reset             Reset current HEAD to the specified state
   switch            Switch branches
   tag               Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
   fetch             Download objects and refs from another repository
   pull              Fetch from and integrate with another repository or a local
 branch
   push              Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
See 'git help git' for an overview of the system.
```

Let's zoom in

MINGW64:/c/Users/maity

```
maity@subham MINGW64 ~
$ git
```

Then ?

GitXam

Current Working Directory :

The current working directory is the directory in which the user is currently working in. Each time you interact with your command prompt, you are working within a directory.

You wanna see the present working directory?

3. Now Type pwd

```
maity@Subham MINGW64 ~
$ pwd
/c/Users/maity
```

Easy, but then ?

GitXam

Change Directory :

To change this current working directory, you can use the "cd" command (where "cd" stands for "change directory").

How do I use the CD command?

1. Let's assume you are in the default directory.

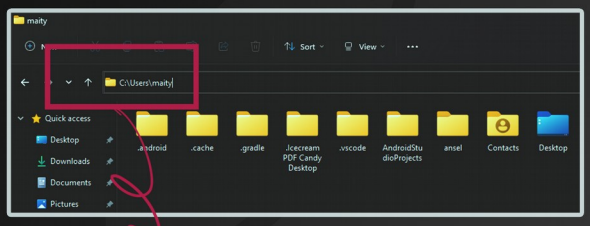
Our default Directory is

```
$ pwd
/c/Users/maity
```

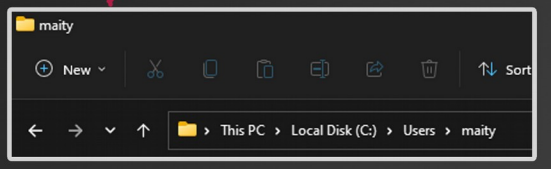
Can I change the directory folder?

2. Let's open the directory

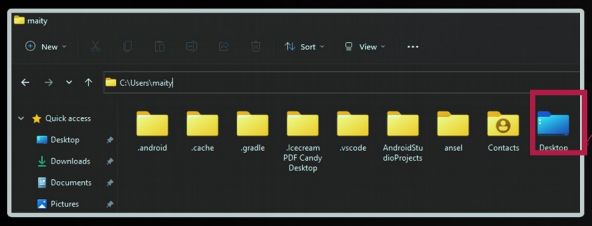
[Any Folder Open>Paste The Directory Adresss (C:\Users\maity)]

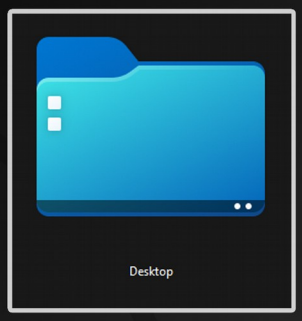


Zoom in



GitXam





You wanna go from the default directory to the desktop folder's directory. Right ?

4. Now Type cd Desktop

```
$ cd Desktop
```

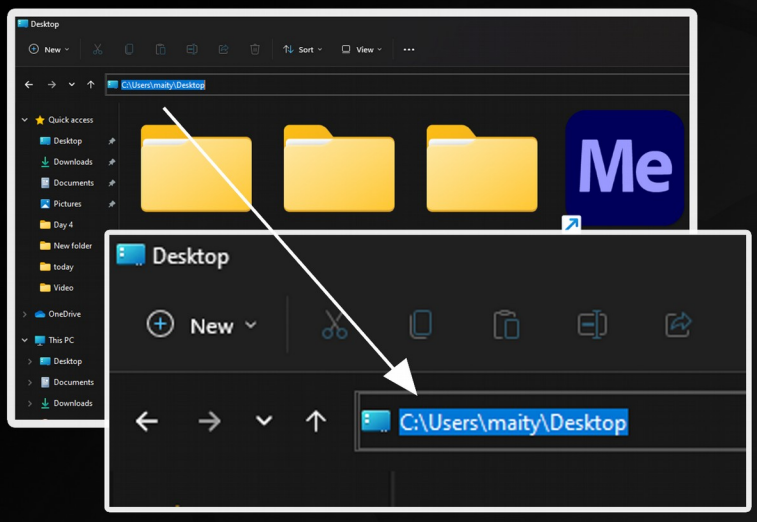
5. Enter pwd again to check the most recent directory

```
maity@Subham MINGW64 ~
$ pwd
/c/Users/maity

maity@Subham MINGW64 ~
$ cd Desktop

maity@Subham MINGW64 ~/Desktop
$ pwd
/c/Users/maity/Desktop
```

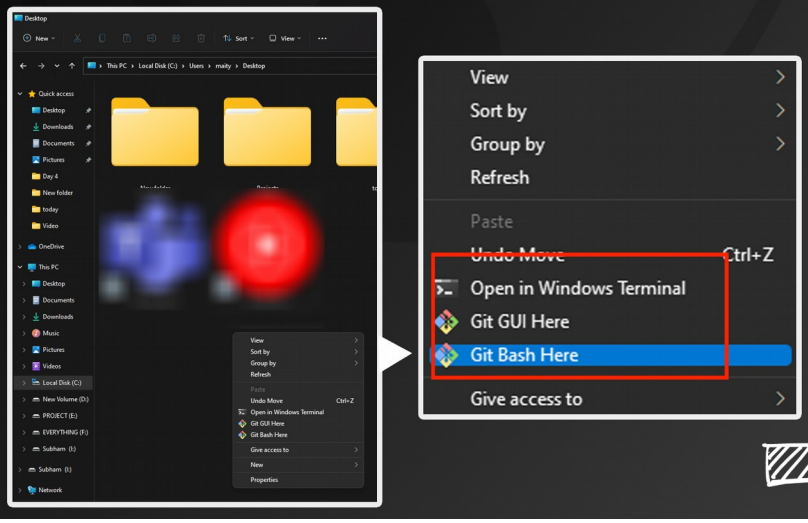
but it is difficult please tell me some easy solution

GitXam

1. It’s not a rocket science just open the Desktop Folder in your PC

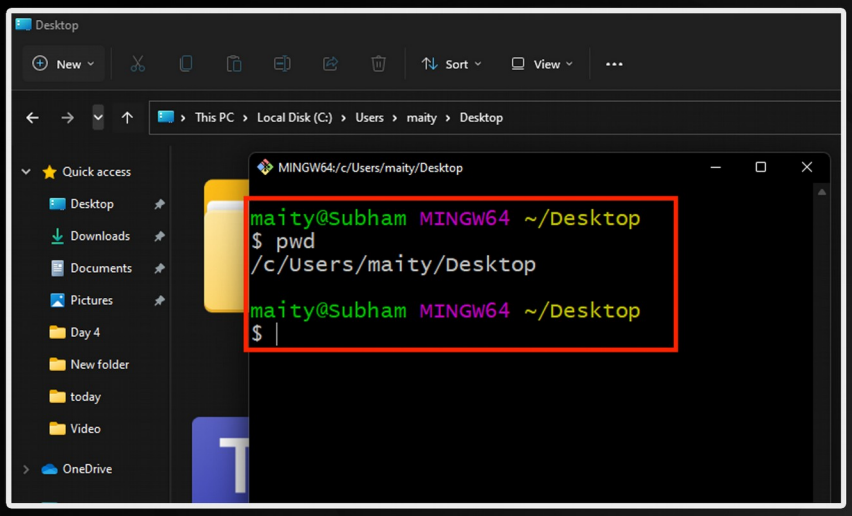


2. Now hold down your Shift button on your keyboard and click the right buttor of your mouse simultaneously and use any Terminal



GitXam

6. Now Type Agin pwd



Kudos

GitXam

Initial Setup 1 Code

git

pwd

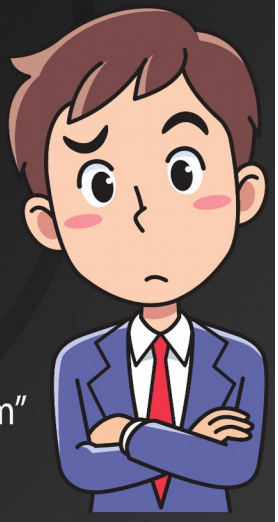
cd

Initial Setup 2

Setup Your Identity (user name and email)

1. The first thing you should do when you install Git is to set your **user name** and **email address**. This is important because every Git commit uses this information, and it's immutably baked into the commits you start creating:

1. Command Line



```
$ git config --global user.name "gitxam"
$ git config --global user.email "codexam.personal@gmail.com"
```

GLobal

GitXam

Global

you need to do this only once if you pass the **--global** option, because then Git will always use that information for anything you do on that system.

If you want to override this with a different name or email address for specific projects, you can run the command without the --global option when you're in that project.

Many of the GUI tools will help you do this when you first run them.

Checking Your Settings

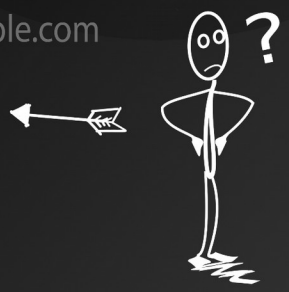
2. If you want to check your configuration settings, you can use the **git config --list** command to list all the settings Git can find at that point:

2. Command Line

$ git config --list

```
user.name=John Doe
user.email=johndoe@example.com
color.status=auto
color.branch=auto
color.interactive=auto
color.diff=auto
```

$ git config --list



I can see keys more than one but why?

GitXam

Keys more than once

You may see keys more than once, because Git reads the same key from different files ([path]/etc/gitconfig and ~/.gitconfig, for example). In this case, Git uses the last value for each unique key it sees.

GitXam

specific key's

3. You can also check what Git thinks a specific key's value is by typing git config <key>:

3. Command Line

```
$ git config user.name
gitxam

$ git config user.email
codexam.personal@gmail.com
```

GitXam

Your Editor

4.you can configure the default text editor that will be used when Git needs you to type in a message. If not configured, Git uses your system's default editor.If you want to use a different text editor, such as Emacs, you can do the following:

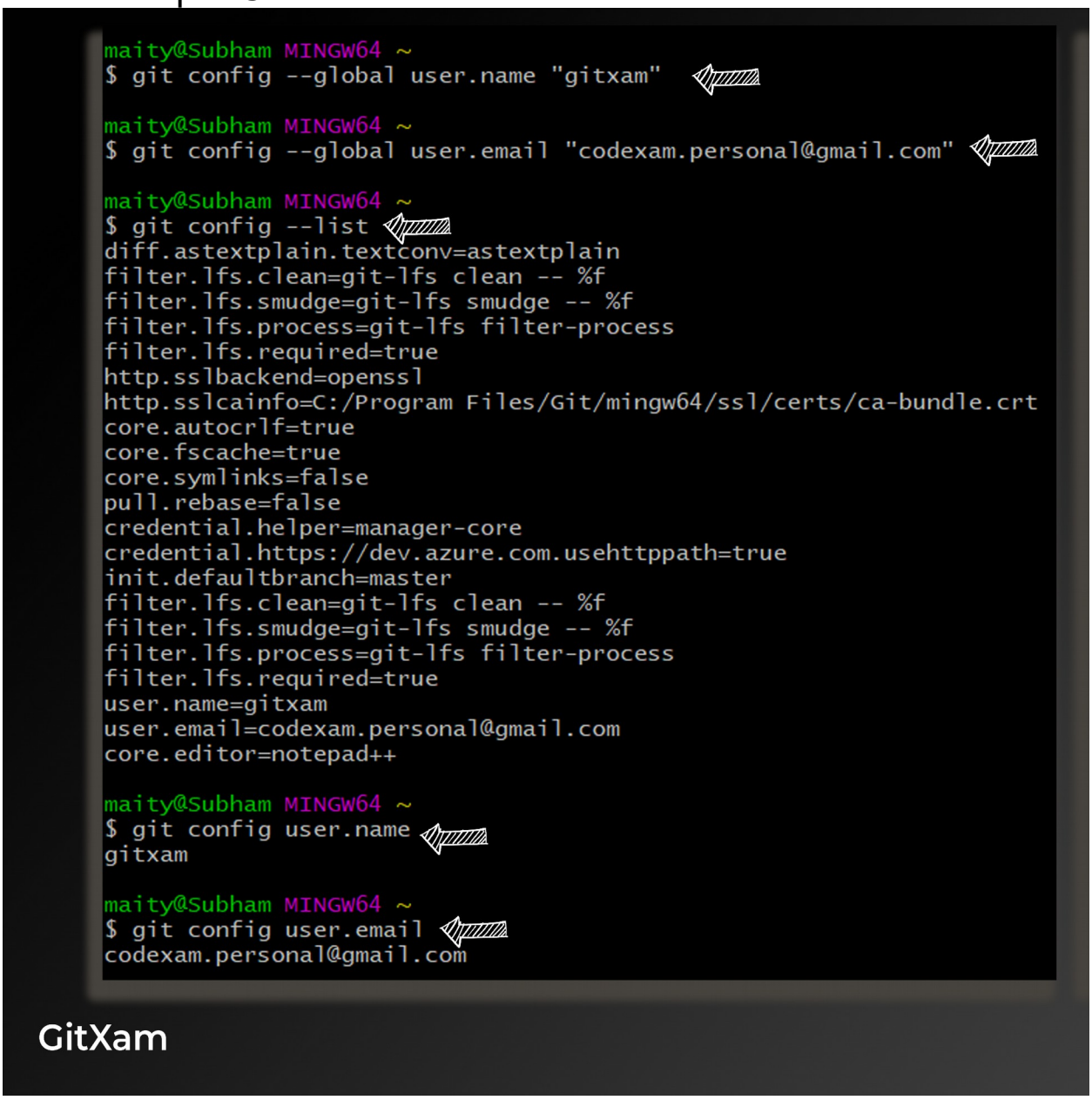
Vim, Emacs and Notepad++ are popular text editors often used by developers on Unix-based systems like Linux and macOS or a Windows system.

4. Command Line

```
$ git config --global core.editor emacs
$ git config --global core.editor vim
$git config --global core.editor notepad++
```

GitXam

Initial Setup 2 Code



Editor

```
maity@Subham MINGW64 ~
$ git config --global core.editor vim

maity@Subham MINGW64 ~
$ git config --global core.editor emacs

maity@Subham MINGW64 ~
$ git config --global core.editor notepad++
```

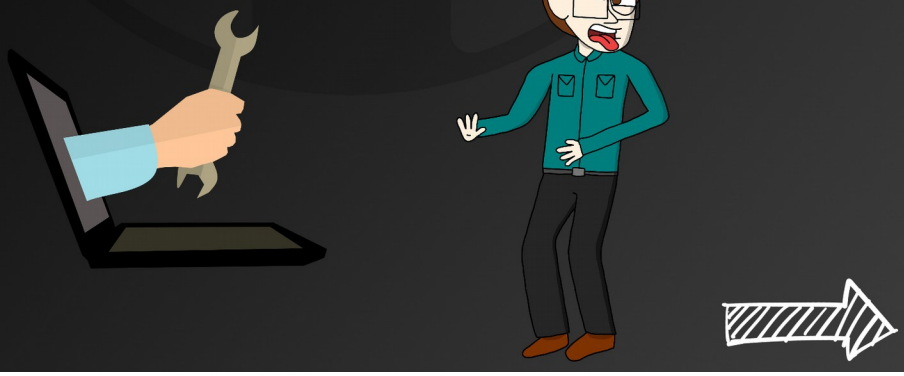
Three Architecture of Git

Three Stage Architecture

1. Let's assume you are working on a project which has an GitXam.html file ,a folder named Main and engine.js file. Now you have completed this project and made it version-1 .

   Maybe you wanna add more features into it right ?

   Step 1: if anything goes wrong, you could roll back to version 1 which runs perfectly. In this case, you will take a backup of the project. Which means you will save a state of this project where it is running without any errors. This process is called "Commit".

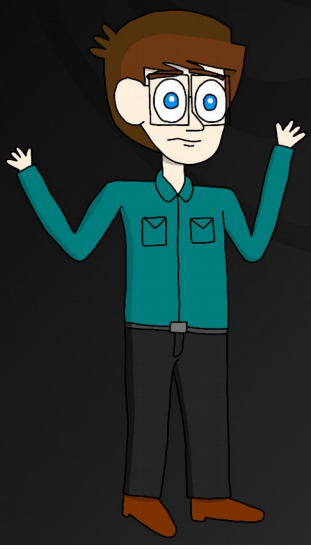


GitXam

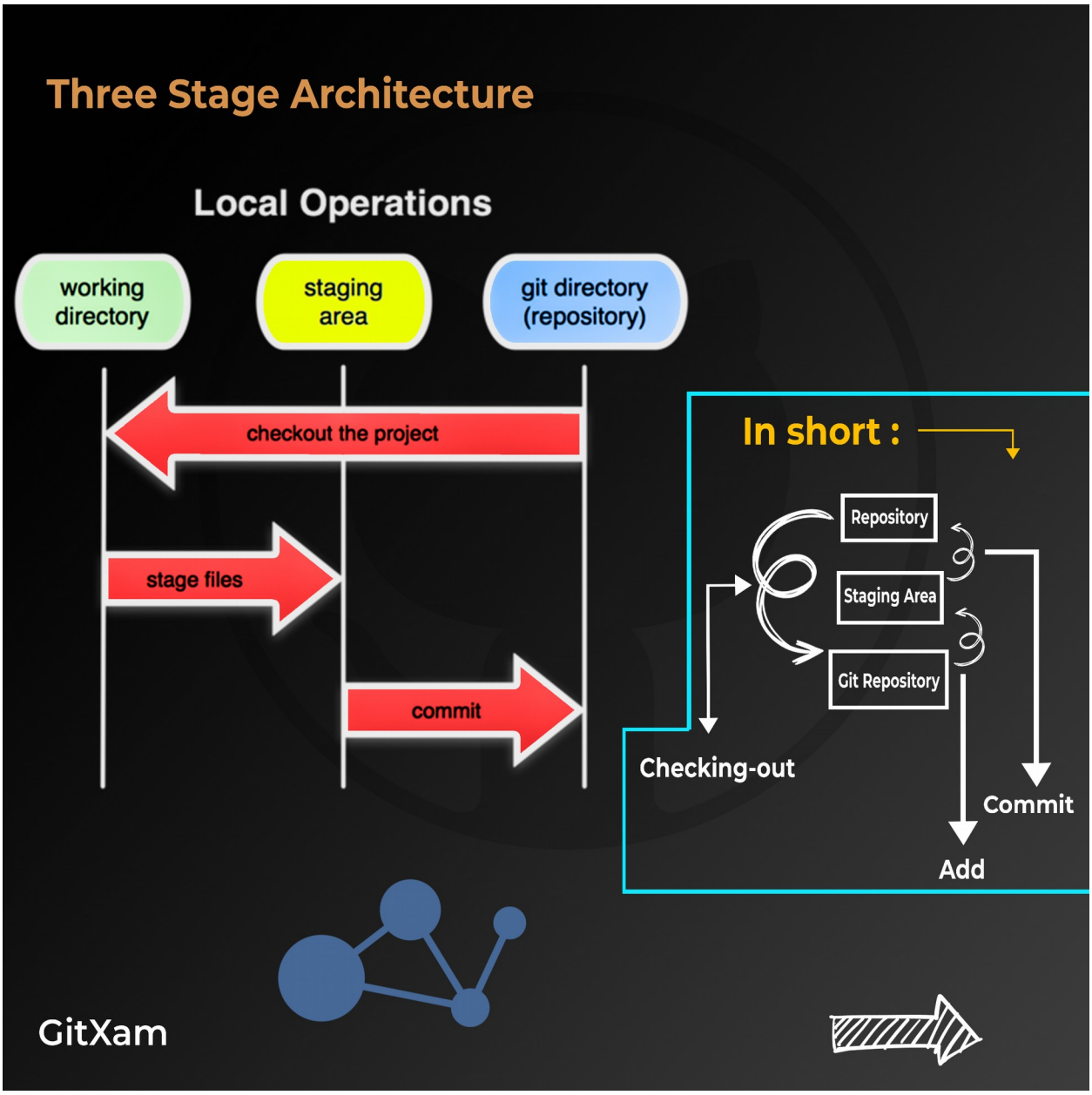
Three Stage Architecture

2. Now you have committed this project as C1 and continued to work. Now You have changed GitXam.html and also the other files, but somehow engine.js file is not working anymore.

   Step2: So, in this case, you will just put GitXam.html file on the second backup file and pull the other working files from the first backup file. This is where **Three-stage architecture** comes.



GitXam

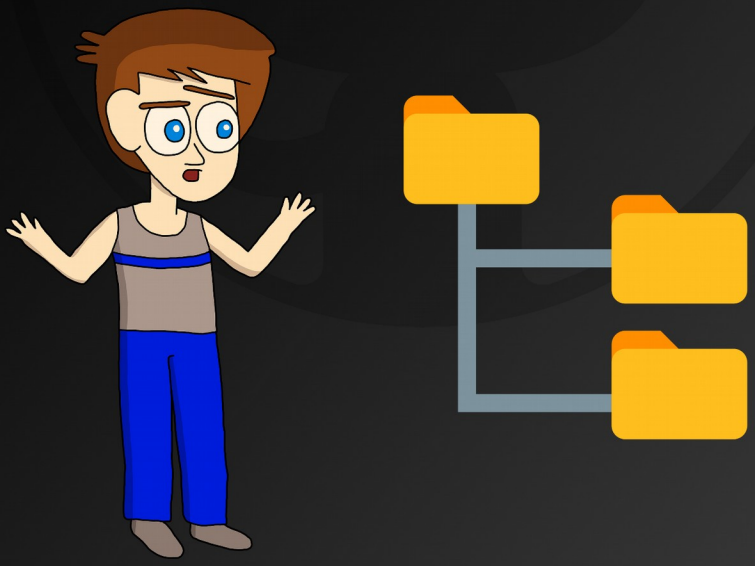

Three Stage Architecture
Local Operations
working directory
staging area
git directory (repository)
checkout the project
stage files
commit
In short :
Repository
Staging Area
Git Repository
Checking-out
Commit
Add
GitXam

We have 3 areas in the Three-stage Architecture

1. Working Directory: Working copy is the place where you make your changes. Whenever you edit something, it is saved in working copy and it is a physically stored in a disk(SSD/HDD).

In short :

The working directory is the folder in your local computer where the project files and folders are stored.



GitXam

We have 3 areas in the Three-stage Architecture

2. Staging Area : The staging area has those files who are supposed to go to the next commit. Only those files which are needed to go to the next commit stay in the staging area. Like, suppose you are working on that previous project and you have upgraded the **Codexam.html** file but somehow you broke the **engine.js** file, so now you will just add the **Codexam.html** file to stage area so it can be added to the next commit and the **engine.js** will be used from a previous version. And as you have not staged the broken **engine.js** file, the broken fill will not be committed.

In short :

The staging area is like a rough draft space, it's where you can **git add** the version of a file or multiple files that you want to save in your next commit (in other words in the next version of your project).

GitXam

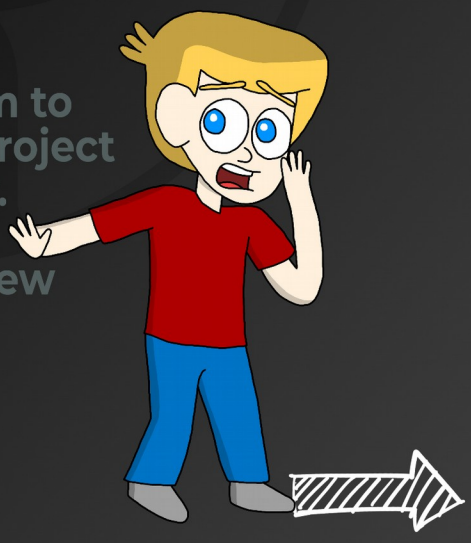
We have 3 areas in the Three-stage Architecture

3. Git Repository: Git repo is a hidden file named **.git**. It stores all the commits and compresses them. So when you need a specific commit it can present that to you.

In short :

A Git repository tracks and saves the history of all changes made to the files in a Git project. It saves this data in a directory called .git, also known as the repository folder.

Git uses a version control system to track all changes made to the project and save them in the repository. Users can then delete or copy existing repositories or create new ones for ongoing projects.



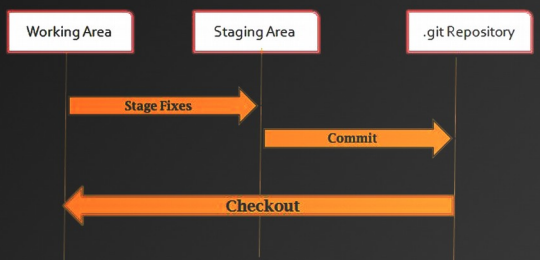
GitXam

Extra knowledge

1. Checking-out: Checking-out is the process of getting files from repository to your working copy. This is because you can only edit files when it is on your working copy. When you are done editing the file, you will save it back to the repository by commiting it.

2. Committing: Committing is the process of putting back the files from working copy to repository.

3. Add: The git add command adds new or changed files in your working directory to the Git staging area



GitXam

Tracking Git Project

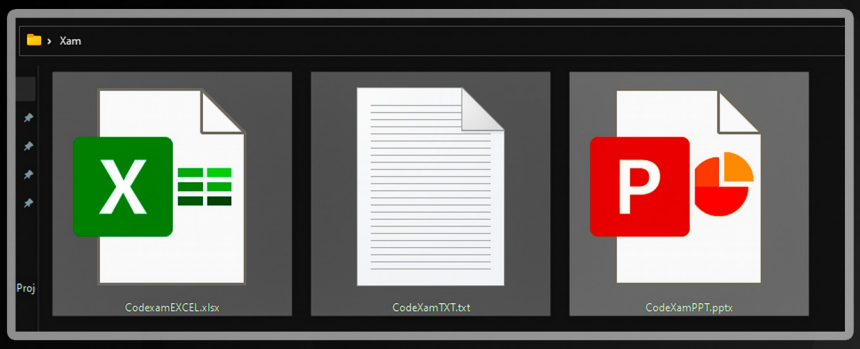
Experiment
Fail
Learn
Repeat

Topic:

Tracking Git Project

GitXam

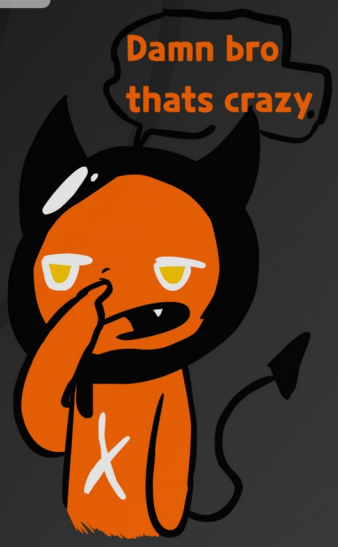
1. Let's say we have 3 files in a folder(Xam), for which we want to create a git repository



a. CodexamEXCEL.xlsx

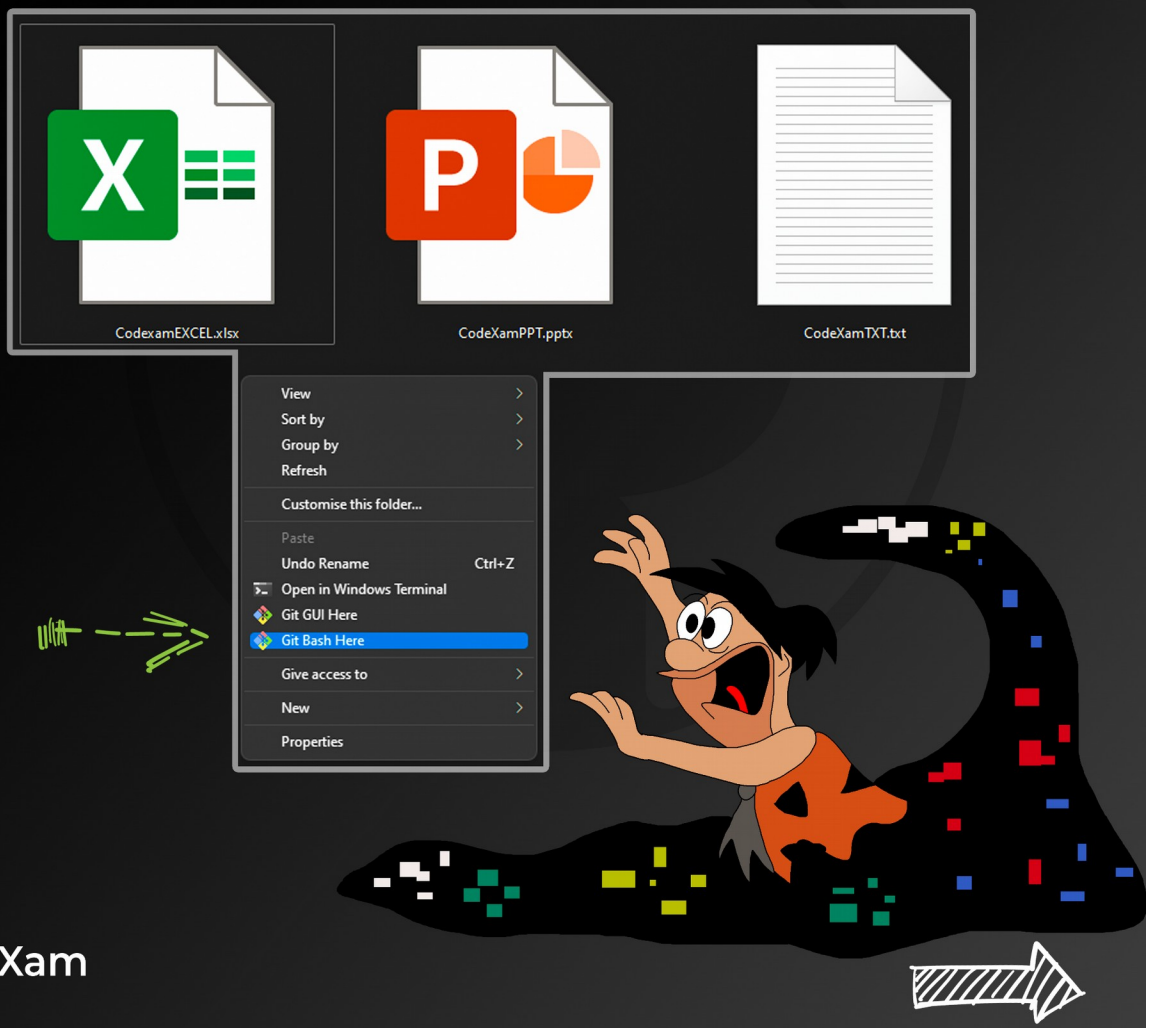
b. CodeXamTXT.txt

c. CodeXamPPT.pptx



GitXam

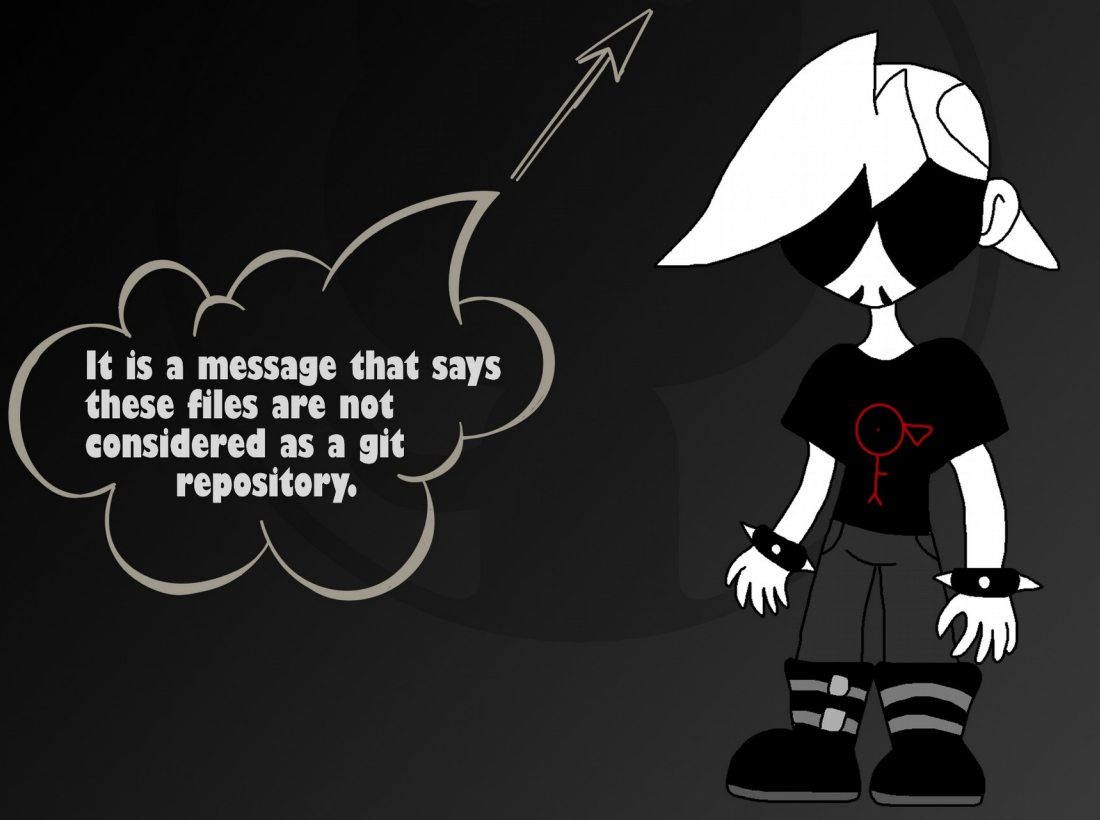
2. Open the folder, Right-Click anywhere while shifting and choose “Git Bash Here”.



GitXam

3. After Git bash is opened type “git status”.

It will give you a message that
“fatal: not a git repository (or any of the parent directories): .git”.

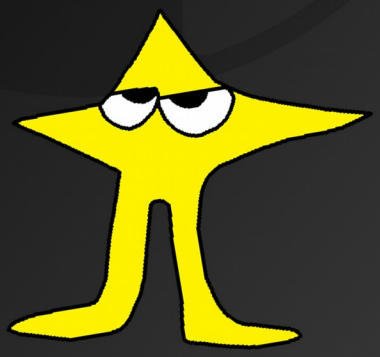


GitXam

4. Now we will type “**git init**” to initialize this folder as a repository.

Now if we type “**git status**” then we will get the files which are present in the folder.
Also, it says there that the files are not tracked.

```
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        CodeXamPPT.pptx
        CodeXamTXT.txt
        CodexamEXCEL.xlsx

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



GitXam

5. Now to track all the files inside this got repository we will type “git add --a”, which will add all these files to the staging area.

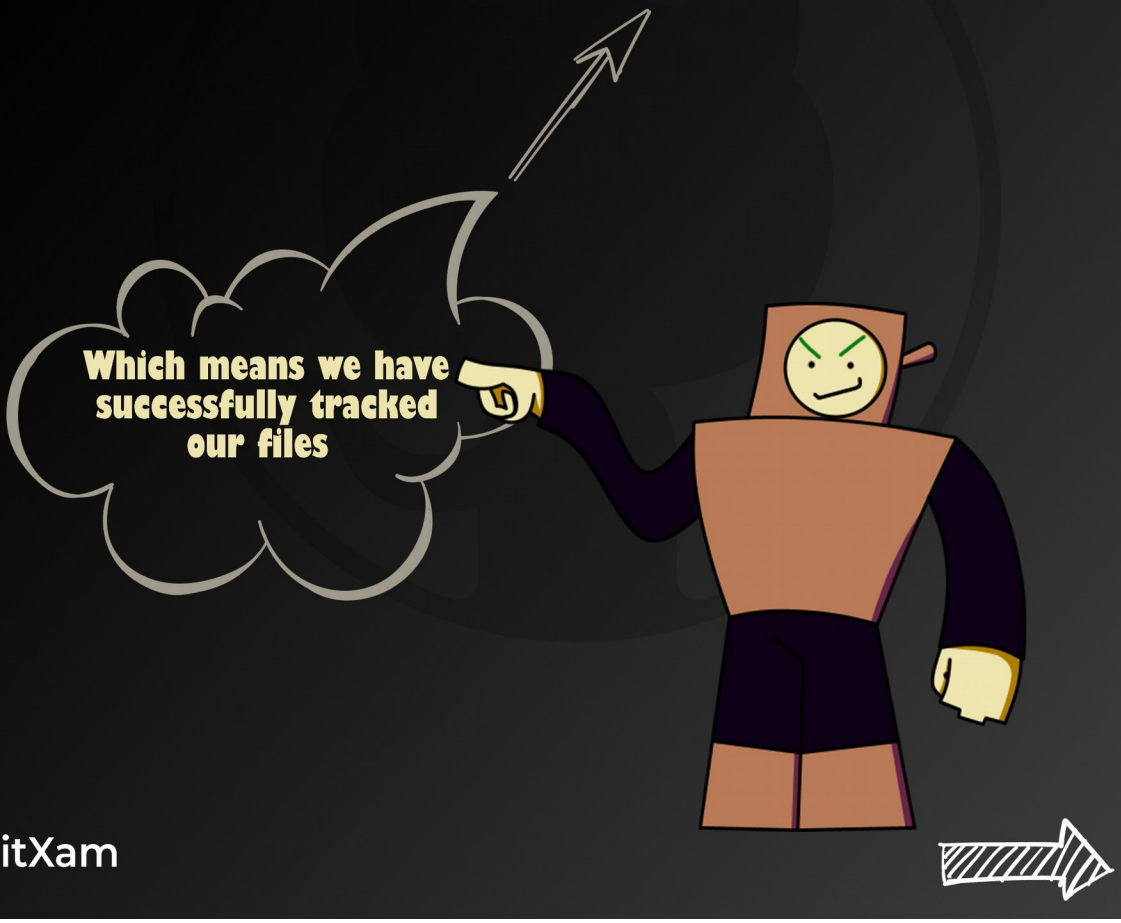
Now if we type “git status” then we will get the files which are present in the folder.
Also, it says Changes to be committed:

```
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   CodeXamPPT.pptx
        new file:   CodeXamTXT.txt
        new file:   CodexamEXCEL.xlsx
```

GitXam

6. Now we have to commit using this command " git commit -m "Initial Commit" ".

Now if we do " git status " again it will say
There's nothing to commit, working tree clean



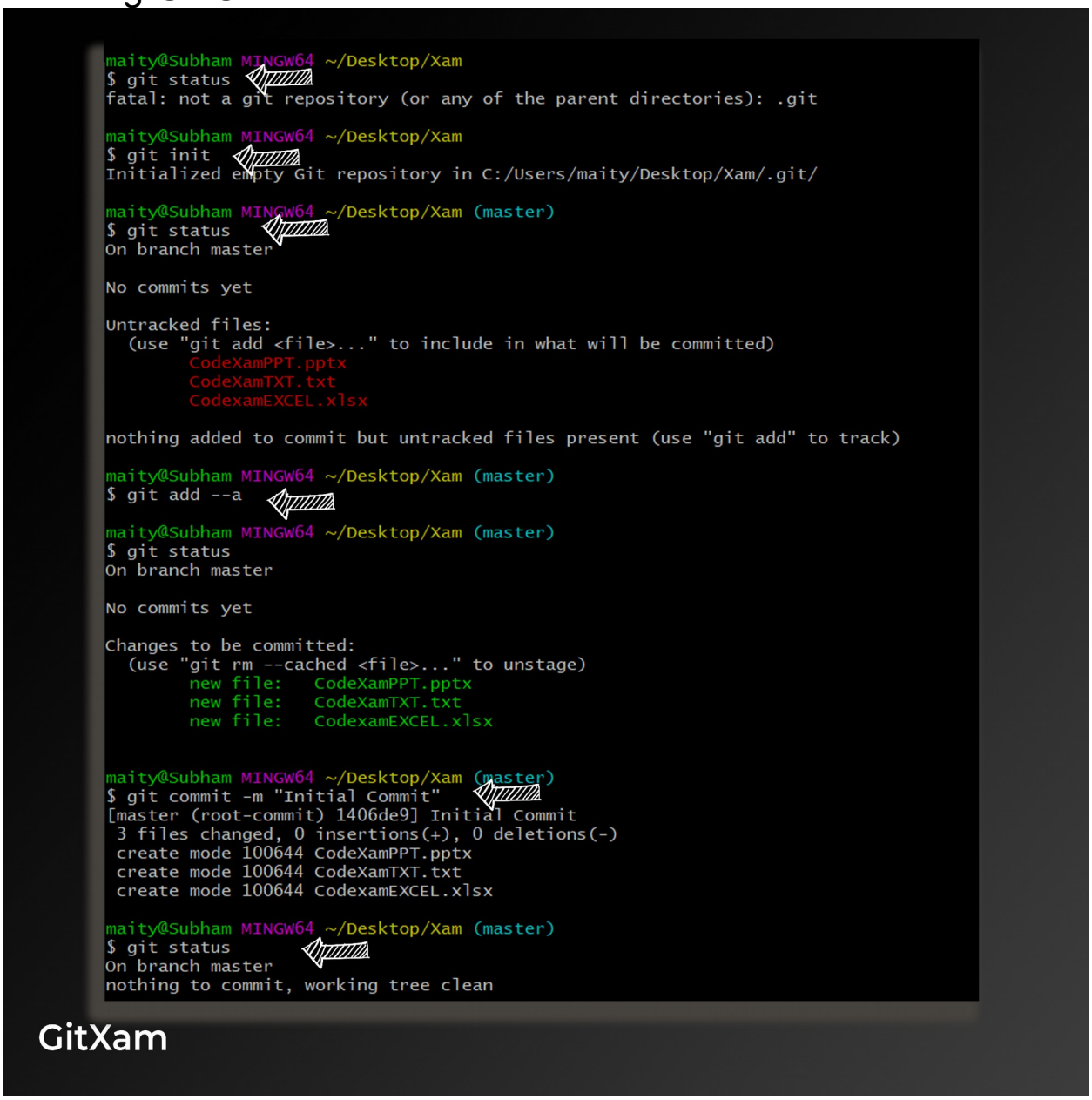
GitXam

7. Now to see the commits we have made, we will use “git log” command.

```
commit 1406de9fe2d13eb79594576bbf26261352a5a257 (HEAD -> master)
Author: gitxam <codexam.personal@gmail.com>
Date:   Sun Nov 7 23:36:13 2021 +0530
```

GitXam

Tracking Git Code



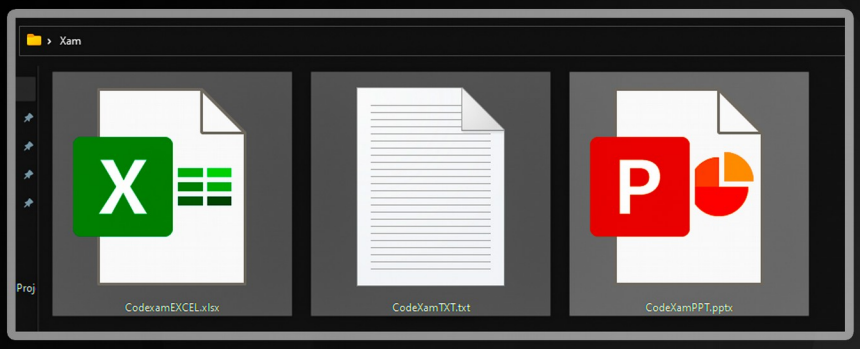
Git Log

```
maity@Subham MINGW64 ~/Desktop/Xam (master)
$ git log
commit 1406de9fe2d13eb79594576bbf26261352a5a257 (HEAD -> master)
Author: gitxam <codexam.personal@gmail.com>
Date:   Sun Nov 7 23:36:13 2021 +0530

    Initial Commit
```

GitXam

Modify And Tracking Git Project

We have 3 files in a folder(Xam), We saw in the last class(post) Right ?



a. CodexamEXCEL.xlsx

b. CodeXamTXT.txt

c. CodeXamPPT.pptx

GitXam

1. We are gonna modify something in CodeXamTXT.txt

What have I done here?

a. Open The Empty Text (CodeXamTXT.txt)

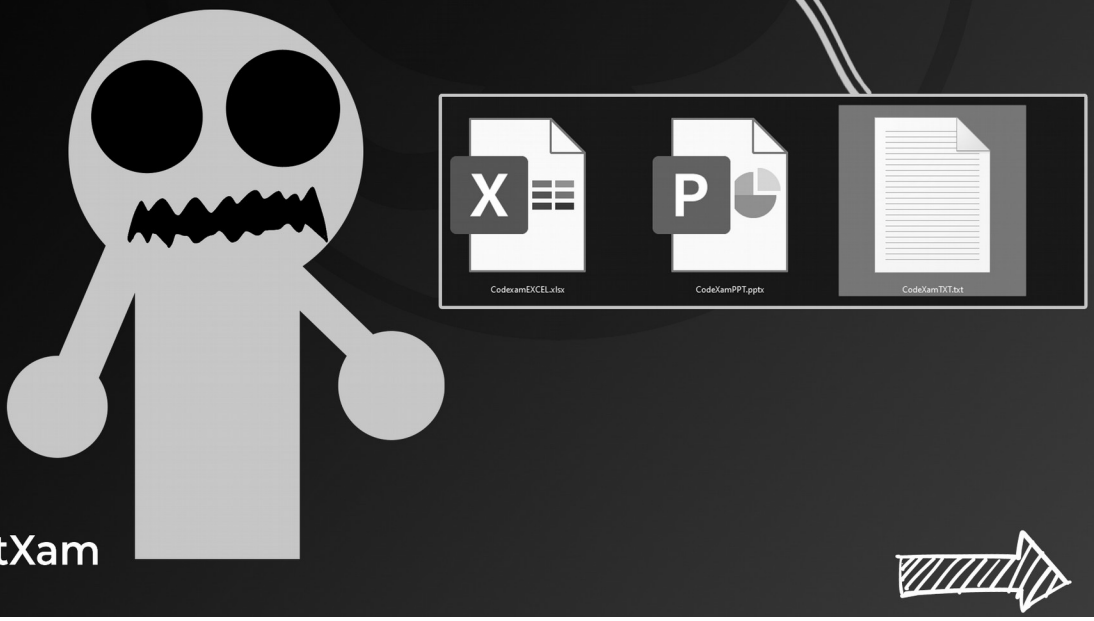
b. I wrote something like screenshot

c. Save that .

GitXam

2. Now if we type "git status", it says Changes not staged for commit:

```
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   CodeXamTXT.txt
```



GitXam

3. Now if we modify a single file then want to stage that file , we will use “git add file.ext” (Where file.ext is the filename and extension).

So which commands do I use for this?

“ git add CodeXamTXT.txt ”

GitXam

4. Now we will commit with a message by typing
" git commit -m "Your commit message" "

So which commands do
I use for this?

" git commit -m "Add CodeXamTXT Again" "

GitXam

5. **Done Now You have successfully tracked your files.**

Now if we type “git status” ,
it says nothing to commit, working tree clean

GitXam

Modify&TrackingCode

After Modify

```
maity@Subham MINGW64 ~/Desktop/Xam (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   CodeXamTXT.txt

no changes added to commit (use "git add" and/or "git commit -a")

maity@Subham MINGW64 ~/Desktop/Xam (master)
$ git add CodeXamTXT.txt

maity@Subham MINGW64 ~/Desktop/Xam (master)
$ git commit -m "Add CodeXamTXT Again"
[master 04de5a7] Add CodeXamTXT Again
 1 file changed, 2 insertions(+)

maity@Subham MINGW64 ~/Desktop/Xam (master)
$ git status
On branch master
nothing to commit, working tree clean
```

GitXam