19BIT0292
Bhaumik Tandan

DIGITAL ASSIGNMENT

OPEN SOURCE PROGRAMING

ITE1008

GIT

ABSTRACT

While creating a project, we come across different versions of it, and it a big hassle to manage and store all these version with different names and then remembering them. In this document we have talked about git, which is a open source distributed version control system (vcs), as well a source code management(scm) system. It keep tracks of all the versions of files of the project, as well as it also help us to role back to each version of it.

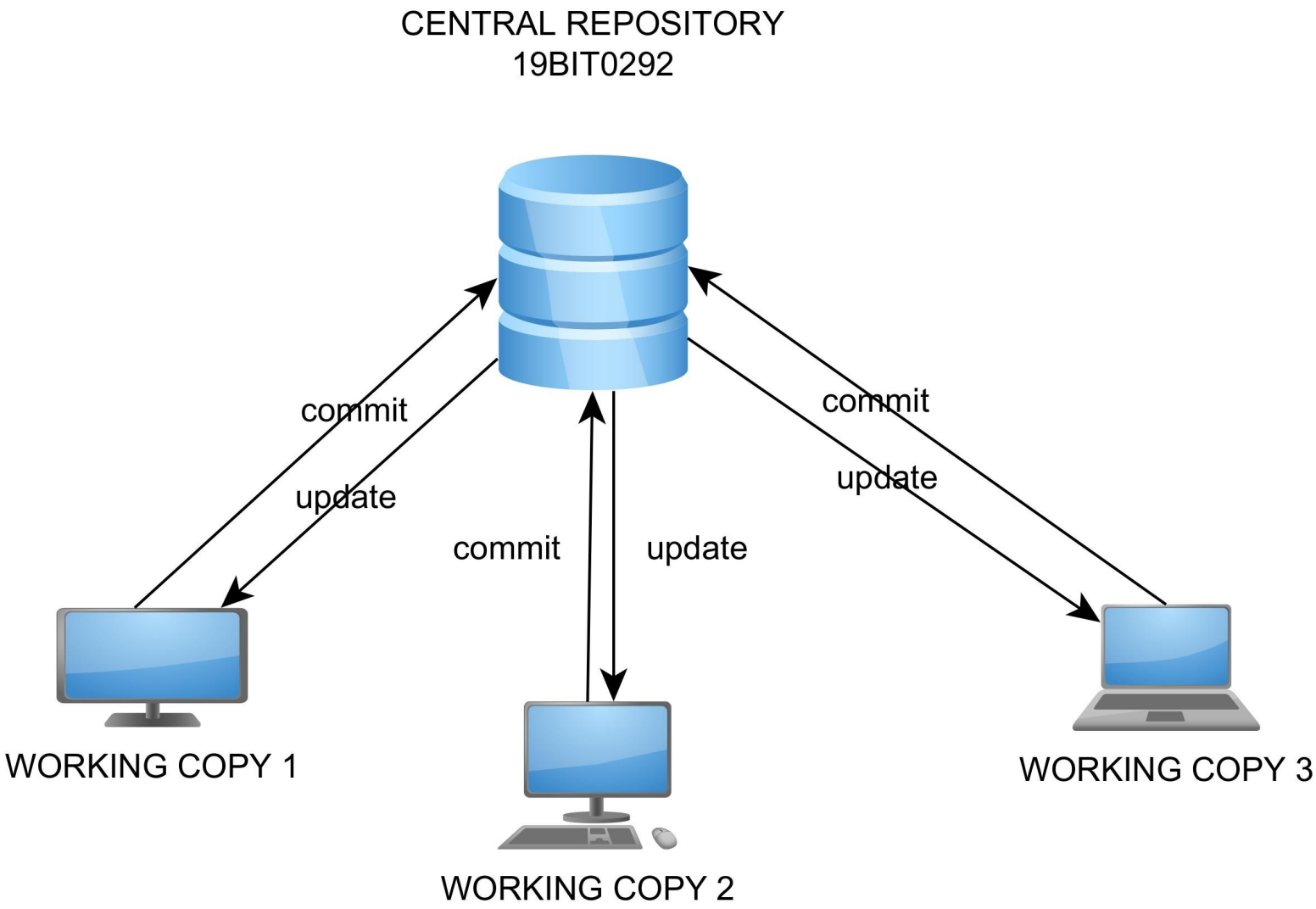
From 1991 to 2002 the updates in the development of Linux kernel were passed through archives and patches. Later in 2002 they began to use BitKeeper, which revoked it's free status in 2005. So there was a crisis in the collaborative development of Linux. Then the developer of Linux kernal, Linux Torvalds came with the idea of Git.

Later he launched git on 7th April, 2005, it revolutionized not only the development of linux but software development as a whole. It was release under (general public license)GPL's open source license. It is written in C so it also avoids runtime overheads which is a problem is case of other high-level languages. As now a days the storage space is abundant, it can not only be used by developers for source code management , but common users can also use it to keep all the version of their documents and file.

This document contains the commands of git, which are most frequently used. We have also demonstrated how to keep the backup of our project on GitHubb and bring it back to our system.

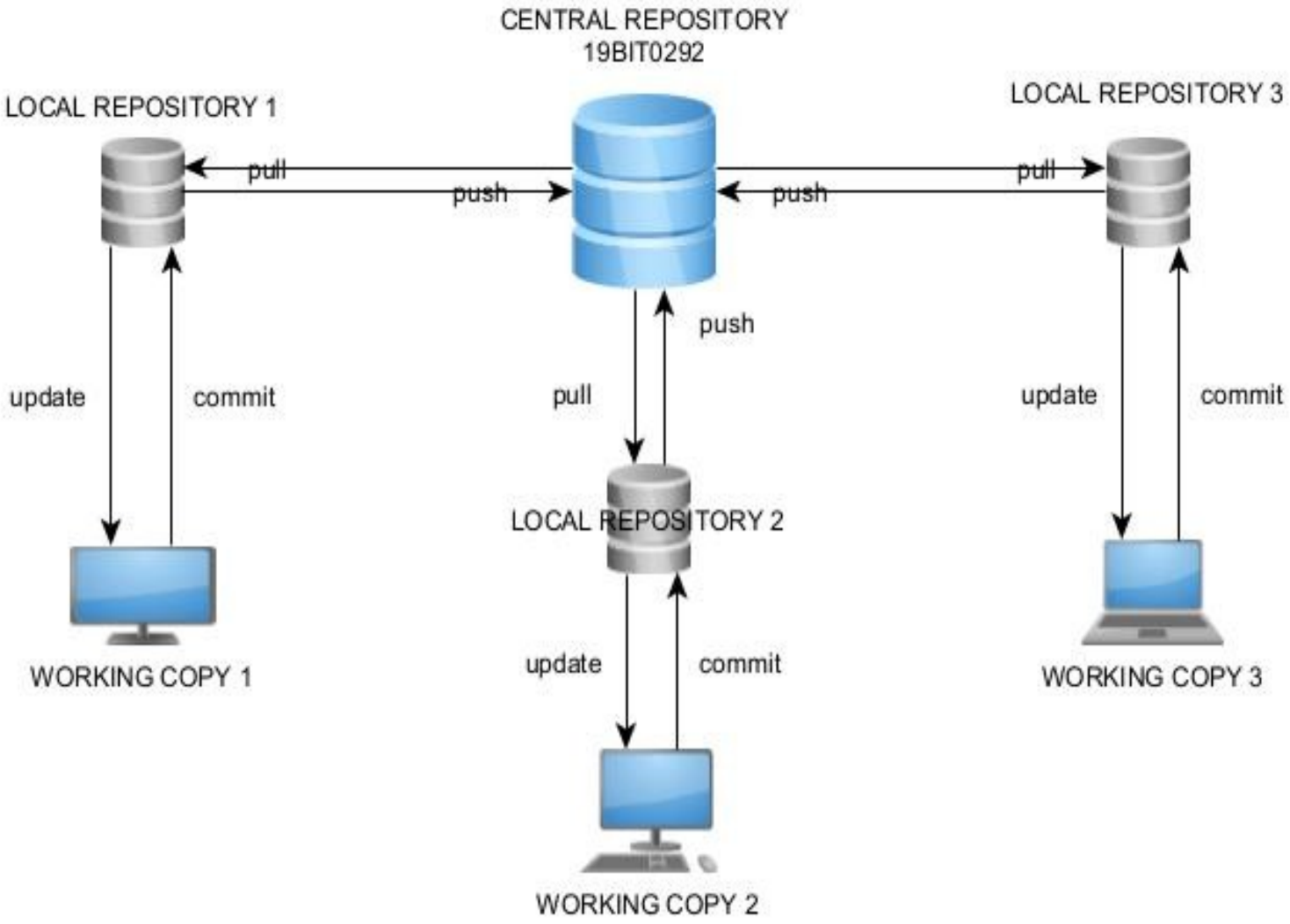
INTRODUCTION

All the changes and history of the folder is stored in a .git folder called repository. In a centralized version control system there is a central repository in a system and all the code contributors update their changes in the central server.



The biggest issue in this was single point of failure, i.e. in case of any fault in the central server the whole progress of the project is lost. But in case of distributed version control system like git there is a copy of the whole repository with each remote system and when needed they match their repository with a central repository hosted on the server.

There is no need for persist network connection between all the developers, and they can totally work offline. Whenever there is decentralization of any work, it makes the decision making quick. It also the case with distributed vcs as it increases the developer's productivity because it increases the executions of small tasks like commit, checkout, etc. very quick.



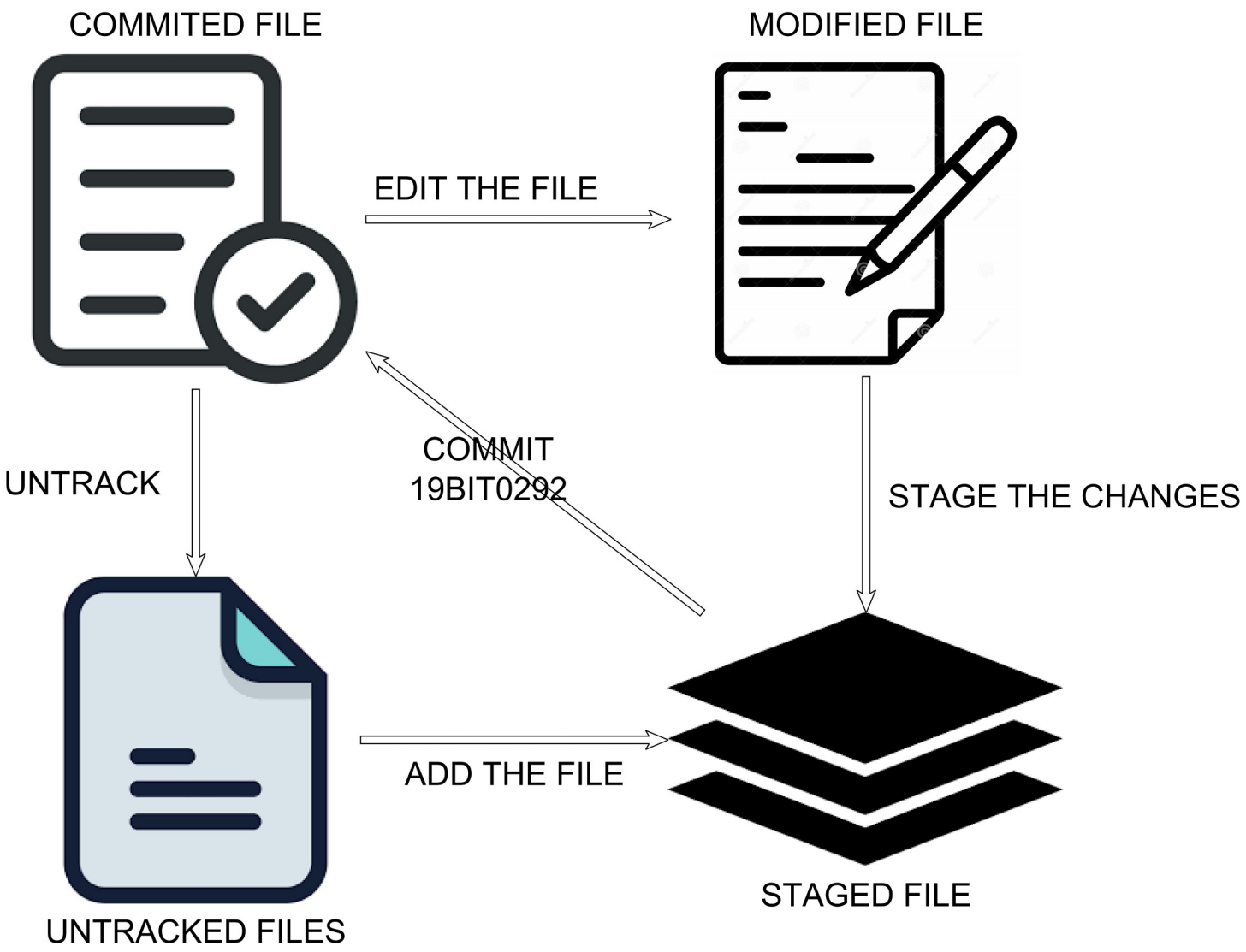
Since there are multiple copies of the repository locally with each contributor, the chances of losing data is very rare and also it gives a huge benefits in terms of speed, as most of the operations are performed locally.

In general when we work in our system we only have a single types of file.

But when we are using git, we can have four types of files:-

1) Untracked file:- A file in this stage does not get tracked and any changes made to it are not the part of our repository.

2) Modified file:- File in this stage are getting tracked but have been modified after the last commit.



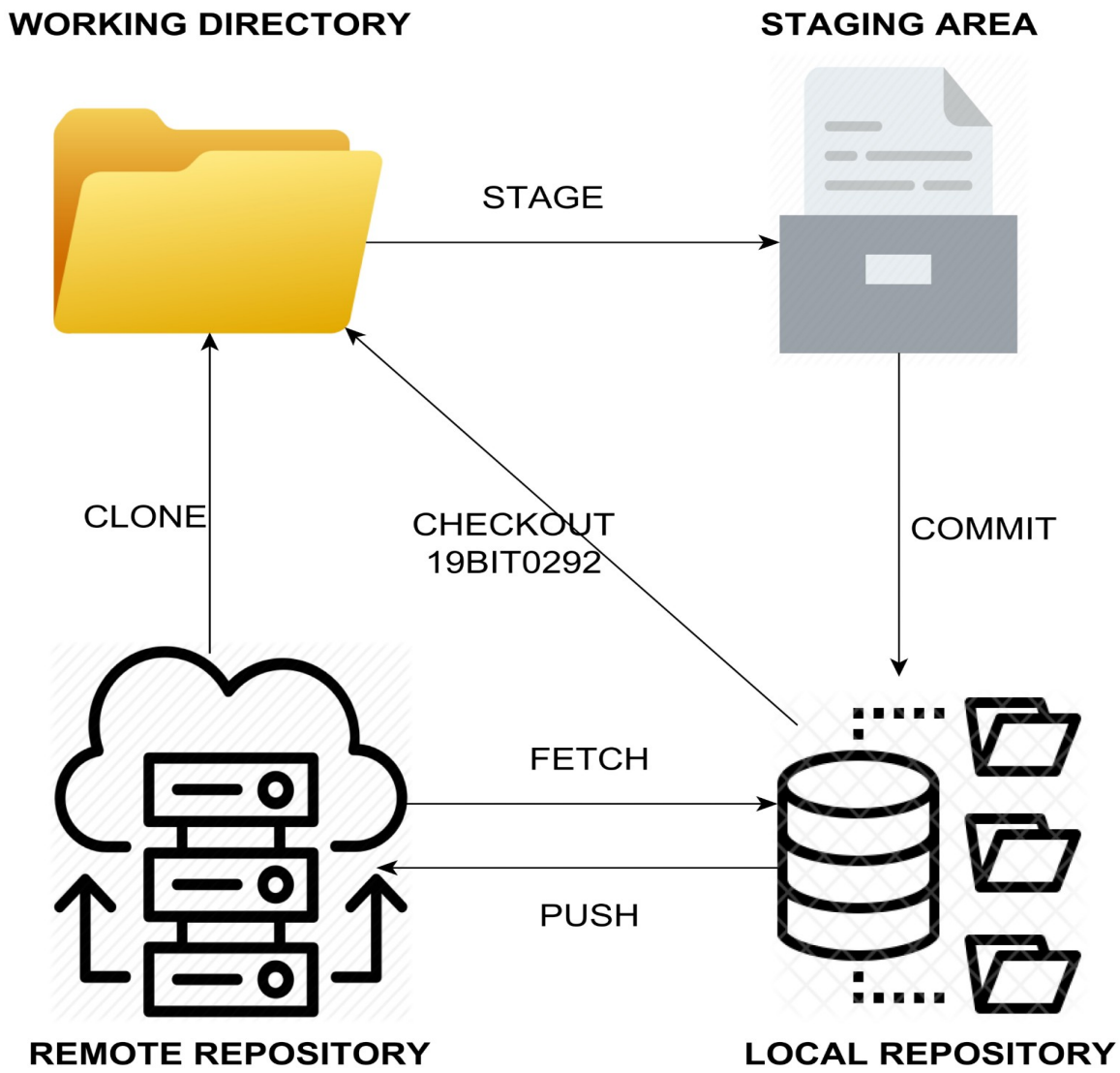
3) Committed file:- This file is up to date with our current repository.

4)Staged file:- These files are ready to get committed.

We have to note that backups are not automatically made in case of git we have to do it manually by running some commands to do so. With respect to git in general we can categories our work space into 4 parts:-

1)Working directory:- This is the normal working space in our computer.

2)Staging Area:- Files in this area are ready to get committed.



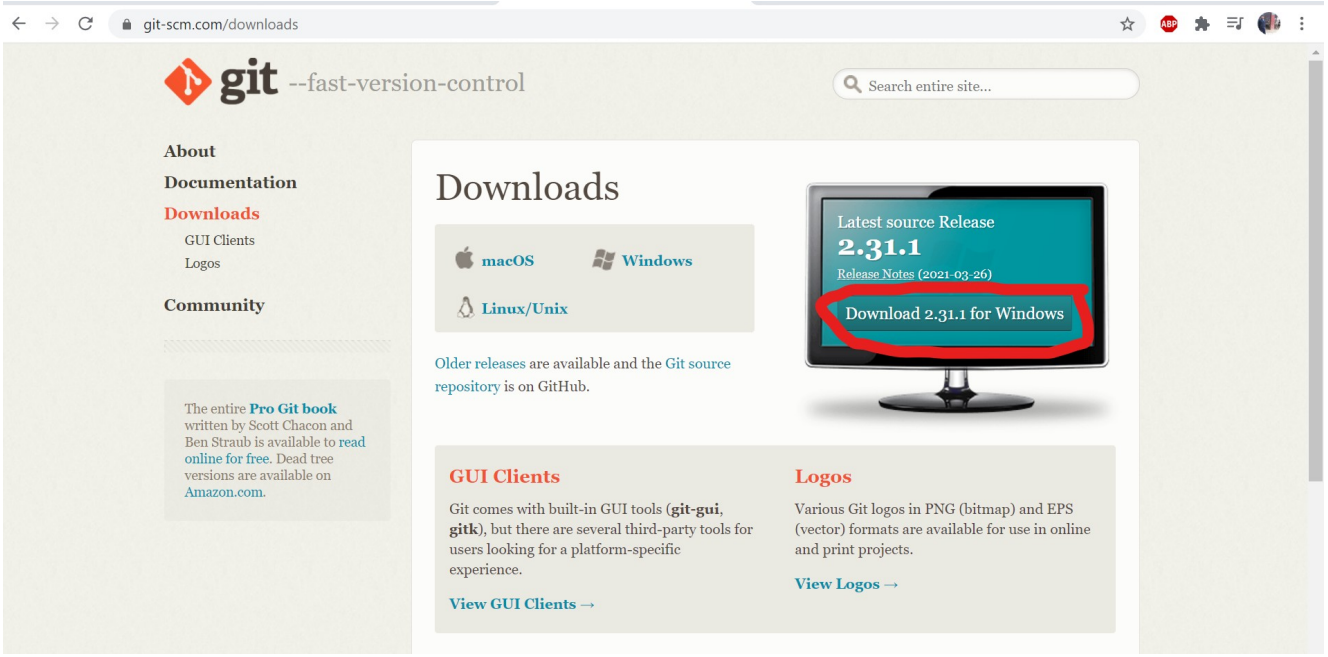
3)Local Repository:- Files in this area are committed and stored in a version of the project.

4)Remote repository:-Files in this area are in any kind of remote centralized server or repository hosting website like GitHub, GitLab, etc.

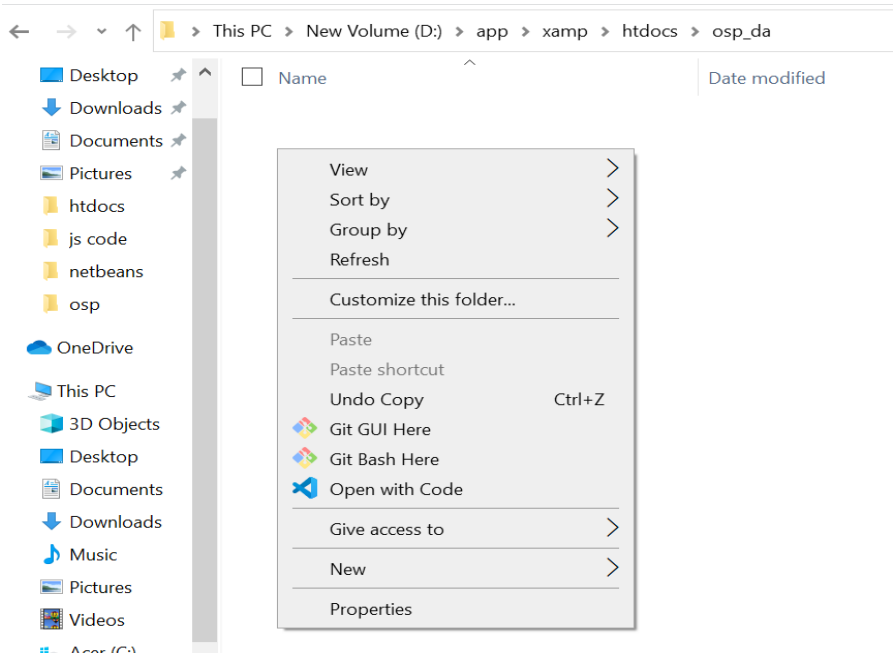
DEMONSTRATION STEPS

INSTALATION

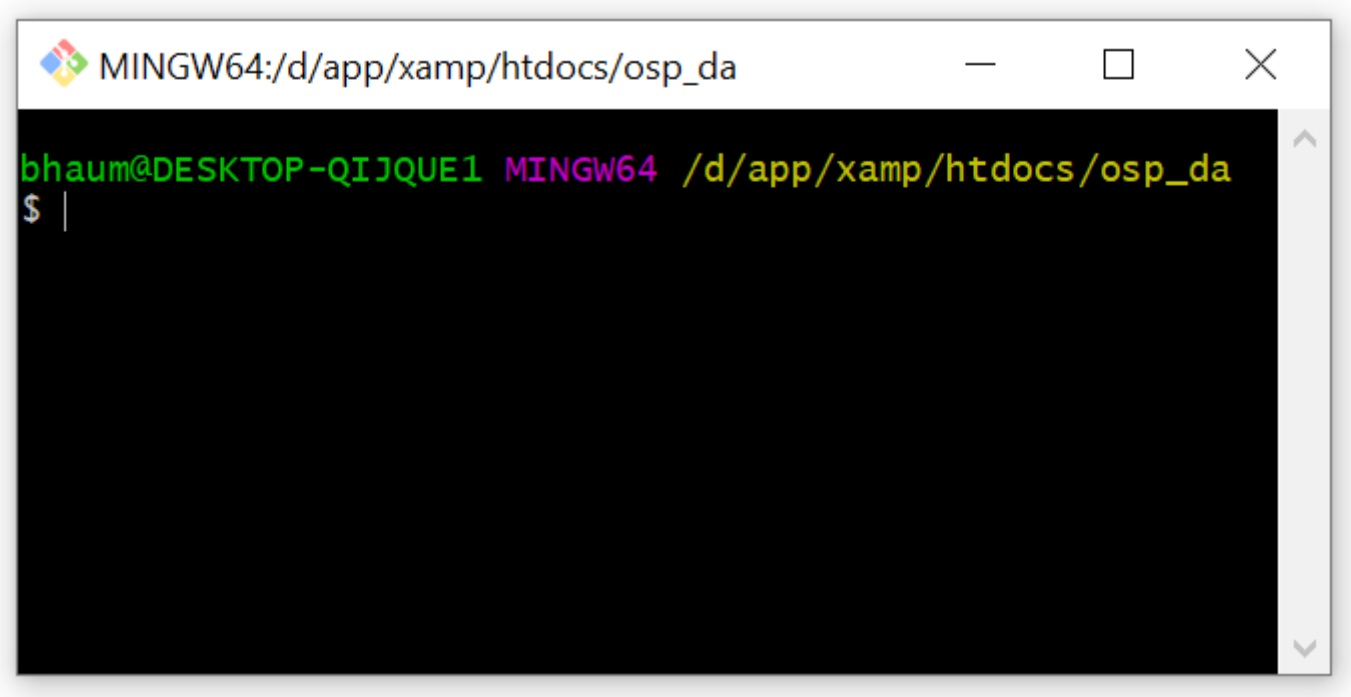
We can install git from here.



We can use any terminal in the system as well as we can also use GIT GUI, but in this demonstration we will use Git Bash. Open the folder and right click and select the option **"Git Bash Here"**



A terminal will open on that path as shown bellow

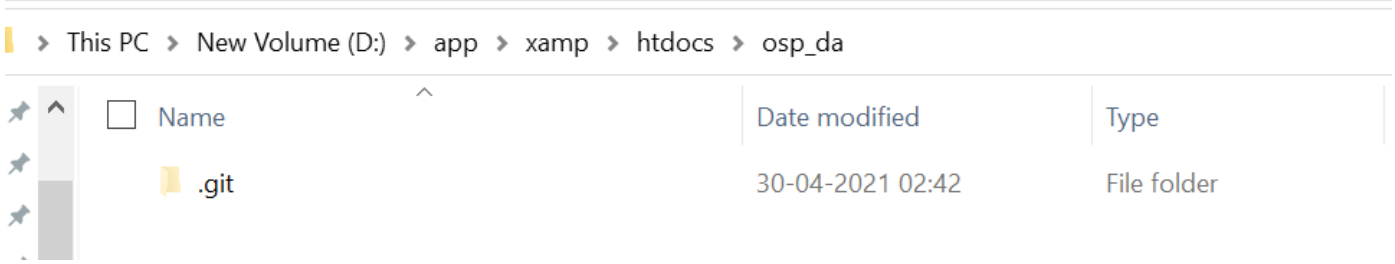


INITIATING A REPOSITORY

A new repository can be initiated in the folder by entering the command **`git init`**.

```
bhaum@DESKTOP-QIJQUE1 MINGW64 /d/app/xamp/htdocs/osp_da
$ git init
Initialized empty Git repository in D:/app/xamp/htdocs/osp_da/.git/
```

After entering the command we will observe that a new repository will be initiated in the folder and we will observe a folder labeled **“.git”** in the folder.

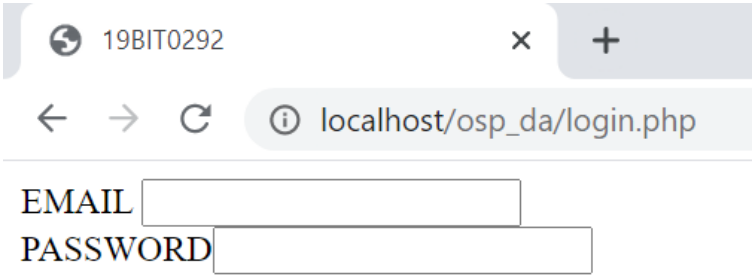


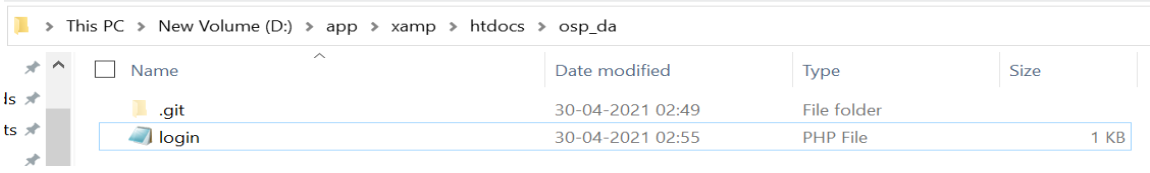
CREATING A PROJECT WITH GIT

We will create a login form, and with the help of which we will understand the use of git.

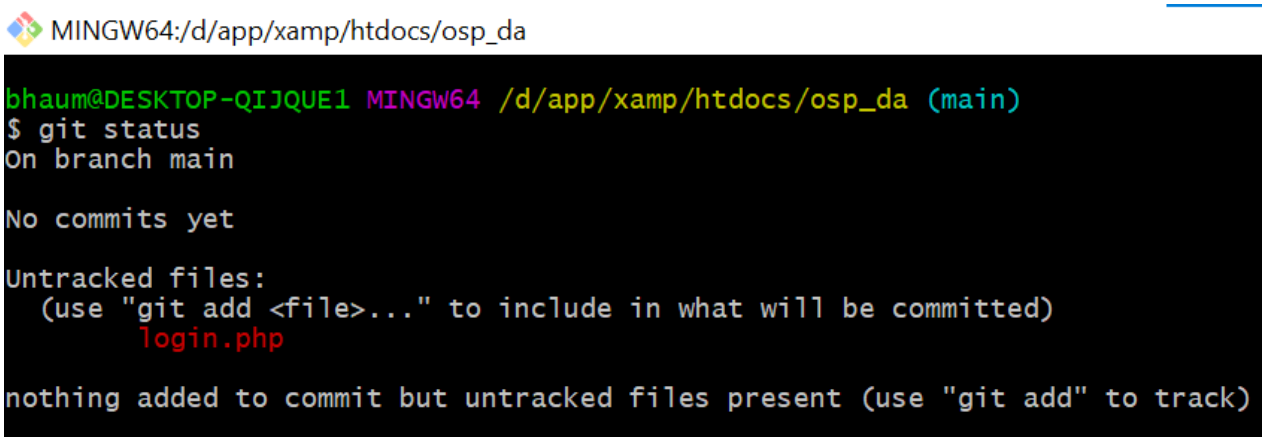
Suppose we have the following php file in our folder and we want to track changes using git here.

```
<html>
<head>
    <title>19BIT0292</title>
</head>
<body>
    <form>
        EMAIL
        <input type="text" name="em"><br>
        PASSWORD<input type="password">
    </form>
</body>
</html>
```





In order to know the current status of the repository we can use the command **`git status`**.

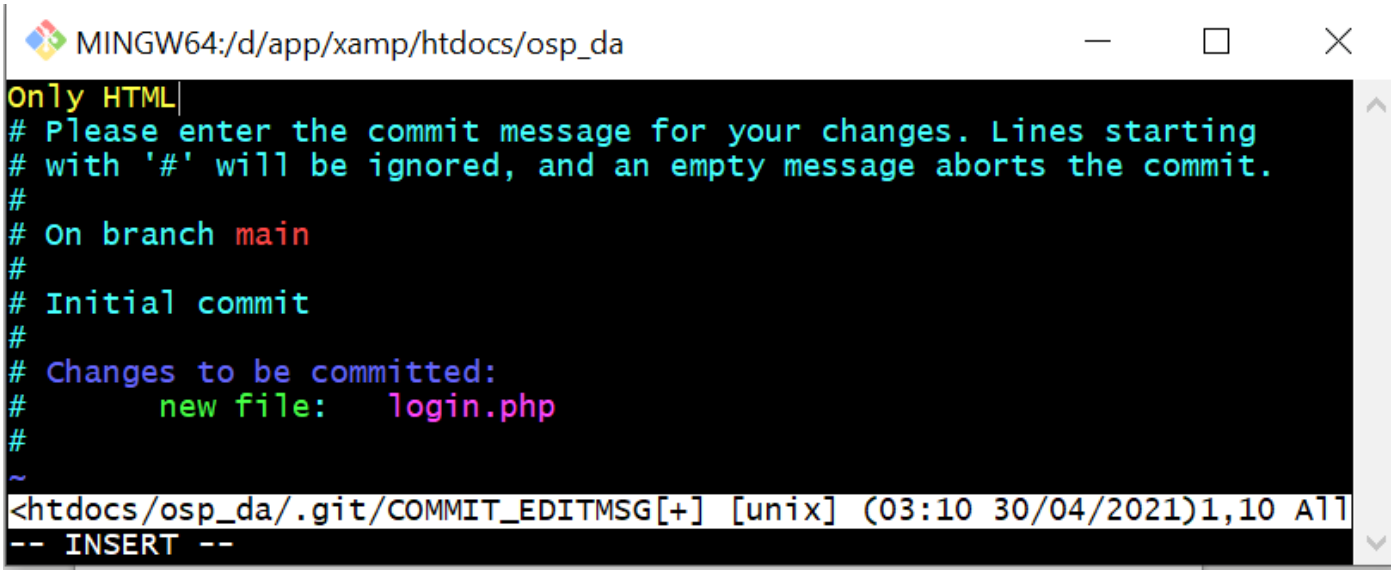


As

We can see in the image above git is telling us that we have an untracked file named login.php and in order to add it to the staging area we have to run the command **`git add login.php`**.

```
bhaum@DESKTOP-QIJQUE1 MINGW64 /d/app/xamp/htdocs/osp_da (main)
$  git add login.php

bhaum@DESKTOP-QIJQUE1 MINGW64 /d/app/xamp/htdocs/osp_da (main)
$ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   login.php
```

The file has been added to the staging area, we save the current version of the file using command **`git commit login.php`**.



On entering the command we will observe that vim editor gets open and we can give the name to our commit.

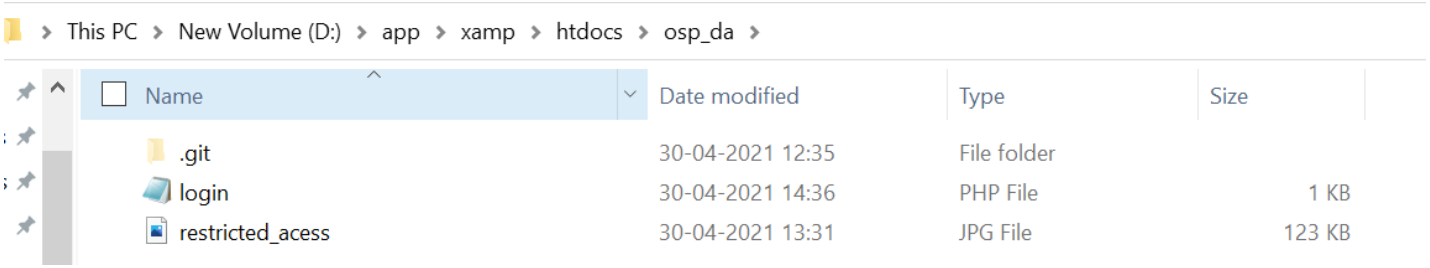
If we want to see the log of all the commits we can use the command **`git log`**.

MINGW64:/d/app/xamp/htdocs/osp_da

```
bhaum@DESKTOP-QIJQUE1 MINGW64 /d/app/xamp/htdocs/osp_da (main)
$ git commit login.php
[main (root-commit) d7ab085] Only HTML
 1 file changed, 12 insertions(+)
 create mode 100644 login.php

bhaum@DESKTOP-QIJQUE1 MINGW64 /d/app/xamp/htdocs/osp_da (main)
$ git log
commit d7ab0859d67182029142776e5d398f8764d810b9 (HEAD -> main)
Author: Bhaumik-Tandan <bhaumik.tandan@gmail.com>
Date:   Fri Apr 30 03:10:52 2021 +0530

    Only HTML

bhaum@DESKTOP-QIJQUE1 MINGW64 /d/app/xamp/htdocs/osp_da (main)
$
```

Our current version has been saved to the repository, now let's add some php to our script, display message if the email id and password matches our current password and also add an image to our login page.

After doing all the changes, the current status of our directory is:



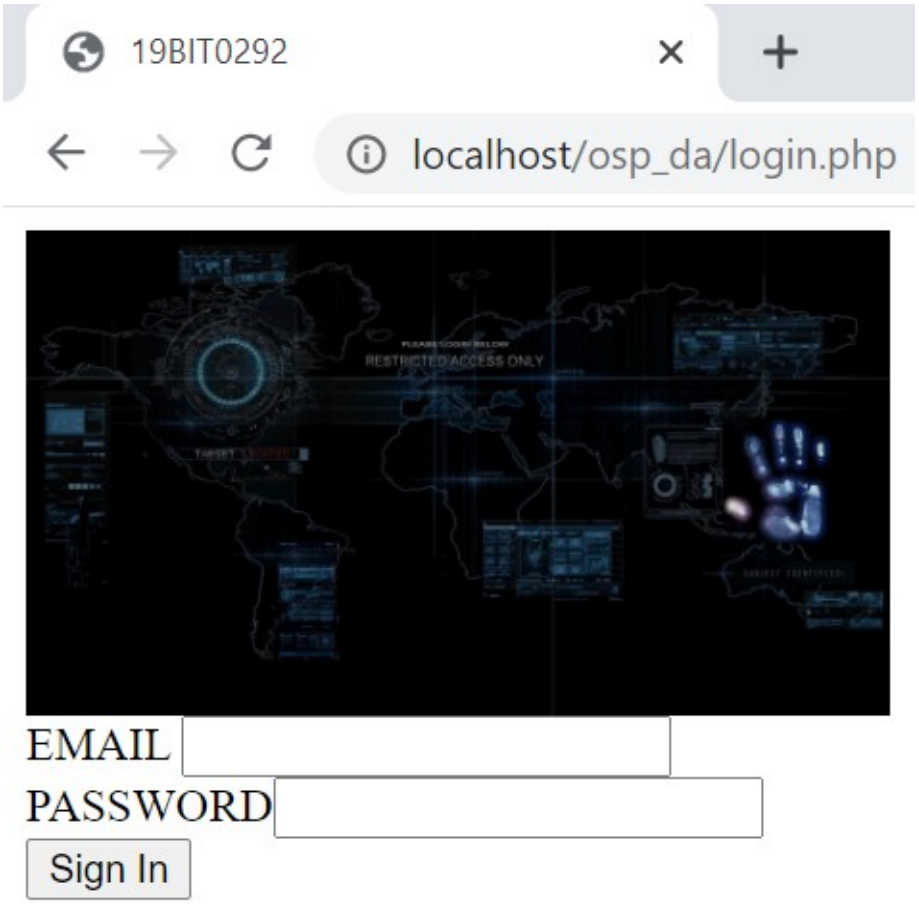
MINGW64:/d/app/xamp/htdocs/osp_da

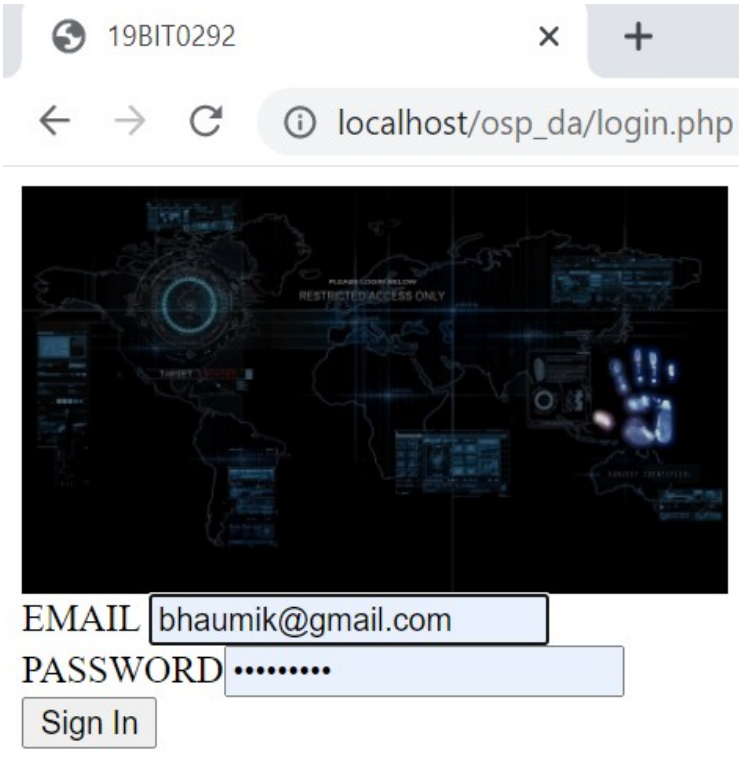
```
bhaum@DESKTOP-QIJQUE1 MINGW64 /d/app/xamp/htdocs/osp_da (main)
$ git status
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   login.php

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

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

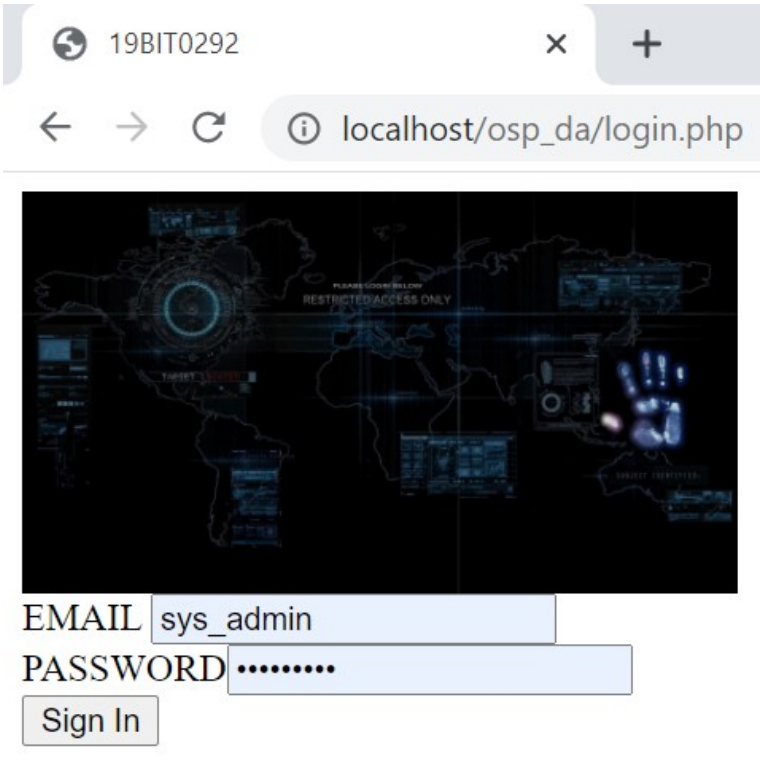
CODE IN login.php

```
<html>
<head>
    <title>19BIT0292</title>
</head>
<body>
<img src="restricted_acess.jpg" alt="restrcted acess" width="300">
    <form method="post">
        EMAIL
        <input type="text" name="em"><br>
        PASSWORD<input type="password" name="pas"><br>
        <button type="submit">Sign In</button>
    </form>
    <?php
         if ($_SERVER["REQUEST_METHOD"] == "POST")
         {
            $email = $_POST['em'];
            $password = $_POST['pas'];
            if ($email=="bhaumik.tandan@gmail.com" &&
$password=="19BIT0292")
            {
                echo "WELCOME";
            }
            else
                echo "WRONG CREDENTIALS";
         }
    ?>
</body>
</html>
```

OUTPUT


19BIT0292
localhost/osp_da/login.php
RESTRICTED ACCESS ONLY
EMAIL
PASSWORD
Sign In


19BIT0292
localhost/osp_da/login.php
RESTRICTED ACCESS ONLY
EMAIL bhaumik@gmail.com
PASSWORD
Sign In
WRONG CREDENTIALS


19BIT0292
localhost/osp_da/login.php
RESTRICTED ACCESS ONLY
EMAIL sys_admin
PASSWORD
Sign In
WELCOME

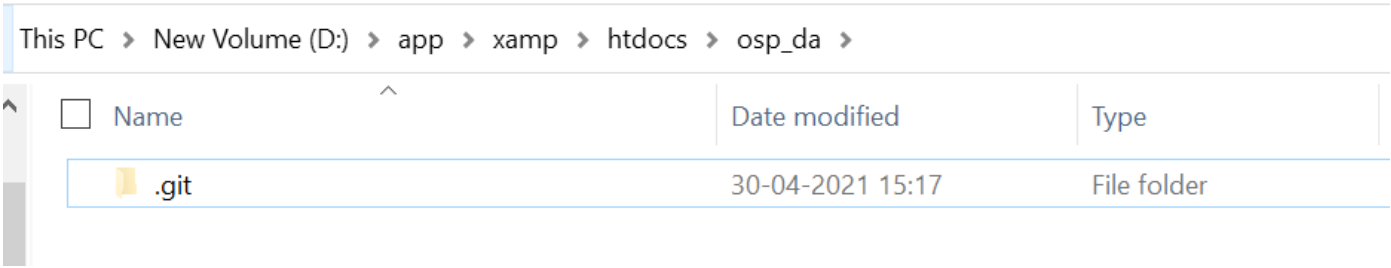
After doing all these changes in our directory, we have to first stage all the changes and then commit it, unlike last time we have multiple files, now inspite of staging the file one by one we will stage all the files by command **`git add .`** .

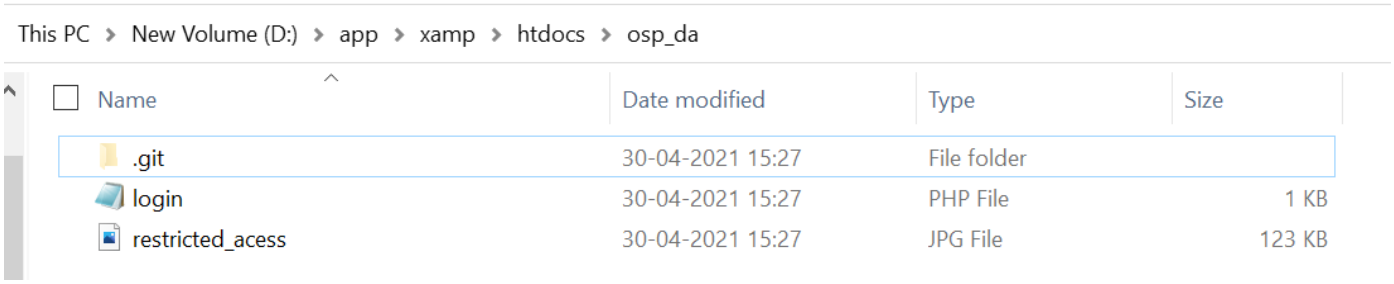
```
bhaum@DESKTOP-QIJQUE1 MINGW64 /d/app/xamp/htdocs/osp_da (main)
$ git add .

bhaum@DESKTOP-QIJQUE1 MINGW64 /d/app/xamp/htdocs/osp_da (main)
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   login.php
        new file:   restricted_acess.jpg
```

In place of writing “git commit” and then writing the message, we can directly give the name to the commit by writing the command **`git commit -m “[Message]”`**.

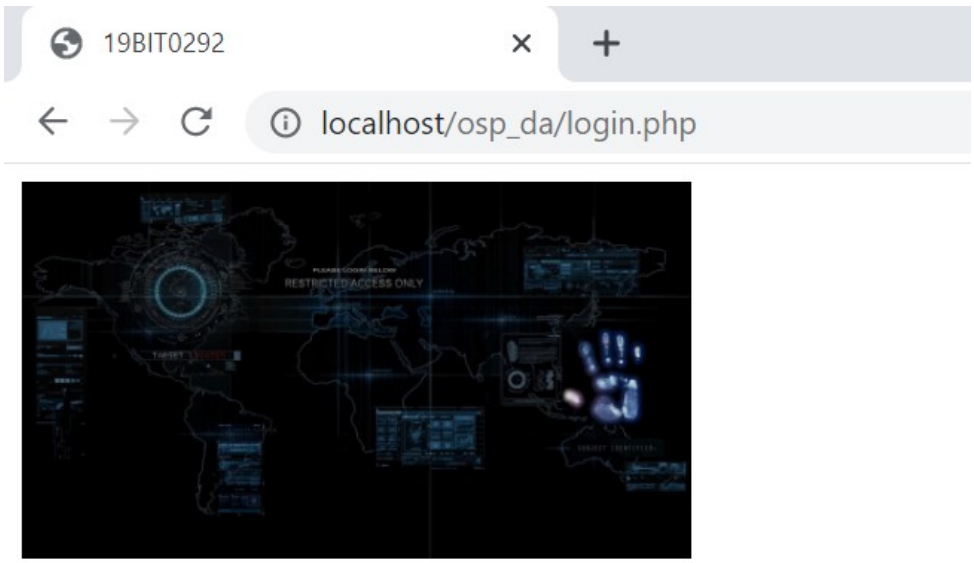
```
bhaum@DESKTOP-QIJQUE1 MINGW64 /d/app/xamp/htdocs/osp_da (main)
$ git commit -m "PHP and image added"
[main a983e91] PHP and image added
 2 files changed, 17 insertions(+), 2 deletions(-)
 create mode 100644 restricted_acess.jpg

bhaum@DESKTOP-QIJQUE1 MINGW64 /d/app/xamp/htdocs/osp_da (main)
$ git log
commit a983e912d314915184b5da3dc111431f3f01a2d9 (HEAD -> main)
Author: Bhaumik-Tandan <bhaumik.tandan@gmail.com>
Date:   Fri Apr 30 15:02:08 2021 +0530

    PHP and image added

commit d7ab0859d67182029142776e5d398f8764d810b9
Author: Bhaumik-Tandan <bhaumik.tandan@gmail.com>
Date:   Fri Apr 30 03:10:52 2021 +0530

    Only HTML
```

If suppose our files got deleted by mistake then we can match out current working directory and thus all our files will come back by using command **`git checkout -f`**.



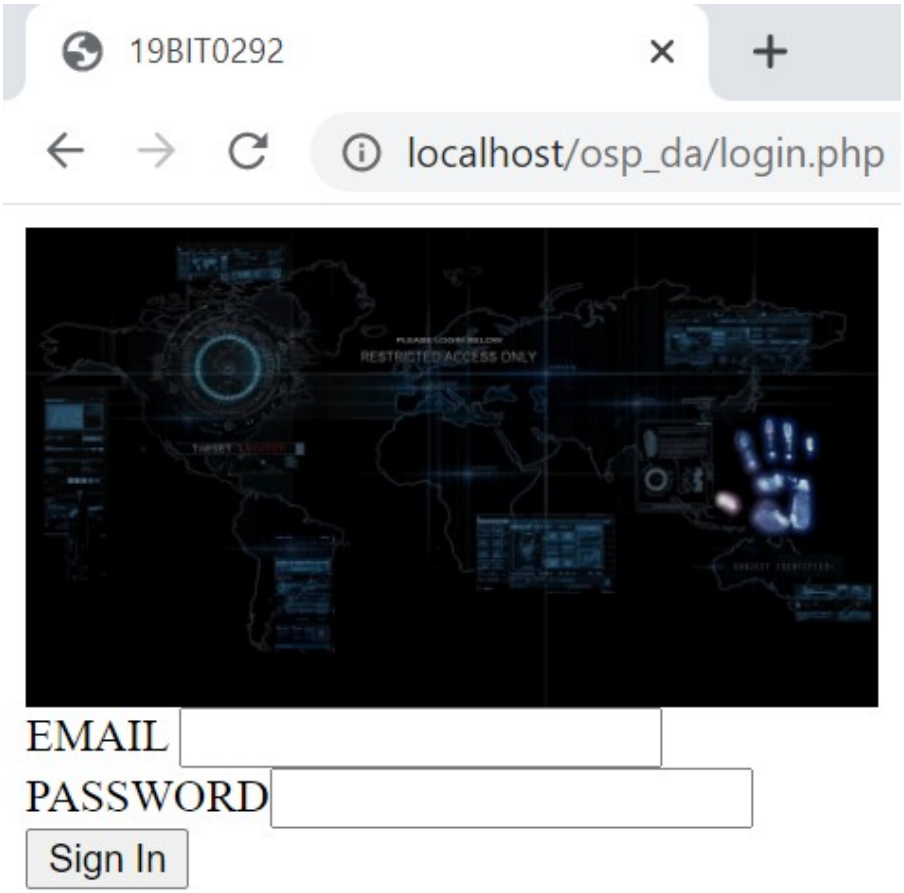


If we want to match only a particular file with the last commit we can also use the command **`git checkout [file name]`**.



Suppose by mistake (as shown above) we deleted the form in "login.php", we can go back to the last stage by command **`git checkout login.php`**.

```
bhaum@DESKTOP-QIJQUE1 MINGW64 /d/app/xamp/htdocs/osp_da (main)
$ git checkout login.php
Updated 1 path from the index
```

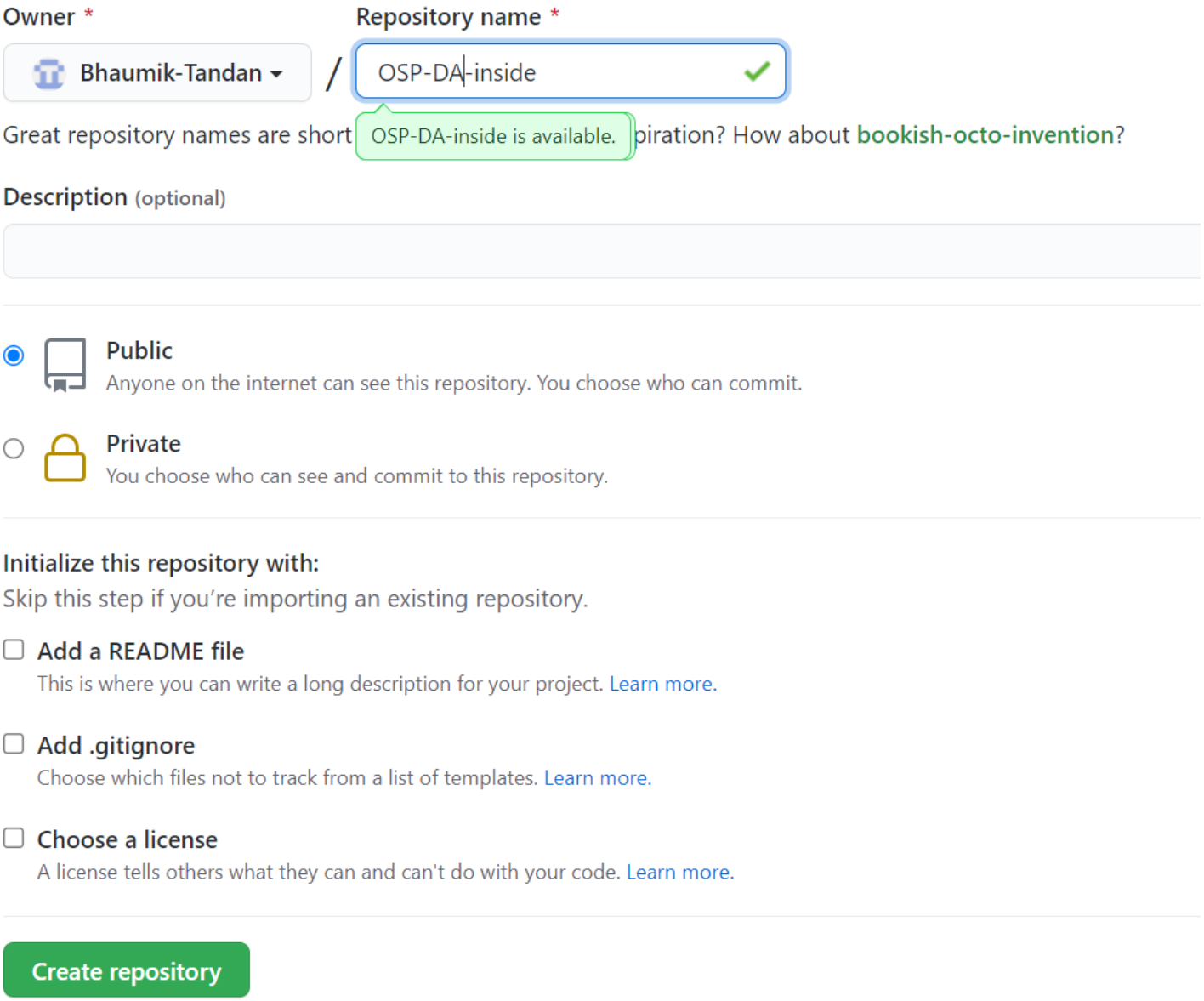


We can unstage our files by writing the command **`git restore --staged [file_name]`**.

```
bhaum@DESKTOP-QIJQUE1 MINGW64 /d/app/xamp/htdocs/osp_da (main)
$ git add a.txt

bhaum@DESKTOP-QIJQUE1 MINGW64 /d/app/xamp/htdocs/osp_da (main)
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   a.txt


bhaum@DESKTOP-QIJQUE1 MINGW64 /d/app/xamp/htdocs/osp_da (main)
$ git restore --staged a.txt

bhaum@DESKTOP-QIJQUE1 MINGW64 /d/app/xamp/htdocs/osp_da (main)
$ git status
On branch main
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        a.txt

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

PUSHING TO REMOTE REPOSITORY

We already have repository, now lets push it on any cloud , we will use GitHubb for the same.

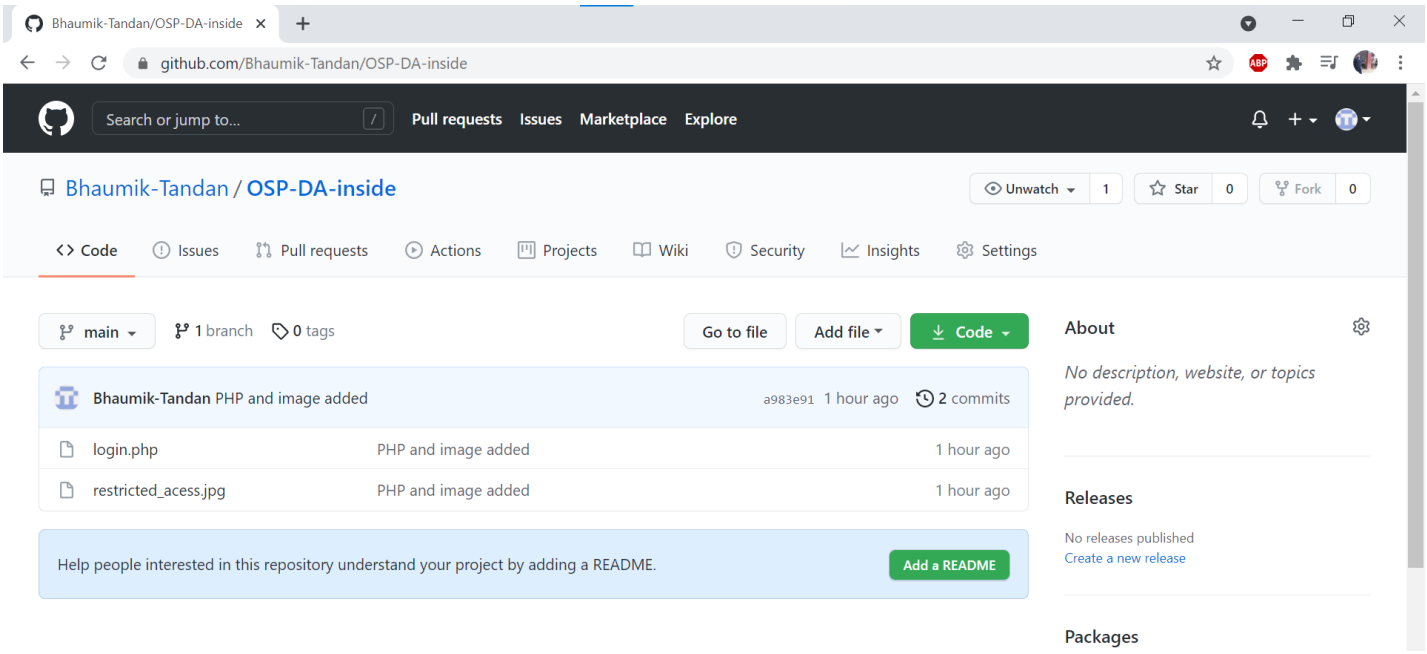


After creating a repository we will get a link like here we git:-
https://github.com/Bhaumik-Tandan/OSP-DA-inside.git

We can push the repository to the above link by writing the command **`git push -u main [link]`**.

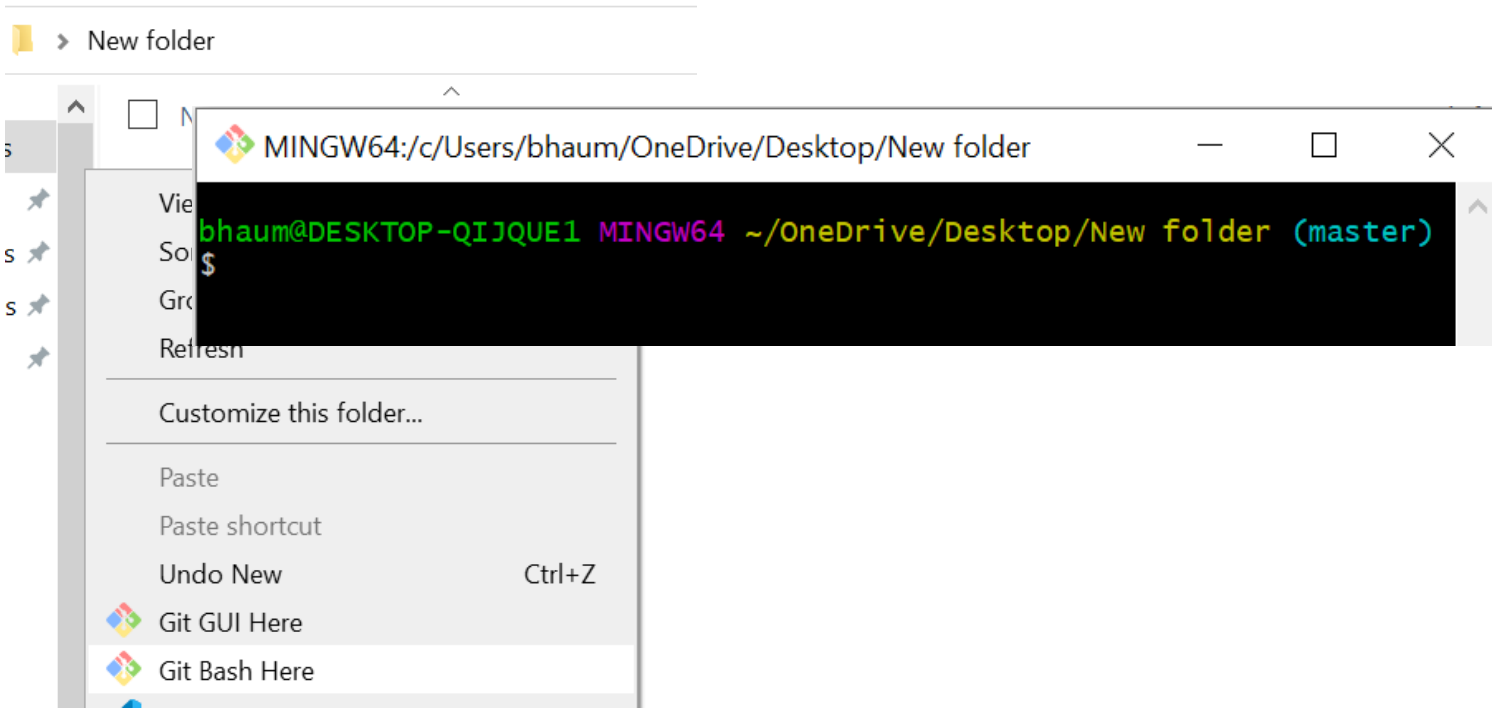
Note that we have used **`-u`** because if we want to again update the cloud repository so we just have to write **`git push`**.

```
bhaum@DESKTOP-QIJQUE1 MINGW64 /d/app/xamp/htdocs/osp_da (main)
$ git push -u https://github.com/Bhaumik-Tandan/OSP-DA-inside.git main
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (7/7), 115.14 KiB | 4.11 MiB/s, done.
Total 7 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/Bhaumik-Tandan/OSP-DA-inside.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'https://github.com/Bhaumik-Tandan/OSP-DA-inside.git'.
```



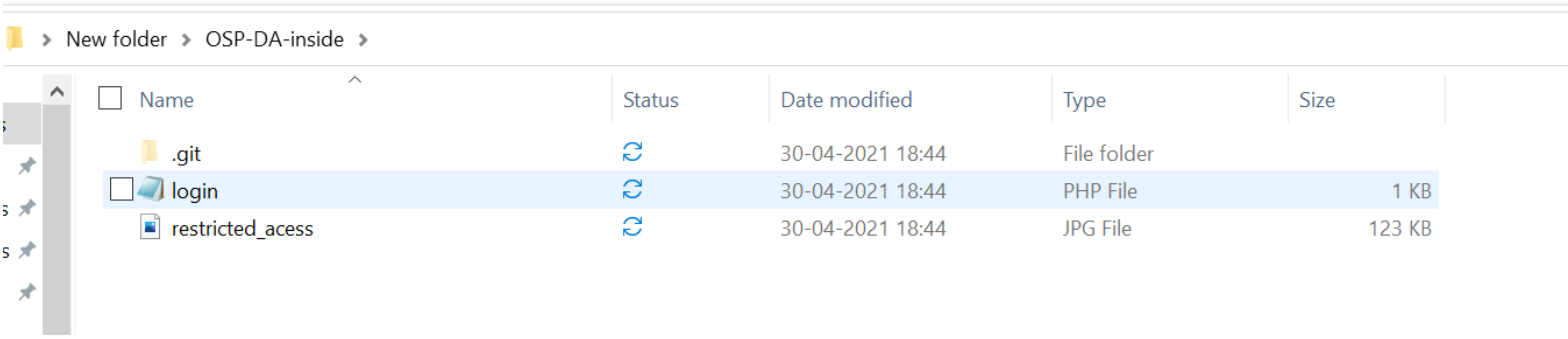
CLONING CLOUD REPOSITORY

If we delete our project from our computer and now we want it back in a folder in our system so we easily do it by opening git bash in the folder and entering the command **`git clone [link]`**.



MINGW64:/c/Users/bhaum/OneDrive/Desktop/New folder

```
bhaum@DESKTOP-QIJQUE1 MINGW64 ~/OneDrive/Desktop/New folder (master)
$ git clone https://github.com/Bhaumik-Tandan/OSP-DA-inside
Cloning into 'OSP-DA-inside'...
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 7 (delta 1), reused 7 (delta 1), pack-reused 0
Receiving objects: 100% (7/7), 115.14 KiB | 1.11 MiB/s, done.
Resolving deltas: 100% (1/1), done.
```



We have all our files in our newly created folder, it not only has our files but also our whole repository if we run the the command **`git log`**, here so we can see out commit history.

MINGW64:/c/Users/bhaum/OneDrive/Desktop/New folder/OSP-DA-inside

```
bhaum@DESKTOP-QIJQUE1 MINGW64 ~/OneDrive/Desktop/New folder/OSP-DA-inside (main)
$ git log
commit a983e912d314915184b5da3dc111431f3f01a2d9 (HEAD -> main, origin/main, origin/HEAD)
Author: Bhaumik-Tandan <bhaumik.tandan@gmail.com>
Date:   Fri Apr 30 15:02:08 2021 +0530

    PHP and image added

commit d7ab0859d67182029142776e5d398f8764d810b9
Author: Bhaumik-Tandan <bhaumik.tandan@gmail.com>
Date:   Fri Apr 30 03:10:52 2021 +0530

    Only HTML
```

CONCLUSION

Git is a very powerful tool for source code management, it makes really easy for a developer to keep the backup of all the versions of their code. But due to it's sophisticated nature, it's users are only limited to developers and technical professionals.

Nowadays when everyone is working on computer, making documents folders etc. is a very common activity and git can assist them to maintain and to oversee the progress of their directory .

But a common user does not use git because adding files to staging area then to the repository, then rolling back the files is a very complex process which a non-technical individual cannot understand. Although many code editors and integrated development environments offer graphic user interface for git, but still, we have to do everything manually.

If we would have the facility of the automatically tracking and committing our files by detecting the current progress, then it will not be difficult for even a non technical professional to use it.

By making it more user friendly and less sophisticated, git can be integrated in softwares which are used by people in daily lives, like MS Word, Libreoffice, 3D development tools, etc. So that even general public can also use it.

Although distributed version control system makes it easy for locally develop the software, but it is very tedious to every time match the local repository with cloud repository, but if git would have had the feature for persist connection of local and cloud repository and automatically updating the cloud repository, by making changes to local repository, it would been more easier to use.