Git in Anaconda Platform

The Academician

Git is a VCS (Version Control System) created in 2005 by Linus Torvalds who started the Linux kernel.

- Git is a free open-source software available for installation on Unix based platforms, Windows and macOS.

- Git is one of the most popular version control systems and it is used in millions of projects.

- *Git has a distributed architecture. This means that every person contributing to a repository has full copy of the repository on their own development machines*.

- Git doesn't rely on any kind of centralized server to provide control organizations to its workflow. *Git can work as a standalone program as a server and as a client. This means that you can use Git on a single machine without even having a network connection.*

- You can use it as a server on a machine where you want to host your repository. And then you can use Git as a client to access the repository from another machine or even the same one.

+

Open Anaconda powershell Prompt, and Spyder

(0) To install git in Anaconda

```
conda install -c anaconda git
```

(1) Check using command:

```
git version
```

```
git version 2.36.1.windows.1
```

Create folder in c drive called **academic**
Create a file inside academic called file.py ()

(2) Configure git

```
git config --global user.email theacademician2021@gmail.com
git config --global user.name TheAcademician
```

(3) Initialize git

```
(base) C:\academic>git init
```

```
Initialized empty Git repository in C:/academic/.git/
```

(4) Add file to track

```
(base) C:\academic>git add file.py
```

(5) Display status of the git

```
(base) C:\academic>git status
```

```
On branch master
No commits yet
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   file1.py
```

(6) Add one more file myFile2.py and stored it into academic folder

```
(base) C:\academic>git status
On branch master
No commits yet
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   file1.py
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        myFile2.py
```

(7) Add all the files for tracking

```
(base) C:\academic>git add *
```

```
(base) C:\academic>git status
On branch master
No commits yet
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   file1.py
        new file:   myFile2.py
```

```
(base) C:\academic>dir .git
 Volume in drive C is Windows
 Volume Serial Number is AE4F-4E27
 Directory of C:\academic\.git
27-06-2022  15:42               130 config
27-06-2022  15:42                73 description
27-06-2022  15:42                23 HEAD
27-06-2022  15:42    <DIR>          hooks
27-06-2022  15:43               184 index
27-06-2022  15:42    <DIR>          info
27-06-2022  15:43    <DIR>          objects
27-06-2022  15:42    <DIR>          refs
               4 File(s)            410 bytes
               4 Dir(s)  220,631,670,784 bytes free
```

(8) Commit with message

```
(base) C:\academic>git commit -m "files are commited"
[master (root-commit) a3acbfc] files are commited
 2 files changed, 13 insertions(+)
 create mode 100644 file1.py
 create mode 100644 myFile2.py
```

```
(base) C:\academic>git status
On branch master
nothing to commit, working tree clean
```

Change few lines in file.py

```
(base) C:\academic>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:   file1.py

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

(9) To show list of commits made in the current Git repository

```
(base) C:\academic>git log
commit a3acbfc1c11dd44c4530f2361256bae97b4c37ed (HEAD -> master)
Author: academician <academician@gmail.com>
Date:   Mon Jun 27 15:50:52 2022 +0530
    files are commited
```

(10) Simultaneously add and commit

```
(base) C:\academic>git commit -a -m "shortcut for git add followed by git commit but"
[master ccf2126] shortcut for git add followed by git commit but
 1 file changed, 3 insertions(+), 1 deletion(-)

(base) C:\academic>git commit -m "shortcut for git add followed by git commit but"
On branch master
nothing to commit, working tree clean
```

To show list of commits made in the current Git repository

```
(base) C:\academic>git log
commit ccf2126189e68adddad803879c3a2e34f4857892 (HEAD -> master)
Author: abc <abc@gmail.com>
Date:   Mon Jun 27 16:01:48 2022 +0530

    shortcut for git add followed by git commit but

commit a3acbfc1c11dd44c4530f2361256bae97b4c37ed
Author: academician <academician@gmail.com>
Date:   Mon Jun 27 15:50:52 2022 +0530

    files are commited
```

(11) To get more information about our changes

```
(base) C:\academic>git log -p

(base) C:\academic>git log --stat
```

(12) To see the changes that are staged but not committed

```
(base) C:\academic>git diff --staged

(base) C:\academic>git status
```

To show information about the commit and its associated patch

```
(base) C:\academic>git show
```

(13) To show information about a particular commit and its associated patch

```
(base) C:\academic>git show a3acbfc1c11dd44c4530f2361256bae97b4c37ed
```

(14) Display list of tracked file

```
(base) C:\academic>git ls-tree --full-tree --name-only -r master
```

(15) To take whatever is currently in our staging area and run the git commit workflow to overwrite the previous commit

```
(base) C:\academic>git commit --amend
```

File renamed or deleted

(16) Rename a file and then check the status again (myFile 2 to myFile)

```
(base) C:\academic>git status

(base) C:\academic>git commit -m "file name changed"

(base) C:\academic>git commit -a -m "file name changed"

(base) C:\academic>git add *

(base) C:\academic>git status
```

(17) Undoing Changes Before Committing

```
(base) C:\academic> git checkout myFile.py
Updated 1 path from the index
```

If you need to check out individual changes instead of the whole file, you can do that using the dash p flag.

```
(base) C:\academic> git checkout -p myFile.py
```

```
(base) C:\academic>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   myFile.py
```

(18) Amending Commits

To update the last commit to include changes. we run git commit --amend, git will take whatever is currently in our staging area and run the git commit workflow to overwrite the previous commit.

```
(base) C:\academic>git commit --amend -m "Need to update"
[master db2c1bf] Need to update
 Date: Tue Jun 28 15:20:14 2022 +0530
 1 file changed, 5 insertions(+), 2 deletions(-)
```

If we realize we've added something to the staging area that we didn't want to commit, we can unstage our changes by using the **git reset** command. To remove from stagging area.

```
(base) C:\academic>git reset
```

```
Unstaged changes after reset:
M       myFile.py
```

(19) Rollbacks commit

git revert makes a new commit which effectively rolls back a previous commit. It’s a bit like an undo command.

```
(base) C:\academic>git revert
```

```

```

```
(base) C:\academic>git status
On branch master
nothing to commit, working tree clean
```

(20) To remove a file

```
(base) C:\academic>git rm file1.py
rm 'file1.py'
```

```
(base) C:\academic>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    file1.py
```

```
(base) C:\academic>git commit -m 'Deleted'
[master d3bbb1f] 'Deleted'
 1 file changed, 9 deletions(-)
 delete mode 100644 file1.py
```

```
(base) C:\academic>git status
On branch master
nothing to commit, working tree clean
```

============================ Thank You ===================================