

**Subject Name: Source Code Management**

**Subject Code: 22CS003**

**Session: 2023-24**

**Department: DCSE**

**CHITKARA**  
UNIVERSITY



**Submitted By:**

Aksh Bansal

2310990270-First Year

G4-A

**Submitted To:**

Dr. Chetna Kaushal

### List of Programs

S. No	Program Title	Page No.
1	Setting up of Git Client	
2	Setting up GitHub Account	
3	Generate logs	
4	Create and visualize branches	
5	Git life cycle description	

## EXPERIMENT NO. 1:

### Aim: Setting up of Git Client

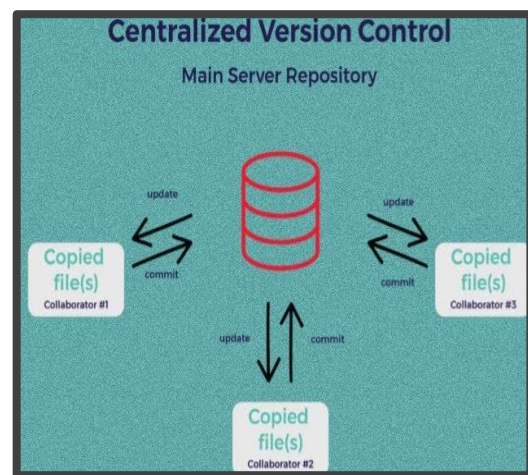
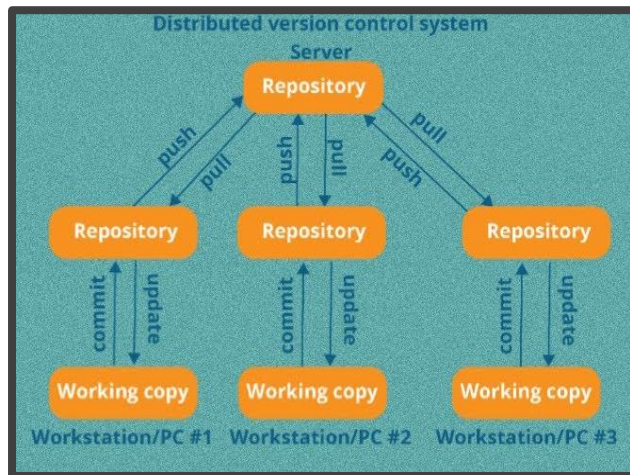
*Theory:*

#### **What is Git?**

Git is a free and open-source version control system used to handle small to very large projects efficiently. This is also used for tracking changes in any set of files and usually helps in coordinating work among members of a team. Hence, enables multiple developers to work together on non-linear development.

**History of VCS:** The very first Version Control System was created in 1972 at Bell Labs where they also developed UNIX. The first one was called SCCS (Source Code Control System). It was available only for UNIX and only worked with Source Code files. Some types of Version Control Systems are:

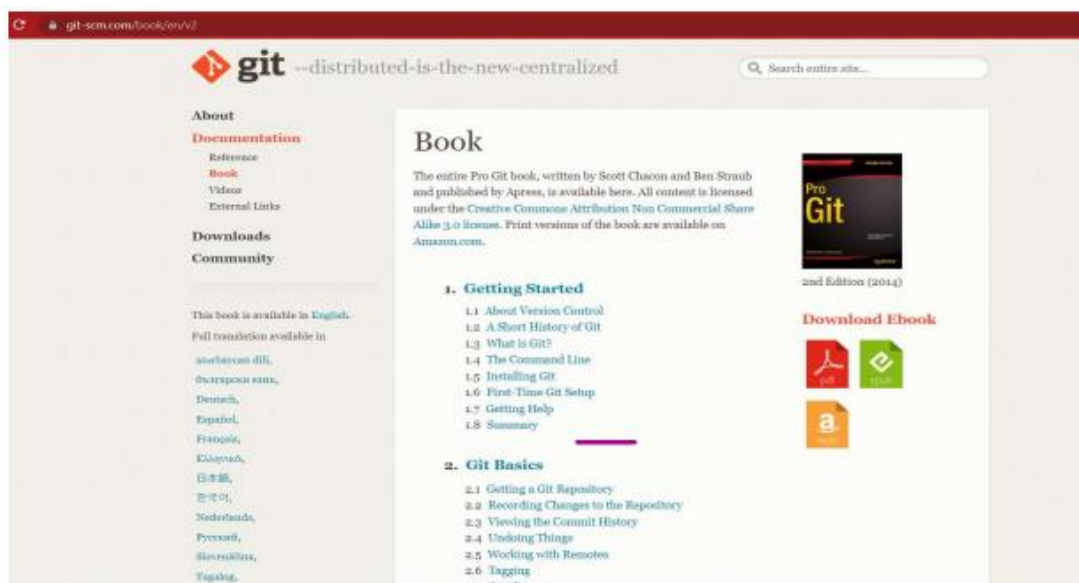
- Local VCS: No internet is needed because it uses a database to keep and track of files.
- Centralized VCS: Centralized version control systems are based on the idea that there is a single “central” copy of your project somewhere (probably on a server), and programmers will “commit” their changes to this central copy. This simply means recording the change in the central system (OS).
- Distributed VCS: A type of version control where the complete codebase including its full version history is mirrored on every developer's computer.



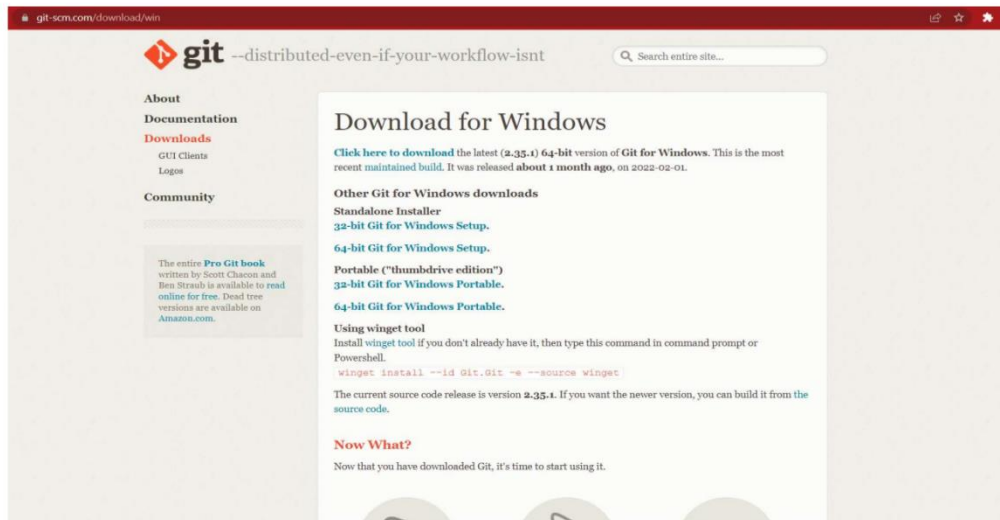
## How to install GIT on Windows?

There are many ways to install Git on Windows. The most official build is available for download on the Git website. Go to <https://git-scm.com/download/win> and after a few settings the download will start automatically.

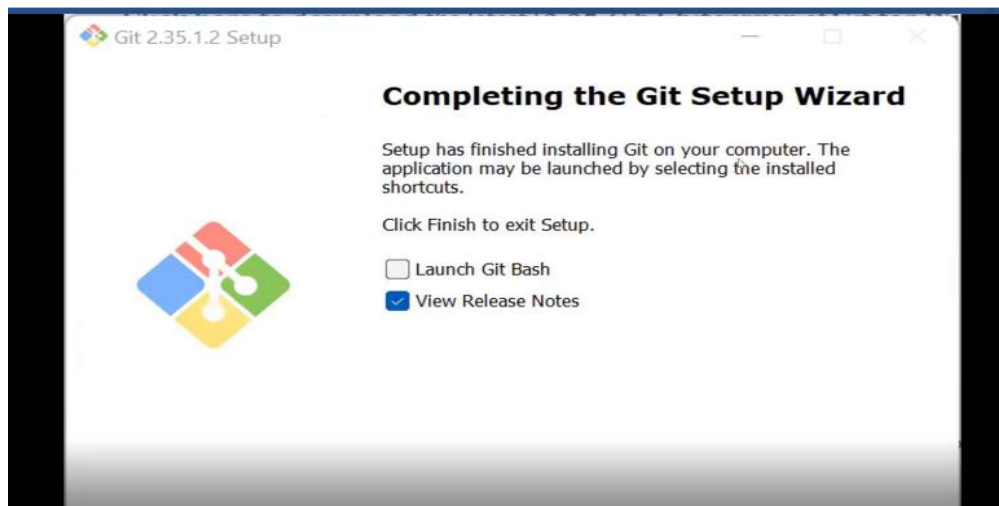
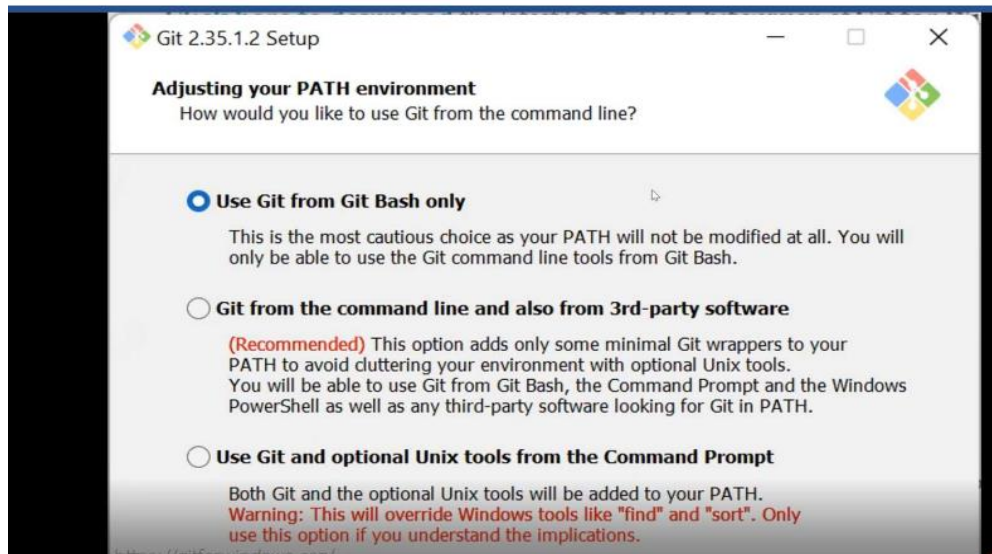
- Visit directly on git book page by <https://git-scm.com/book/en/v2>



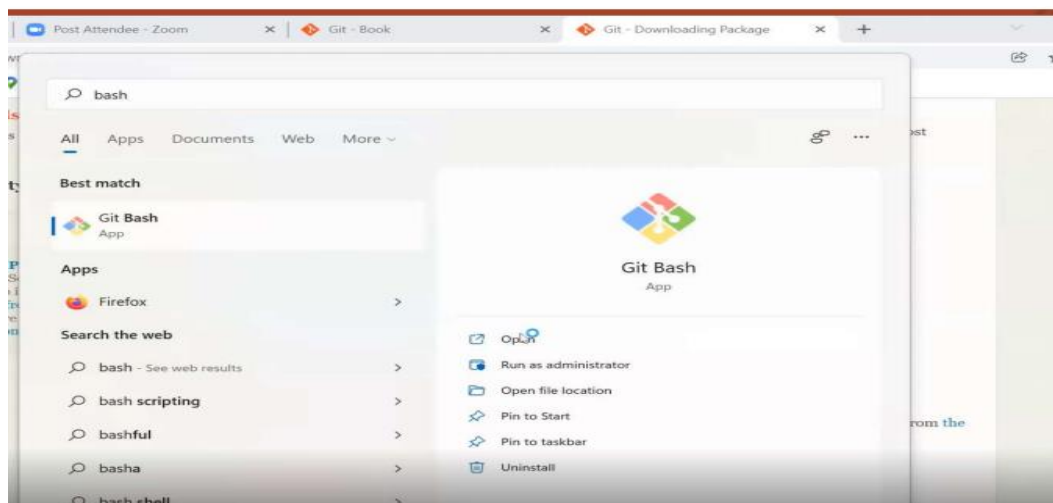
- Then click on Installation Git and click on whatever system you want, available are three- Windows, Apple and Linux.



- After some more simple and easy settings and choosing your favourable environment and doing some SSH settings, it finally starts exporting the files in system and completes the Git hub wizard.



- Git bash got installed in system and seemed and opened on clicking seems of like:



```
MINGW64/c:/Users/Lenovo
Lenovo@LENOVO MINGW64 ~ (main)
$ git --version
git version 2.43.0.windows.1
Lenovo@LENOVO MINGW64 ~ (main)
$
```

You can also check the version of installed software by checking git version.

## **EXPERIMENT NO. 2:**

### **Aim: Setting up GitHub Account**

#### ***Theory:***

#### **What is GitHub?**

GitHub is a code hosting platform for version control and collaboration. GitHub is a development platform inspired by the way you work. From open source to business, we can host and review code, manage projects, and build software alongside 36 million developers.

#### **Advantages:**

- Documentation.
- Showcase your work.
- Markdown.
- GitHub is a repository.
- Track changes in your code across versions.
- Integration options.

#### **Procedure:**

Search about GitHub: <https://github.com/signup>





By signing up for git you must remember your email and pass phases or password. For a new user, you must add your email and click on Sign up for GitHub. Otherwise click on Sign In at the top right corner.

## For linking Git Hub with Git Bash:

Username-

git config --global user.name "username in github"

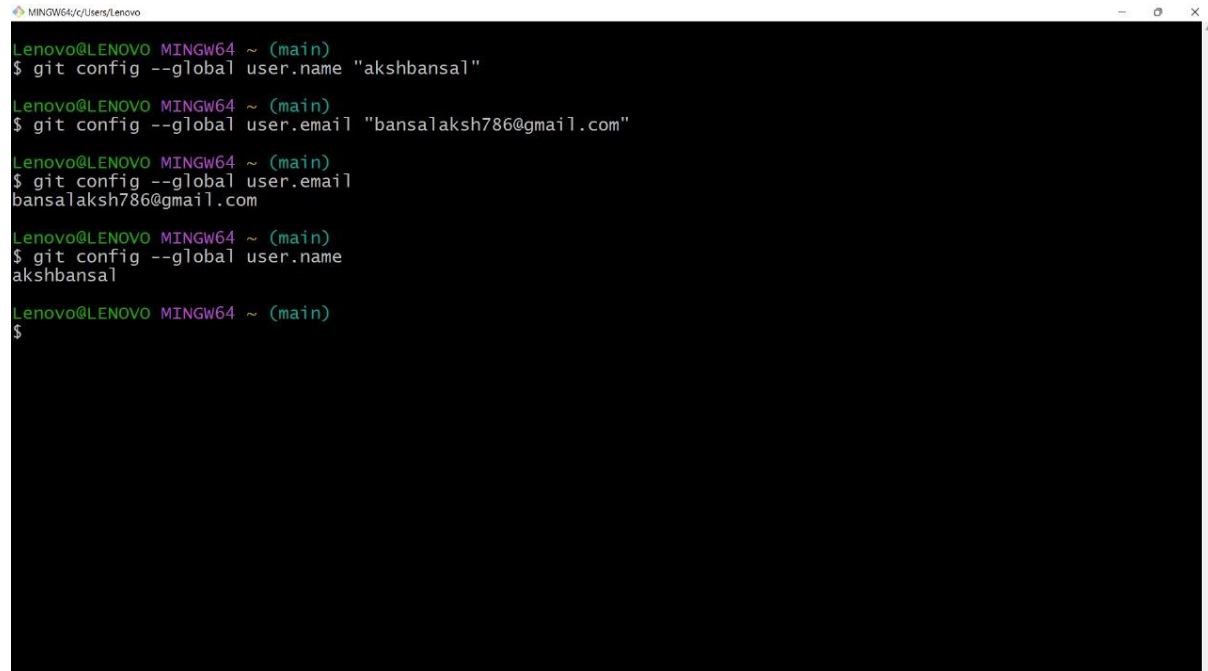
Email-

git config --global user.email "your email in github"

Check Username & Email:

git config user.name

git config user.email



```
Lenovo@LENOVO MINGW64 ~ (main)
$ git config --global user.name "akshbansal"

Lenovo@LENOVO MINGW64 ~ (main)
$ git config --global user.email "bansalaksh786@gmail.com"

Lenovo@LENOVO MINGW64 ~ (main)
$ git config --global user.email
bansalaksh786@gmail.com

Lenovo@LENOVO MINGW64 ~ (main)
$ git config --global user.name
akshbansal

Lenovo@LENOVO MINGW64 ~ (main)
$
```

## **EXPERIMENT NO. 3:**

### **Aim: Generate Logs on Git Hub**

#### ***Theory:***

**Git Logs:** The git log command shows a list of all the commits made to a repository. You can see the hash of each Git commit, the message associated with each commit, and more metadata. This command is basically used for displaying the history of a repository.

#### **Why do we need logs?**

Git log is a utility tool to review and read a history of everything that happens to a repository. Anything we change at what time, by which log, everything is getting recorded in git logs.

```

MINGW64/d/scm_file
Lenovo@LENOVO MINGW64 /d
$ mkdir scm_file

Lenovo@LENOVO MINGW64 /d
$ cd scm_file/

Lenovo@LENOVO MINGW64 /d/scm_file
$ git init
Initialized empty Git repository in D:/scm_file/.git/

Lenovo@LENOVO MINGW64 /d/scm_file (master)
$ touch scmfile.txt

Lenovo@LENOVO MINGW64 /d/scm_file (master)
$ vi scmfile.txt

Lenovo@LENOVO MINGW64 /d/scm_file (master)
$ cat scmfile.txt
Hello again

Lenovo@LENOVO MINGW64 /d/scm_file (master)
$ ls
scmfile.txt

Lenovo@LENOVO MINGW64 /d/scm_file (master)
$

```

```

MINGW64/d/scmpractical
$ cat scmfile.txt
hello again

Lenovo@LENOVO MINGW64 /d/scmpractical (master)
$ ls
scmfile.txt

Lenovo@LENOVO MINGW64 /d/scmpractical (master)
$ git add scmfile.txt
warning: in the working copy of 'scmfile.txt', LF will
be replaced by CRLF the next time Git touches it

Lenovo@LENOVO MINGW64 /d/scmpractical (master)
$ git commit -m "commit 1 on scmfile.txt"
[master (root-commit) b0ea103] commit 1 on scmfile.txt
1 file changed, 1 insertion(+)
create mode 100644 scmfile.txt

Lenovo@LENOVO MINGW64 /d/scmpractical (master)
$ git status
On branch master
nothing to commit, working tree clean

Lenovo@LENOVO MINGW64 /d/scmpractical (master)
$ git log
commit b0ea103d0ea6bea076474e77b953290d9dc06ed0 (HEAD -
> master)
Author: akshbansal <bansalaksh786@gmail.com>
Date: Sun Feb 11 21:45:33 2024 +0530

    commit 1 on scmfile.txt

Lenovo@LENOVO MINGW64 /d/scmpractical (master)
$

```

You can use command **git log** to access logs (every change you make with time and date).

## EXPERIMENT NO. 04

### Aim: Creating and Visualizing the Branches On Git Client

#### Theory:

#### How to create branches?

The main branch in which we are working is master branch. you can use the “git branch” command with the branch name and the commit SHA for the new branch.

1. For creating a new branch: git branch “name of the branch”.



```

MINGW64/d/scmpractical
Lenovo@LENOVO MINGW64 /d/scmpractical (master)
$ git branch feature1
Lenovo@LENOVO MINGW64 /d/scmpractical (master)
$ |
  
```

2. To check how many branches we have:



```

MINGW64/d/scm_file
Lenovo@LENOVO MINGW64 /d/scm_file (master)
$ git branch feature1
Lenovo@LENOVO MINGW64 /d/scm_file (master)
$ git branch
* master
  feature1
Lenovo@LENOVO MINGW64 /d/scm_file (master)
$
  
```

As you can see here one branch is showing that I create- feature1.

3. To change the present working branch: git checkout "name of the branch" and command to go back to the master directory:

```

MINGW64/d/scmpactical
Lenovo@LENOVO MINGW64 /d/scmpactical (master)
$ git branch feature1

Lenovo@LENOVO MINGW64 /d/scmpactical (master)
$ git branch
feature1
* master

Lenovo@LENOVO MINGW64 /d/scmpactical (master)
$ git checkout feature1
Switched to branch 'feature1'

Lenovo@LENOVO MINGW64 /d/scmpactical (feature1)
$ |

```

Here, you see by using checkout command we can switch branches and from branches to even master branch too.

### Visualizing branches:

for visualizing, we have to create a new file in the branch that we made "feature1" instead of the master branch. After this we have to do three step architecture that is working directory, staging area and git repository.

Firstly I've changed the branch from master to feature1 that I previously made and after that I check git status. Now I add text in scmfile2 file (scmfile.txt) and use git add "file\_name". Then I use git commit -m "key\_name" command for the changes I made and insertions I do.

At last I check my activities with the help of git log command.

```

MINGW64/d/scmpractical

Lenovo@LENOVO MINGW64 /d/scmpractical (feature1)
$ git status
On branch feature1
nothing to commit, working tree clean

Lenovo@LENOVO MINGW64 /d/scmpractical (feature1)
$ touch scmfile2.txt

Lenovo@LENOVO MINGW64 /d/scmpractical (feature1)
$ vi scmfile2.txt

Lenovo@LENOVO MINGW64 /d/scmpractical (feature1)
$ git add scmfile2.txt
warning: in the working copy of 'scmfile2.txt', LF will
be replaced by CRLF the next time Git touches it

Lenovo@LENOVO MINGW64 /d/scmpractical (feature1)
$ git commit -m "commit 2 on scmfile2.txt"
[feature1 36287a6] commit 2 on scmfile2.txt
1 file changed, 1 insertion(+)
create mode 100644 scmfile2.txt

Lenovo@LENOVO MINGW64 /d/scmpractical (feature1)
$ git log
commit 36287a61b6b36fbaad9fe16b13864fbd8b3b3f3 (HEAD -
> feature1)
Author: akshbansal <bansalaksh786@gmail.com>
Date: Sun Feb 11 21:49:50 2024 +0530

    commit 2 on scmfile2.txt

commit b0ea103d0ea6bea076474e77b953290d9dc06ed0 (master
)
Author: akshbansal <bansalaksh786@gmail.com>
Date: Sun Feb 11 21:45:33 2024 +0530

    commit 1 on scmfile.txt

Lenovo@LENOVO MINGW64 /d/scmpractical (feature1)
$ ls
scmfile.txt  scmfile2.txt

Lenovo@LENOVO MINGW64 /d/scmpractical (feature1)
$

```

## **EXPERIMENT NO. 05**

### **Aim: Git lifecycle description**

*Theory:*

**Stages in GIT Life Cycle:** Files in a Git project have various stages like Creation, Modification, Refactoring, and Deletion and so on. Irrespective of whether this project is tracked by Git or not, these phases are still prevalent. However, when a project is under Git version control system, they are present in three major Git states in addition to these basic ones. Here are the three Git states:

- Working directory
- Staging area
- Git directory

#### **Working Directory:**

When a project is residing in our local system we don't know whether the project is tracked by Git or not. In any of the case, this project directory is called our Working directory.

#### **Staging Area:**

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)

#### **Git Directory:**



The .git folder contains all information that is necessary for the project and all information relating commits, remote repository address, etc. It also contains a log that stores the commit history. This log can help you to roll back to the desired version of the code

**Remote Repository:** Remote repositories are hosted on a server that is accessible for all team members - most likely on the internet or on a local network. Assessable and reachable by all.

**Screenshot:**



```

MINGW64/d/scmpractical/git-project
Lenovo@LENOVO MINGW64 /d/scmpractical (feature1)
$ git branch beta

Lenovo@LENOVO MINGW64 /d/scmpractical (feature1)
$ pwd
/d/scmpractical

Lenovo@LENOVO MINGW64 /d/scmpractical (feature1)
$ mkdir git-project

Lenovo@LENOVO MINGW64 /d/scmpractical (feature1)
$ cd git-project

Lenovo@LENOVO MINGW64 /d/scmpractical/git-project (feature1)
$ git init
Initialized empty Git repository in D:/scmpractical/git-project/.git/

Lenovo@LENOVO MINGW64 /d/scmpractical/git-project (master)
$ touch python.py

Lenovo@LENOVO MINGW64 /d/scmpractical/git-project (master)
$ vi python.py

Lenovo@LENOVO MINGW64 /d/scmpractical/git-project (master)
$ git add python.py
warning: in the working copy of 'python.py', LF will be replaced by CRLF the next time Git touches it

Lenovo@LENOVO MINGW64 /d/scmpractical/git-project (master)
$ git commit -m "created new file"
[master (root-commit) 0a31bf0] created new file
1 file changed, 1 insertion(+)
create mode 100644 python.py

Lenovo@LENOVO MINGW64 /d/scmpractical/git-project (master)
$ git branch beta

Lenovo@LENOVO MINGW64 /d/scmpractical/git-project (master)
$ git checkout beta
Switched to branch 'beta'

Lenovo@LENOVO MINGW64 /d/scmpractical/git-project (beta)
$ git branch
* beta
  master

Lenovo@LENOVO MINGW64 /d/scmpractical/git-project (beta)
$

```