SOFTWARE ENGINEERING

UNIT – 2

TOPIC – 7

BASIC GIT COMMANDS - VERSION, CONFIG, INIT, STATUS, ADD, COMMIT, DIFF, HELP

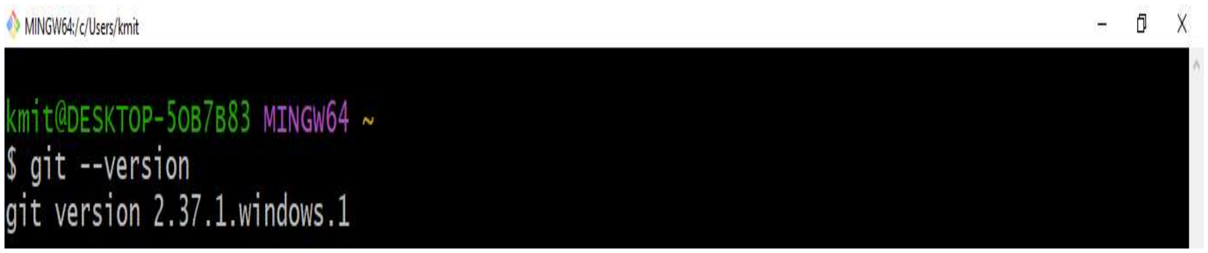
Basic Git Commands

1. **Git Version:**

   The `git --version` command is used to check if Git is installed on your system. This command displays the current version of Git that is available.

   For example, running: `git --version`

might return `git version 2.37.1`, confirming that Git is installed.



```
kmit@DESKTOP-5OB7B83 MINGW64 ~
$ git --version
git version 2.37.1.windows.1
```

Knowing the installed Git version helps in troubleshooting and ensuring compatibility with certain Git features or integrations. It is the first step before using Git functionalities.

2. **Git Config:**

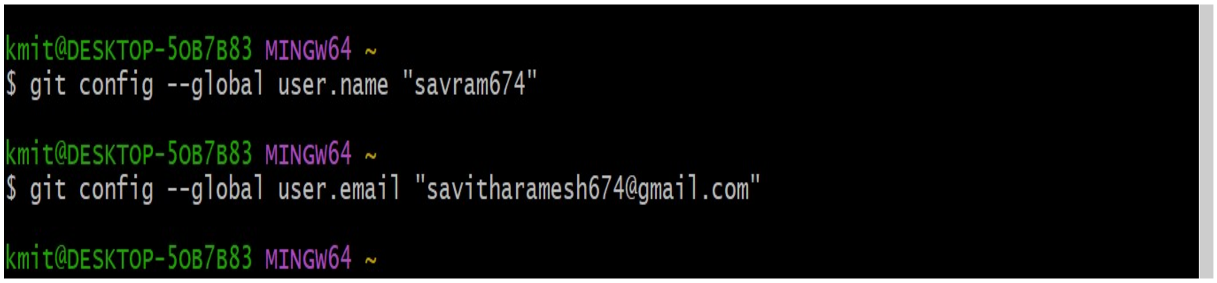
   The `git config` command is used to set up the user identity for commits, including name and email address. This ensures that every commit you make in the project history is associated with your correct identity.

Usage Example:

```
git config --global user.name "Your Name"
git config --global user.email "your.email@example.com"
```

Explanation:

- `--global:` Applies the configuration globally for the system, meaning it applies to all projects.
- `user.name` and `user.email:` These options specify the identity that will appear in the commit history.



```
kmit@DESKTOP-5OB7B83 MINGW64 ~
$ git config --global user.name "savram674"

kmit@DESKTOP-5OB7B83 MINGW64 ~
$ git config --global user.email "savitharamesh674@gmail.com"

kmit@DESKTOP-5OB7B83 MINGW64 ~
```

Example Output:

```
git config --global user.name "Madhurika"
git config --global user.email "madhurika.kmit@example.com"
```

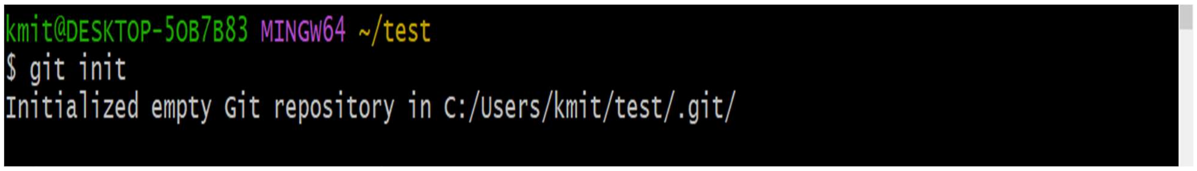
In this example, Jane Doe's details will be associated with all commits made on this system unless overridden for a specific repository.

3. Git Init:

The `git init` command is used to create a new Git repository. This initializes a directory as a new Git repository and creates a `.git` subdirectory to store all Git-related files.

Usage Example: `git init`

Explanation: This command creates an empty Git project within the directory. After running `git init`, the project is now under version control, and you can start tracking changes.



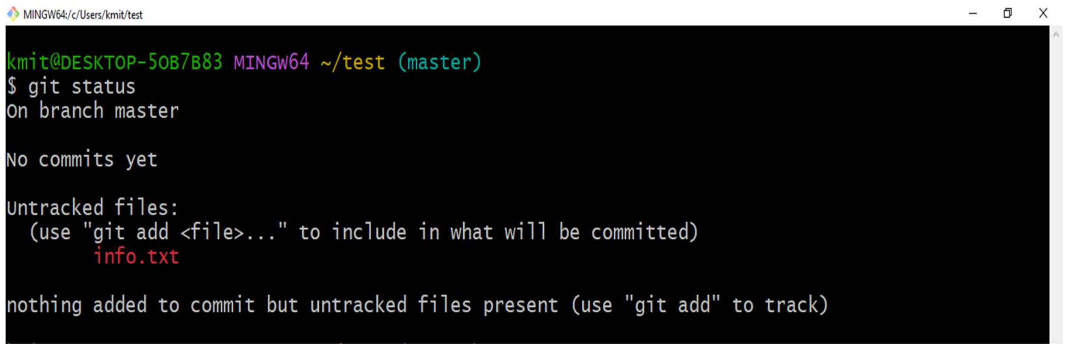
```
kmit@DESKTOP-5OB7B83 MINGW64 ~/test
$ git init
Initialized empty Git repository in C:/Users/kmit/test/.git/
```

Real-World Example: Imagine you have a directory called `my-project`. Running `git init` inside it will turn it into a Git repository, allowing you to track its changes.

4. **Git Status:**

   The `git status` command displays the state of your working directory and staging area. It shows what changes have been staged, what files are not being tracked by Git, and what changes are yet to be committed.

Usage Example: `git status`



```
MINGW64:/c/Users/kmit/test
kmit@DESKTOP-5OB7B83 MINGW64 ~/test (master)
$ git status
On branch master

No commits yet

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

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

Explanation: This command is essential to get a summary of your repository's current state. It lets you know whether files need to be added to the staging area or if there are changes to commit.

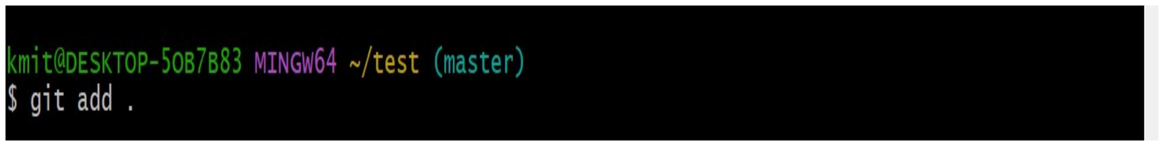
5. **Git Add:**

   The `git add` command is used to add changes to the staging area. You must stage changes before committing them to ensure only the desired changes are included in the commit.

Usage Example: `git add .`

Explanation:

- `.` stages all changes in the current directory.
- Specific files can be added by replacing `.` with the file names.



```
kmit@DESKTOP-5OB7B83 MINGW64 ~/test (master)
$ git add .
```

Example: `git add index.html`

This adds the file `index.html` to the staging area.

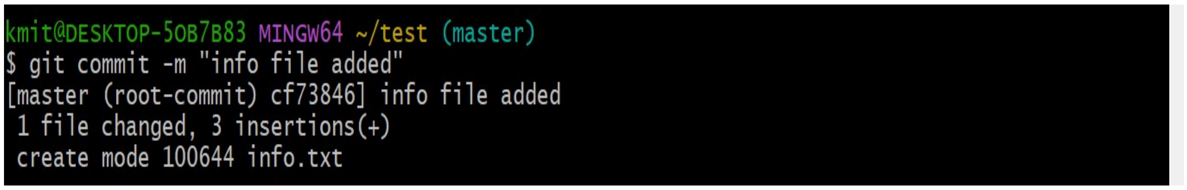
6. **Git Commit:**

   The `git commit` command saves your changes to the local repository. Each commit represents a snapshot of the repository at a particular point in time. You must include a message explaining what the commit does.

Usage Example: `git commit -m "Added new feature"`

Explanation:

- `-m`: Allows you to add a commit message directly from the command line.
- The commit message should be concise but descriptive enough to explain the change.



```
kmit@DESKTOP-50B7B83 MINGW64 ~/test (master)
$ git commit -m "info file added"
[master (root-commit) cf73846] info file added
 1 file changed, 3 insertions(+)
 create mode 100644 info.txt
```

Example Output:

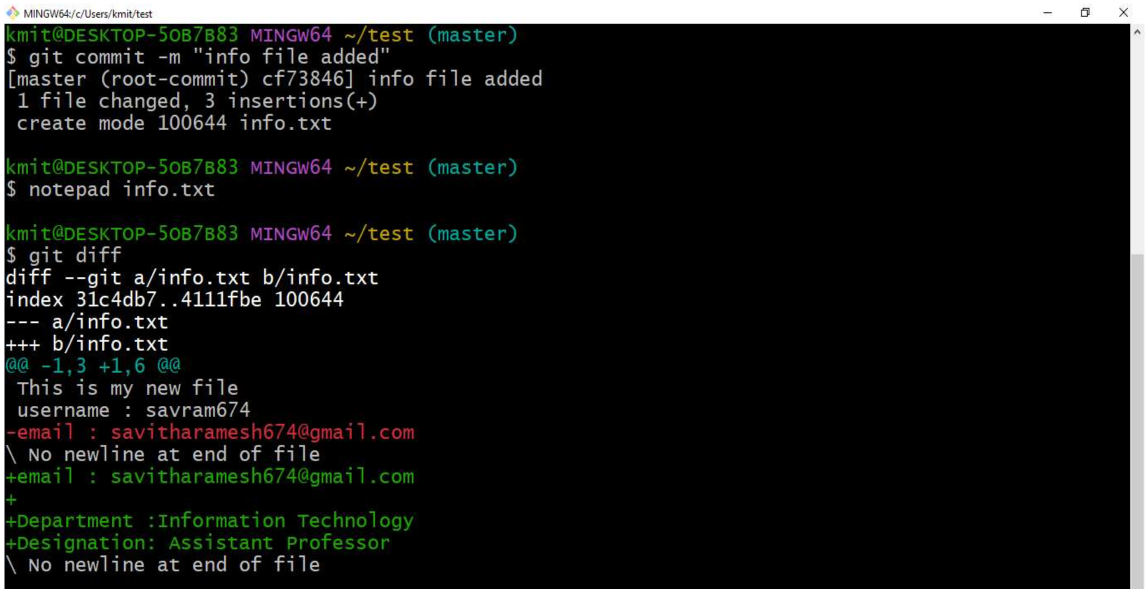
```
[master (root-commit) 1a2b3c4] Added new feature
```

7. **Git Diff:**

   The `git diff` command shows the differences between files in your working directory and the staging area. It is useful to review changes before committing them.

Usage Example:

```
git diff
```



```
MINGW64:/c/Users/kmit/test
kmit@DESKTOP-5OB7B83 MINGW64 ~/test (master)
$ git commit -m "info file added"
[master (root-commit) cf73846] info file added
 1 file changed, 3 insertions(+)
 create mode 100644 info.txt

kmit@DESKTOP-5OB7B83 MINGW64 ~/test (master)
$ notepad info.txt

kmit@DESKTOP-5OB7B83 MINGW64 ~/test (master)
$ git diff
diff --git a/info.txt b/info.txt
index 31c4db7..4111fbe 100644
--- a/info.txt
+++ b/info.txt
@@ -1,3 +1,6 @@
 This is my new file
 username : savram674
-email : savitharamesh674@gmail.com
\ No newline at end of file
+email : savitharamesh674@gmail.com
+
+Department :Information Technology
+Designation: Assistant Professor
\ No newline at end of file
```

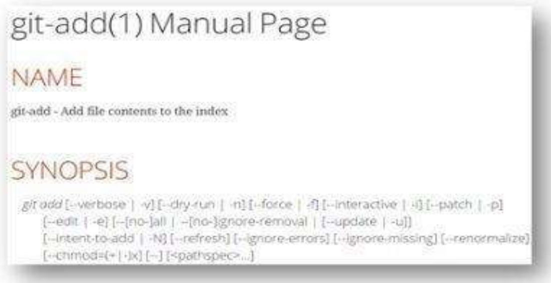
Explanation: Running `git diff` will display the differences between your modified files and the last commit, helping you see exactly what changes will be staged and committed.

8. **Git Help:**

   The `git help` command provides documentation and detailed usage information for any Git command.

Usage Example: `git help <command>`

Explanation: You can use `git help` to get detailed information about any Git command, such as `git help add` to learn more about the `git add` command.



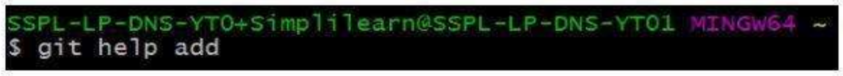
git-add(1) Manual Page

NAME

git-add - Add file contents to the index

SYNOPSIS

```
git add [--verbose | -v] [--dry-run | -n] [--force | -f] [--interactive | -i] [--patch | -p]
        [--edit | -e] [--[no-]all | --[no-]ignore-removal | [--update | -u]]
        [--intent-to-add | -N] [--refresh] [--ignore-errors] [--ignore-missing] [--renormalize]
        [--chmod=(+|-)x] [--] [<pathspec>...]
```



```
SSPL-LP-DNS-YTO+Simplilearn@SSPL-LP-DNS-YT01 MINGW64 ~
$ git help add
```