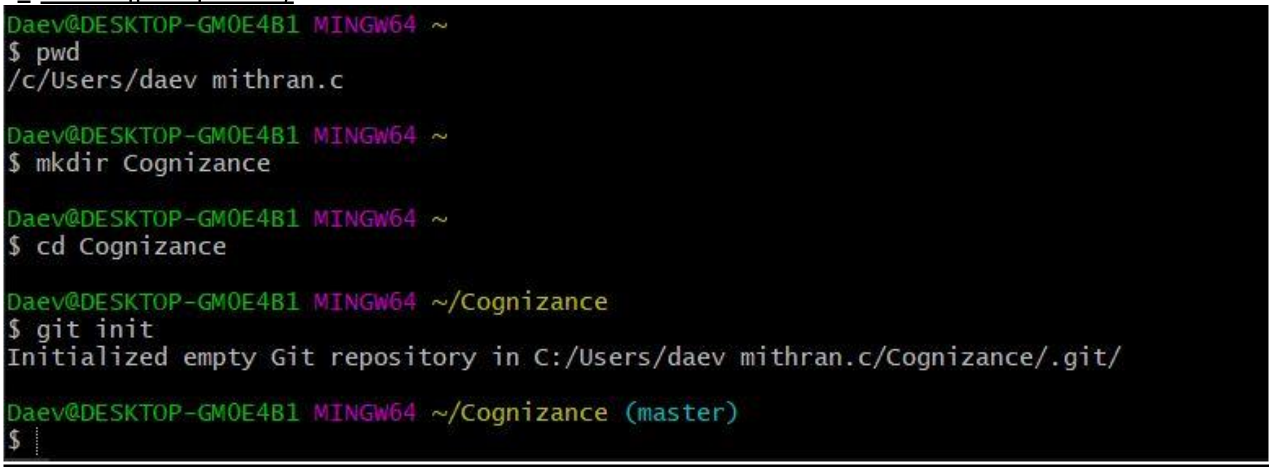
COGNIZANCE

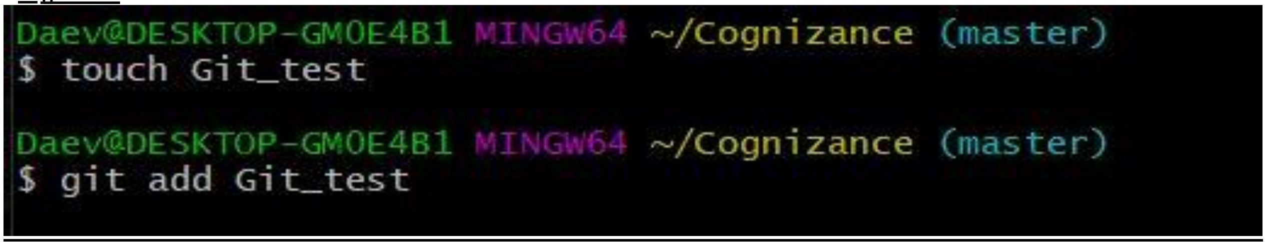
TASK 1 [GIT & GITHUB]

1. Creating a repository



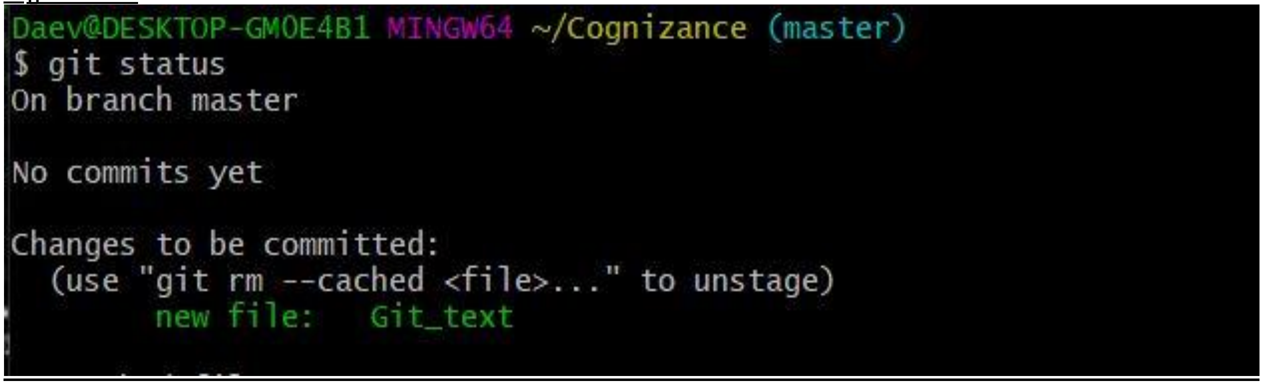
```
Daev@DESKTOP-GM0E4B1 MINGW64 ~
$ pwd
/c/Users/daev mithran.c

Daev@DESKTOP-GM0E4B1 MINGW64 ~
$ mkdir Cognizance

Daev@DESKTOP-GM0E4B1 MINGW64 ~
$ cd Cognizance

Daev@DESKTOP-GM0E4B1 MINGW64 ~/Cognizance
$ git init
Initialized empty Git repository in C:/Users/daev mithran.c/Cognizance/.git/

Daev@DESKTOP-GM0E4B1 MINGW64 ~/Cognizance (master)
$
```

2. git add



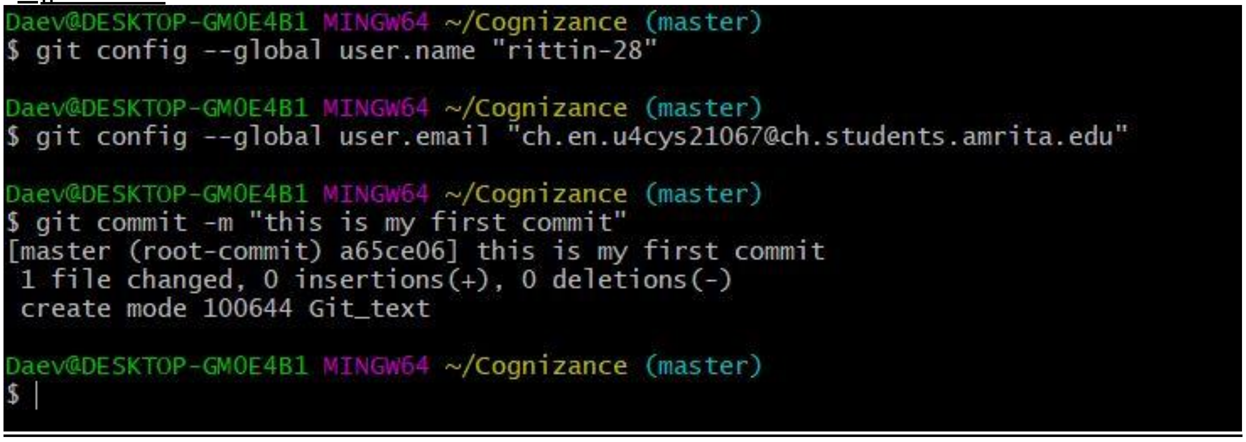
```
Daev@DESKTOP-GM0E4B1 MINGW64 ~/Cognizance (master)
$ touch Git_test

Daev@DESKTOP-GM0E4B1 MINGW64 ~/Cognizance (master)
$ git add Git_test
```

3.git status



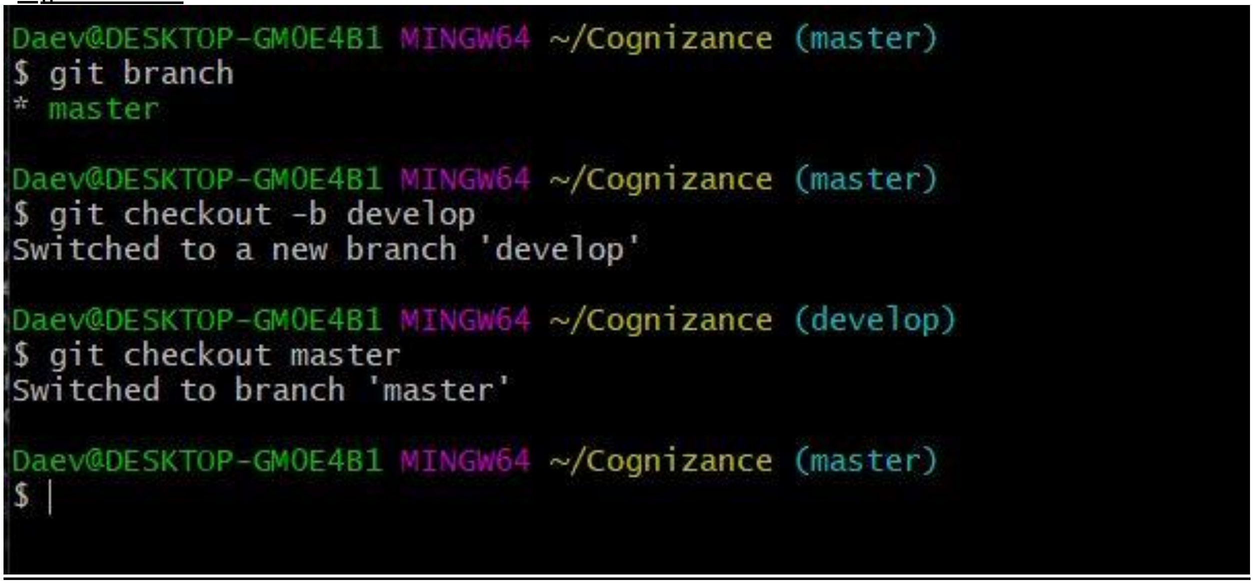
```
Daev@DESKTOP-GM0E4B1 MINGW64 ~/Cognizance (master)
$ git status
On branch master

No commits yet

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

4.git commit



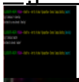
```
Daev@DESKTOP-GM0E4B1 MINGW64 ~/Cognizance (master)
$ git config --global user.name "rittin-28"

Daev@DESKTOP-GM0E4B1 MINGW64 ~/Cognizance (master)
$ git config --global user.email "ch.en.u4cys21067@ch.students.amrita.edu"

Daev@DESKTOP-GM0E4B1 MINGW64 ~/Cognizance (master)
$ git commit -m "this is my first commit"
[master (root-commit) a65ce06] this is my first commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Git_text

Daev@DESKTOP-GM0E4B1 MINGW64 ~/Cognizance (master)
$ |
```

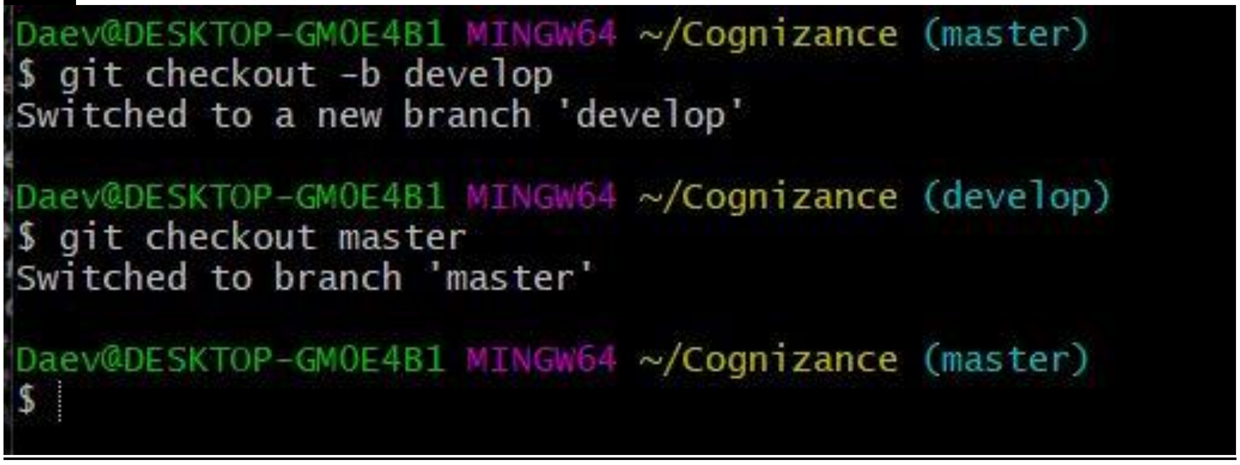
5.git checkout



```
Daev@DESKTOP-GM0E4B1 MINGW64 ~/Cognizance (master)
$ git branch
* master

Daev@DESKTOP-GM0E4B1 MINGW64 ~/Cognizance (master)
$ git checkout -b develop
Switched to a new branch 'develop'

Daev@DESKTOP-GM0E4B1 MINGW64 ~/Cognizance (develop)
$ git checkout master
Switched to branch 'master'

Daev@DESKTOP-GM0E4B1 MINGW64 ~/Cognizance (master)
$ |
```

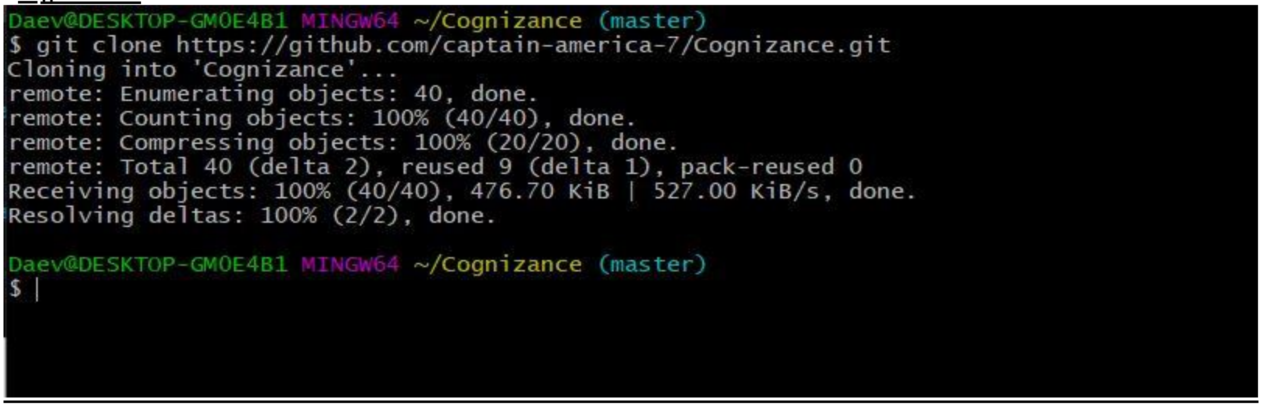
6.git checkout -b





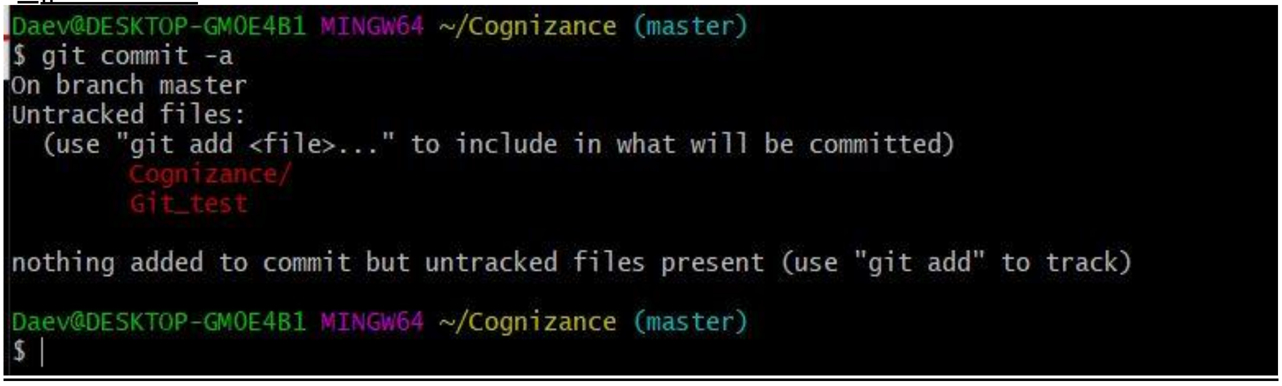
```
Daev@DESKTOP-GM0E4B1 MINGW64 ~/Cognizance (master)
$ git checkout -b develop
Switched to a new branch 'develop'

Daev@DESKTOP-GM0E4B1 MINGW64 ~/Cognizance (develop)
$ git checkout master
Switched to branch 'master'

Daev@DESKTOP-GM0E4B1 MINGW64 ~/Cognizance (master)
$
```

7.git clone



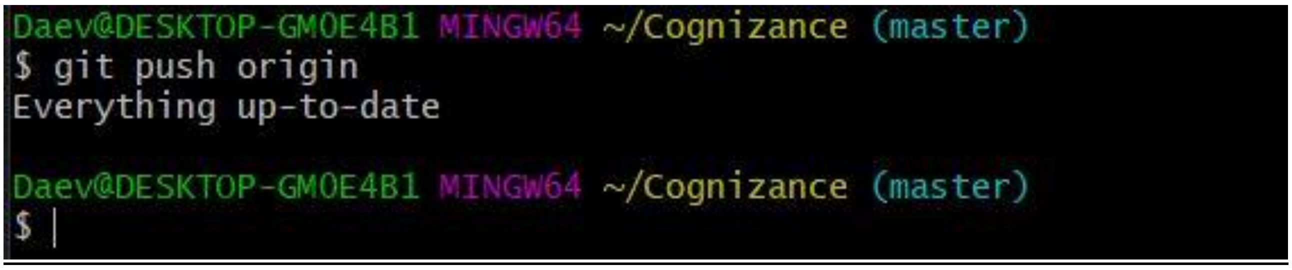
```
Daev@DESKTOP-GM0E4B1 MINGW64 ~/Cognizance (master)
$ git clone https://github.com/captain-america-7/Cognizance.git
Cloning into 'Cognizance'...
remote: Enumerating objects: 40, done.
remote: Counting objects: 100% (40/40), done.
remote: Compressing objects: 100% (20/20), done.
remote: Total 40 (delta 2), reused 9 (delta 1), pack-reused 0
Receiving objects: 100% (40/40), 476.70 KiB | 527.00 KiB/s, done.
Resolving deltas: 100% (2/2), done.

Daev@DESKTOP-GM0E4B1 MINGW64 ~/Cognizance (master)
$ |
```

8.git commit -a



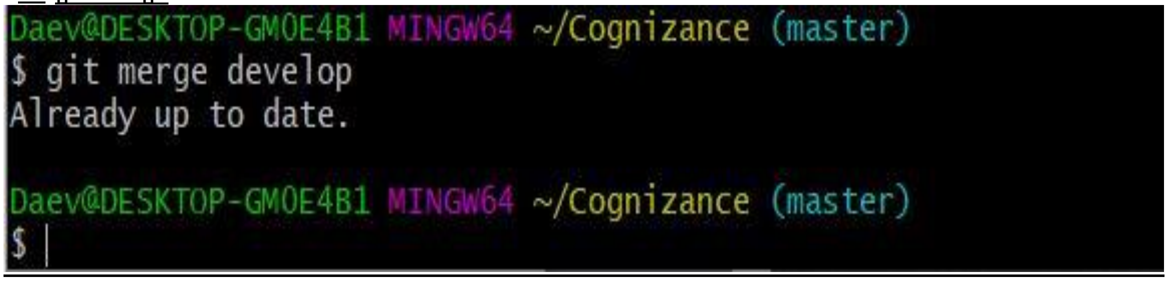
```
Daev@DESKTOP-GM0E4B1 MINGW64 ~/Cognizance (master)
$ git commit -a
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Cognizance/
        Git_test

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

Daev@DESKTOP-GM0E4B1 MINGW64 ~/Cognizance (master)
$ |
```

9.git push



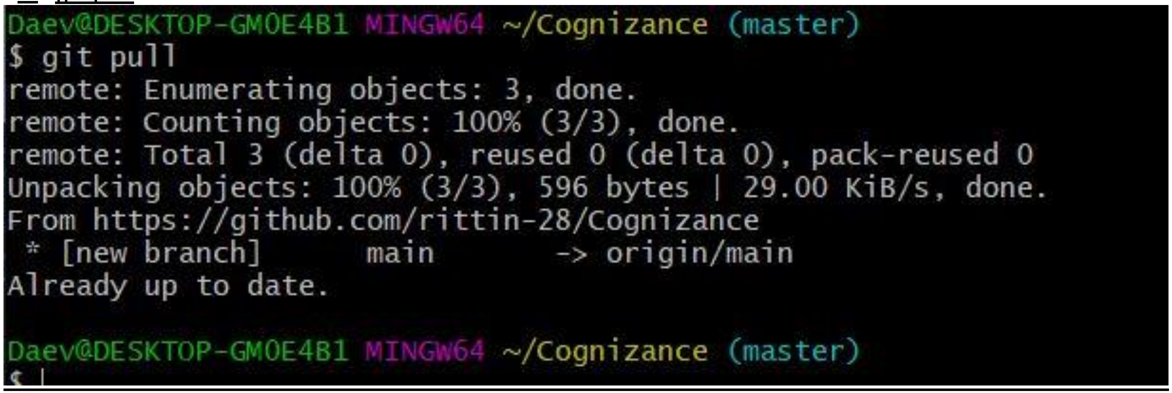
```
Daev@DESKTOP-GM0E4B1 MINGW64 ~/Cognizance (master)
$ git push origin
Everything up-to-date

Daev@DESKTOP-GM0E4B1 MINGW64 ~/Cognizance (master)
$ |
```

10. git merge



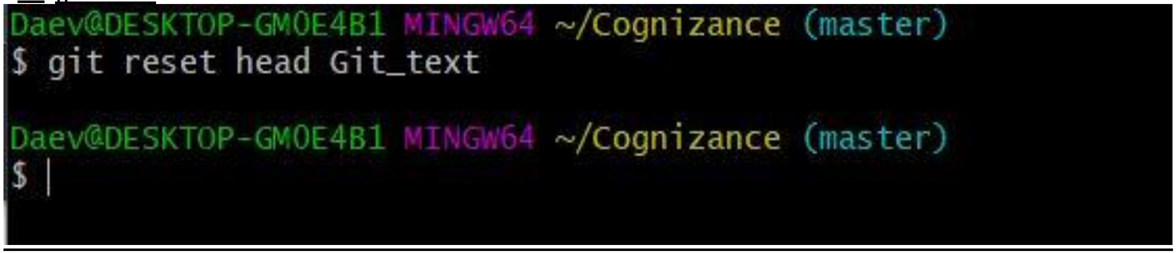
```
Daev@DESKTOP-GM0E4B1 MINGW64 ~/Cognizance (master)
$ git merge develop
Already up to date.

Daev@DESKTOP-GM0E4B1 MINGW64 ~/Cognizance (master)
$ |
```

11. git pull



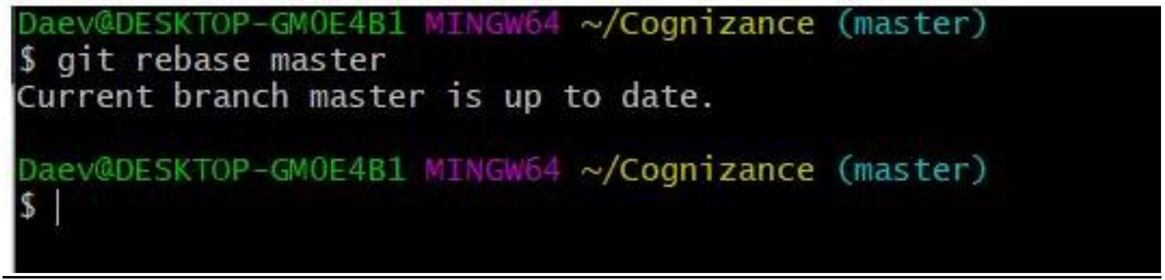
```
Daev@DESKTOP-GM0E4B1 MINGW64 ~/Cognizance (master)
$ git pull
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 596 bytes | 29.00 KiB/s, done.
From https://github.com/rittin-28/Cognizance
 * [new branch]      main       -> origin/main
Already up to date.

Daev@DESKTOP-GM0E4B1 MINGW64 ~/Cognizance (master)
```

12. git reset

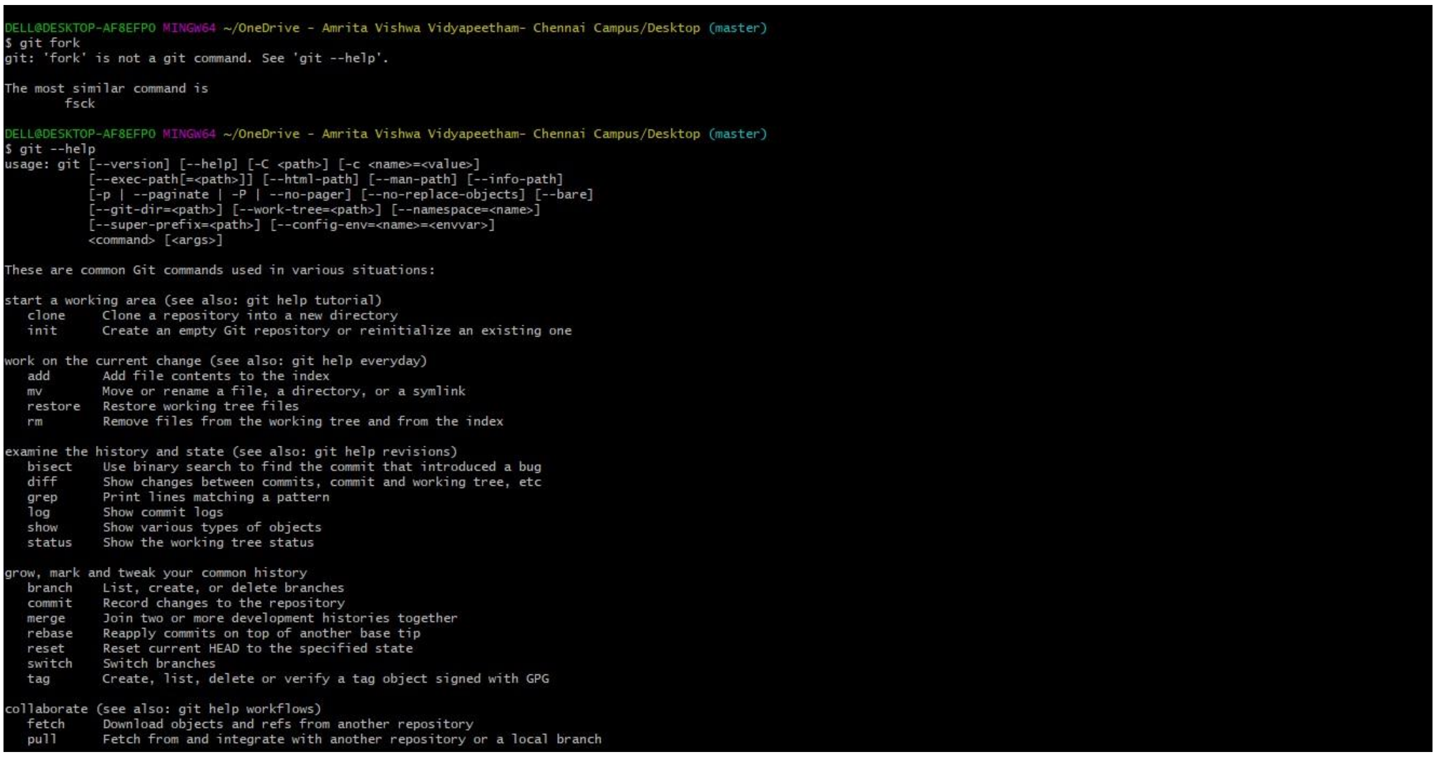


```
Daev@DESKTOP-GM0E4B1 MINGW64 ~/Cognizance (master)
$ git reset head Git_text

Daev@DESKTOP-GM0E4B1 MINGW64 ~/Cognizance (master)
$ |
```

13. git rebase

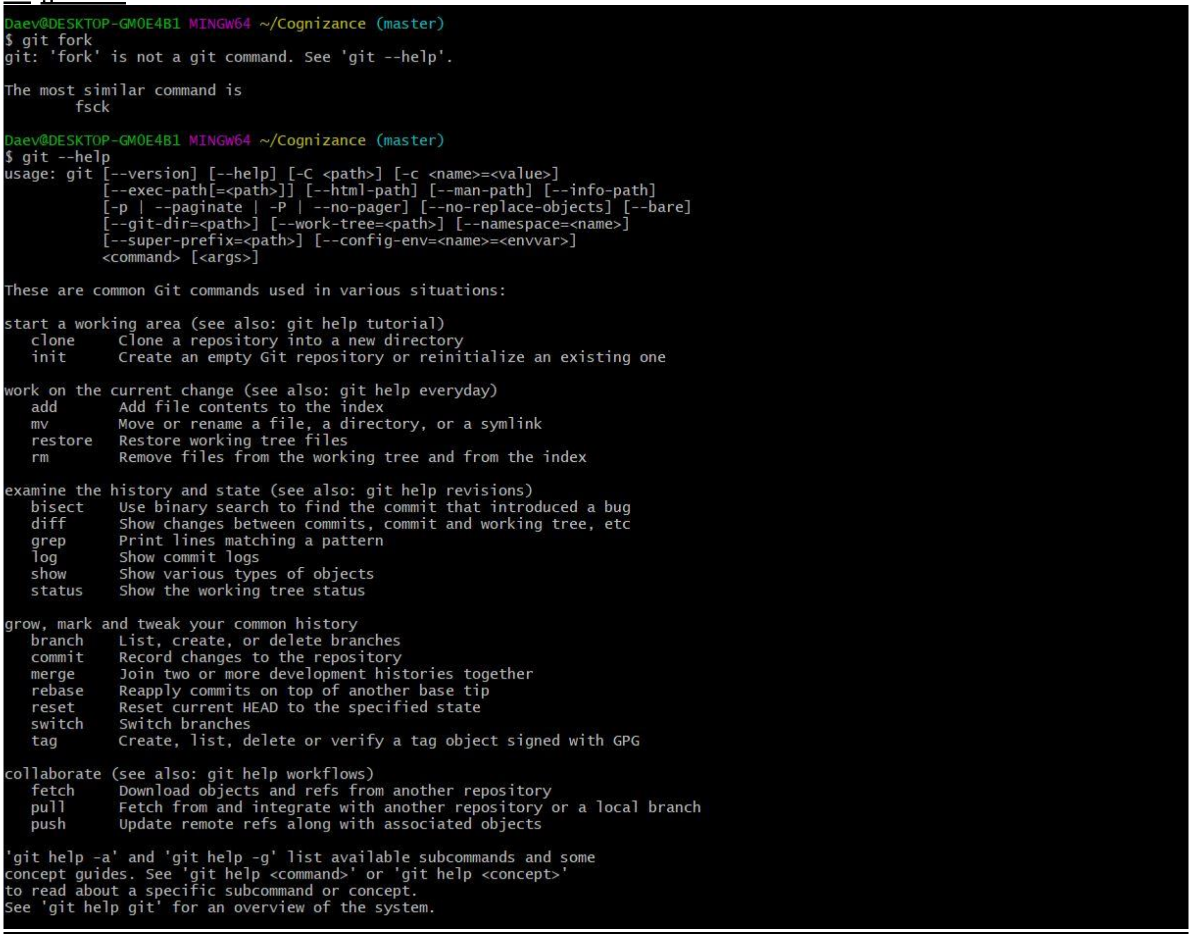


```
Daev@DESKTOP-GM0E4B1 MINGW64 ~/Cognizance (master)
$ git rebase master
Current branch master is up to date.

Daev@DESKTOP-GM0E4B1 MINGW64 ~/Cognizance (master)
$ |
```

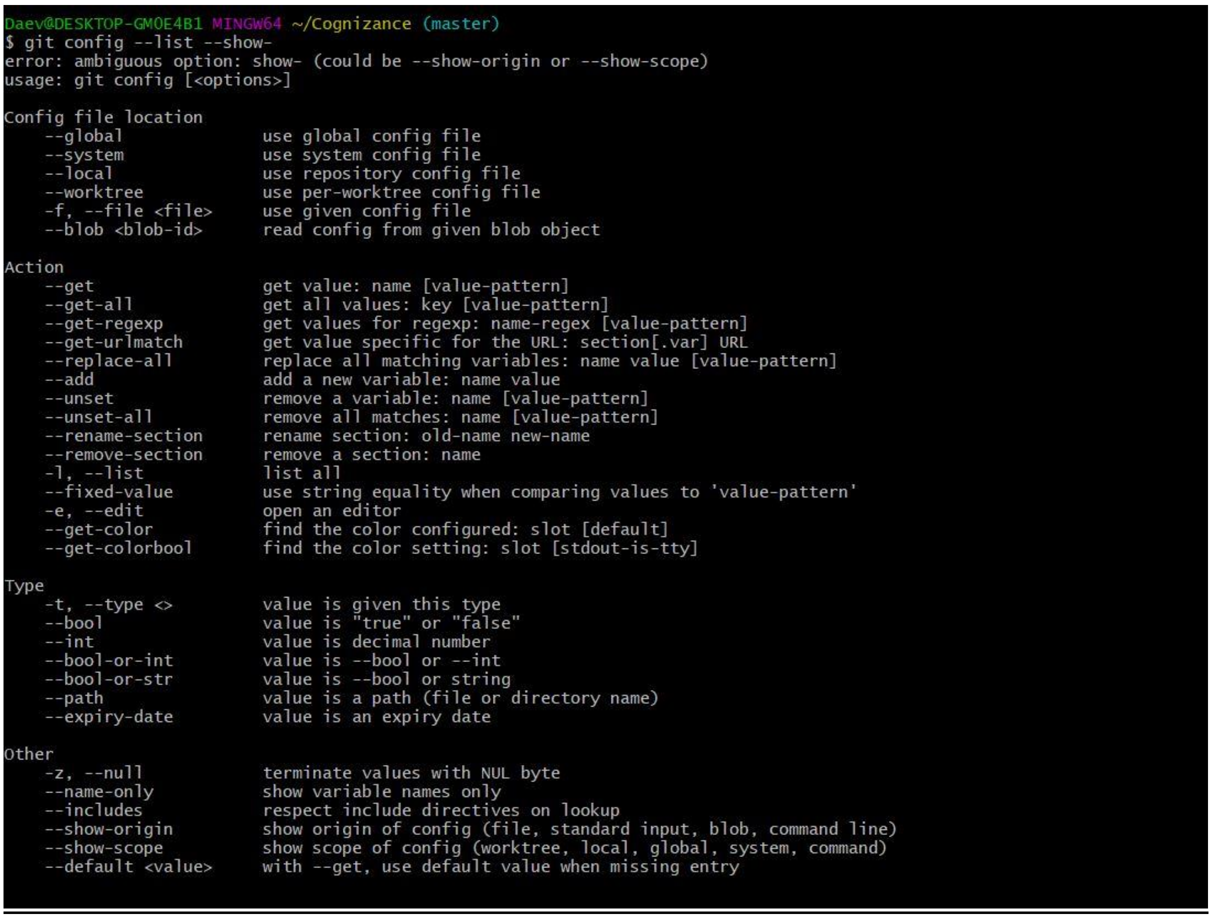


```
DELL@DESKTOP-AF8EFP0 MINGW64 ~/OneDrive - Amrita Vishwa Vidyapeetham- Chennai Campus/Desktop (master)
$ git fork
git: 'fork' is not a git command. See 'git --help'.

The most similar command is
        fsck

DELL@DESKTOP-AF8EFP0 MINGW64 ~/OneDrive - Amrita Vishwa Vidyapeetham- Chennai Campus/Desktop (master)
$ git --help
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

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
```

14. git fork



```
Daev@DESKTOP-GM0E4B1 MINGW64 ~/Cognizance (master)
$ git fork
git: 'fork' is not a git command. See 'git --help'.

The most similar command is
        fsck

Daev@DESKTOP-GM0E4B1 MINGW64 ~/Cognizance (master)
$ git --help
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

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.
```



```
Daev@DESKTOP-GM0E4B1 MINGW64 ~/Cognizance (master)
$ git config --list --show-
error: ambiguous option: show- (could be --show-origin or --show-scope)
usage: git config [<options>]

Config file location
    --global              use global config file
    --system              use system config file
    --local               use repository config file
    --worktree            use per-worktree config file
    -f, --file <file>     use given config file
    --blob <blob-id>      read config from given blob object

Action
    --get                 get value: name [value-pattern]
    --get-all             get all values: key [value-pattern]
    --get-regexp          get values for regexp: name-regex [value-pattern]
    --get-urlmatch        get value specific for the URL: section[.var] URL
    --replace-all         replace all matching variables: name value [value-pattern]
    --add                 add a new variable: name value
    --unset               remove a variable: name [value-pattern]
    --unset-all           remove all matches: name [value-pattern]
    --rename-section      rename section: old-name new-name
    --remove-section      remove a section: name
    -l, --list            list all
    --fixed-value         use string equality when comparing values to 'value-pattern'
    -e, --edit            open an editor
    --get-color           find the color configured: slot [default]
    --get-colorbool       find the color setting: slot [stdout-is-tty]

Type
    -t, --type <>         value is given this type
    --bool                value is "true" or "false"
    --int                 value is decimal number
    --bool-or-int         value is --bool or --int
    --bool-or-str         value is --bool or string
    --path                value is a path (file or directory name)
    --expiry-date         value is an expiry date

Other
    -z, --null            terminate values with NUL byte
    --name-only           show variable names only
    --includes            respect include directives on lookup
    --show-origin         show origin of config (file, standard input, blob, command line)
    --show-scope          show scope of config (worktree, local, global, system, command)
    --default <value>     with --get, use default value when missing entry
```

BY RITTIN MITHRA.C

CH.EN.U4CYS21067