EXPERIMENT NO.3

AIM:

To understand Cmd version control commands

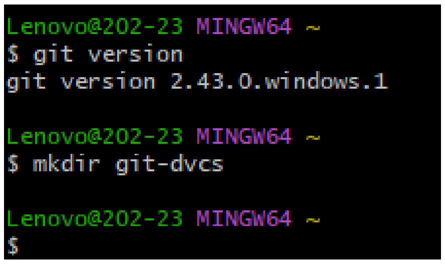
THEORY:

Introduction to Cmd Version Control Commands:

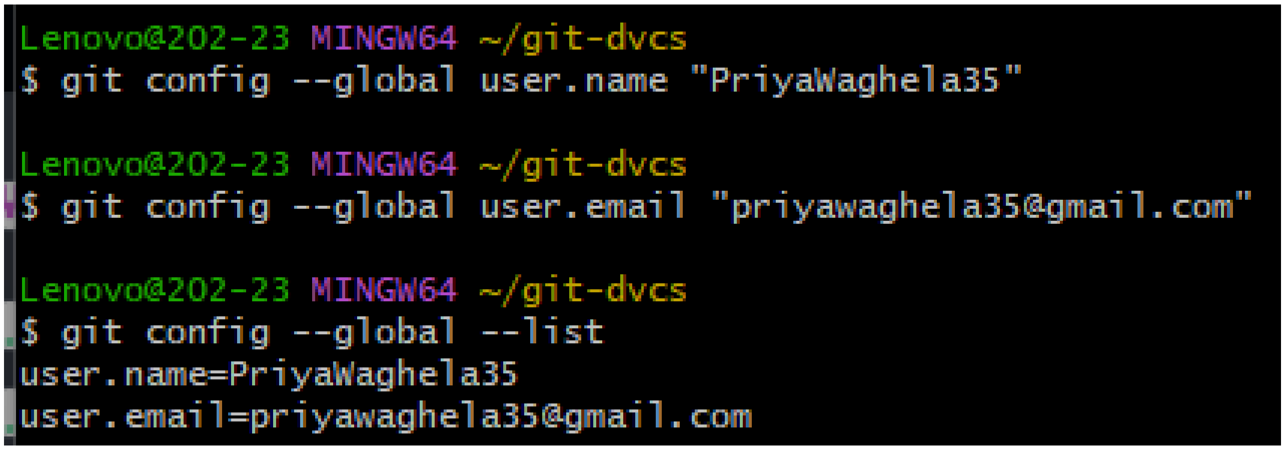
Command-line version control commands, commonly associated with tools like Git, play a pivotal role in efficiently managing and tracking changes within software development projects. These commands provide developers with a powerful set of tools for versioning, collaboration, and project management.

1. Initialization and Configuration:

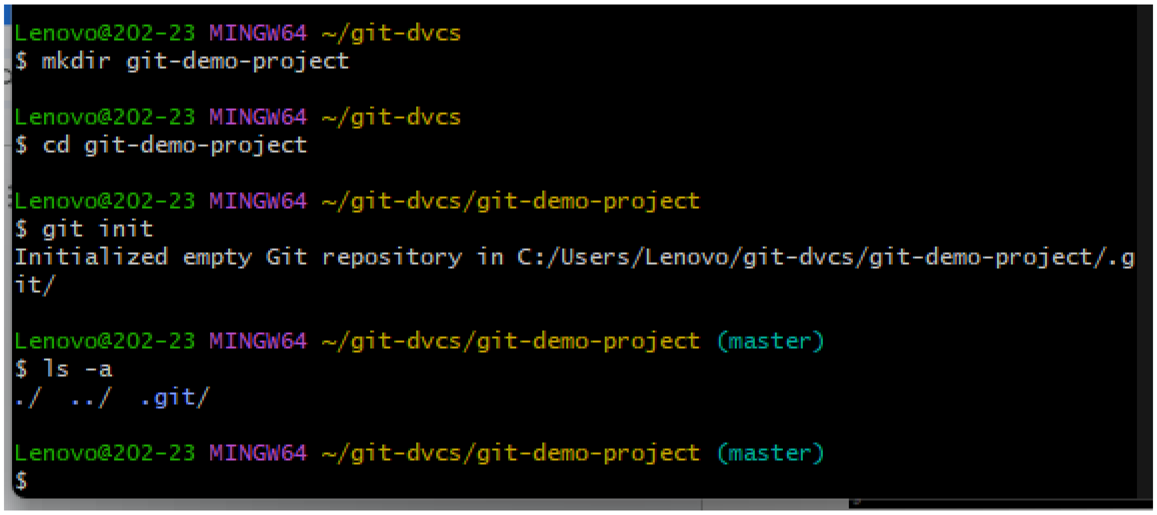
To commence version control, developers initialize a repository using the git init command. Configuration commands such as git config allow users to set global or repository-specific settings, including user information and default behaviors.



```
Lenovo@202-23 MINGW64 ~
$ git version
git version 2.43.0.windows.1

Lenovo@202-23 MINGW64 ~
$ mkdir git-dvcs

Lenovo@202-23 MINGW64 ~
$
```



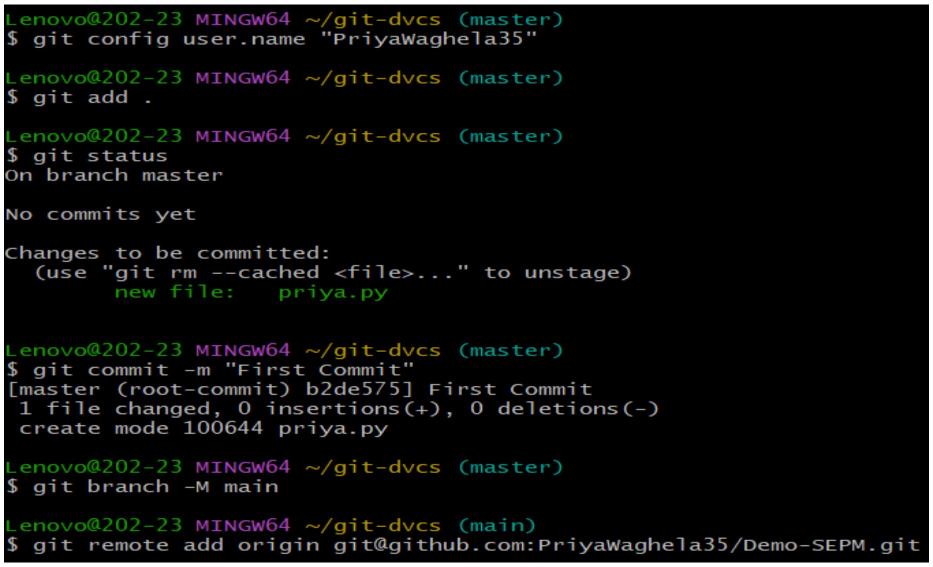
```
Lenovo@202-23 MINGW64 ~/git-dvcs
$ git config --global user.name "PriyaWaghela35"

Lenovo@202-23 MINGW64 ~/git-dvcs
$ git config --global user.email "priyawaghela35@gmail.com"

Lenovo@202-23 MINGW64 ~/git-dvcs
$ git config --global --list
user.name=PriyaWaghela35
user.email=priyawaghela35@gmail.com
```



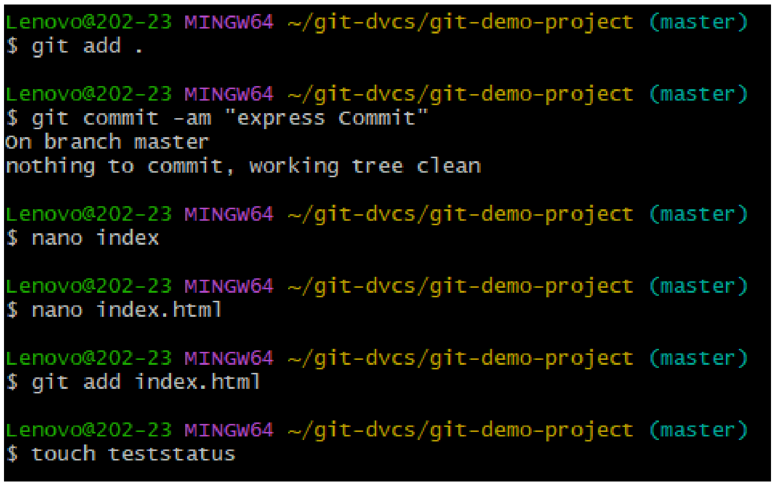
```
Lenovo@202-23 MINGW64 ~/git-dvcs
$ mkdir git-demo-project

Lenovo@202-23 MINGW64 ~/git-dvcs
$ cd git-demo-project

Lenovo@202-23 MINGW64 ~/git-dvcs/git-demo-project
$ git init
Initialized empty Git repository in C:/Users/Lenovo/git-dvcs/git-demo-project/.g
it/

Lenovo@202-23 MINGW64 ~/git-dvcs/git-demo-project (master)
$ ls -a
./  ../  .git/

Lenovo@202-23 MINGW64 ~/git-dvcs/git-demo-project (master)
$
```

2. Basic Workflow Commands:

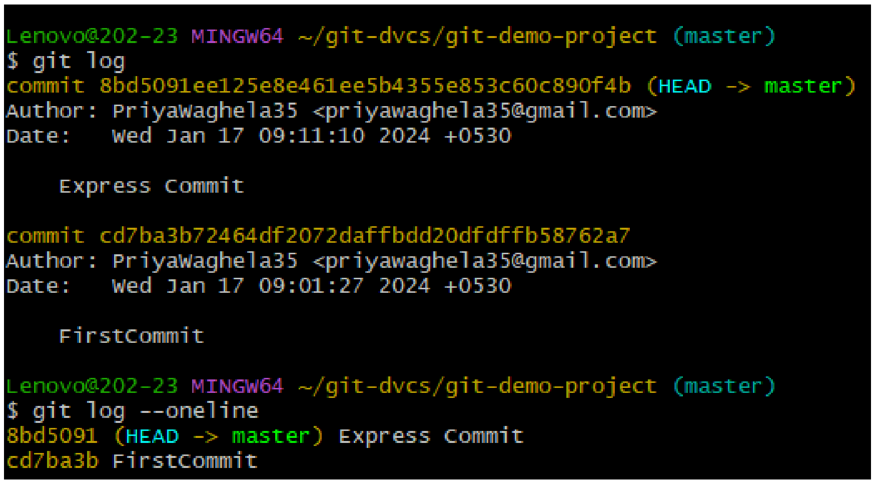
- git add: This command stages changes for commit, allowing developers to selectively include modifications in the next snapshot.
- git commit: Commits the staged changes, creating a snapshot of the project's state with a descriptive message.
- git status: Provides information about the current state of the working directory, highlighting modified, staged, and untracked files.
- git log: Displays a chronological history of commits, including commit messages, authors, and timestam



```
Lenovo@202-23 MINGW64 ~/git-dvcs (master)
$ git config user.name "PriyaWaghela35"

Lenovo@202-23 MINGW64 ~/git-dvcs (master)
$ git add .

Lenovo@202-23 MINGW64 ~/git-dvcs (master)
$ git status
On branch master

No commits yet

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


Lenovo@202-23 MINGW64 ~/git-dvcs (master)
$ git commit -m "First Commit"
[master (root-commit) b2de575] First Commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 priya.py

Lenovo@202-23 MINGW64 ~/git-dvcs (master)
$ git branch -M main

Lenovo@202-23 MINGW64 ~/git-dvcs (main)
$ git remote add origin git@github.com:PriyaWaghela35/Demo-SEPM.git
```



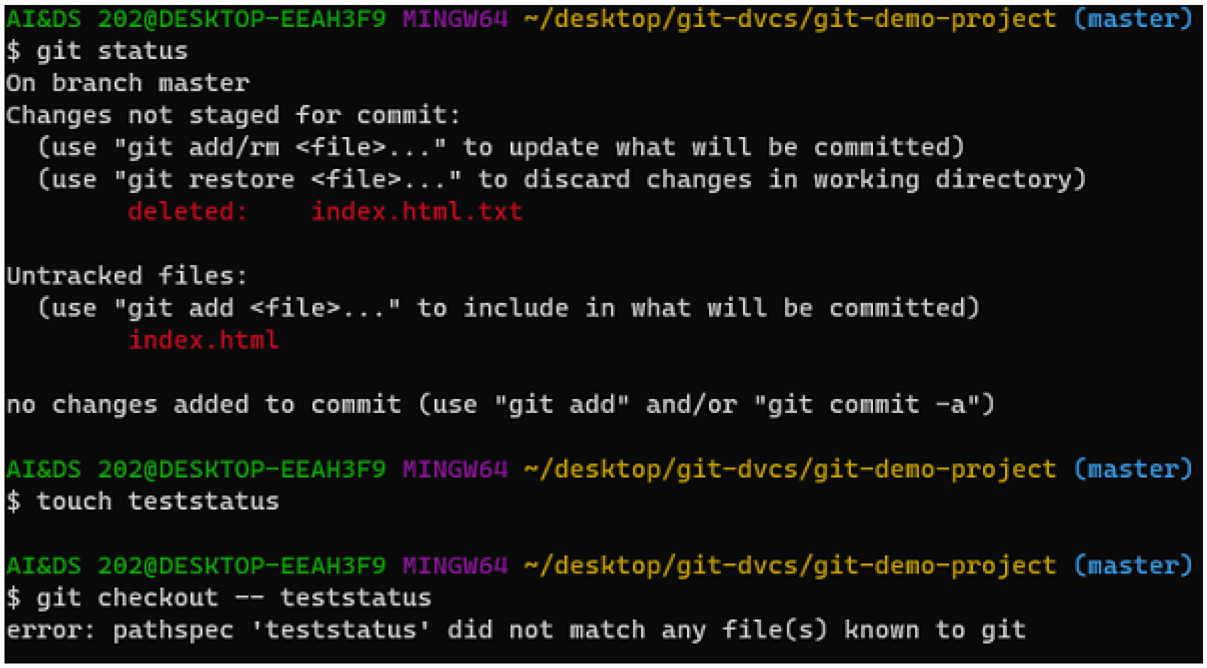
```
Lenovo@202-23 MINGW64 ~/git-dvcs/git-demo-project (master)
$ git add .

Lenovo@202-23 MINGW64 ~/git-dvcs/git-demo-project (master)
$ git commit -am "express Commit"
On branch master
nothing to commit, working tree clean

Lenovo@202-23 MINGW64 ~/git-dvcs/git-demo-project (master)
$ nano index

Lenovo@202-23 MINGW64 ~/git-dvcs/git-demo-project (master)
$ nano index.html

Lenovo@202-23 MINGW64 ~/git-dvcs/git-demo-project (master)
$ git add index.html

Lenovo@202-23 MINGW64 ~/git-dvcs/git-demo-project (master)
$ touch teststatus
```



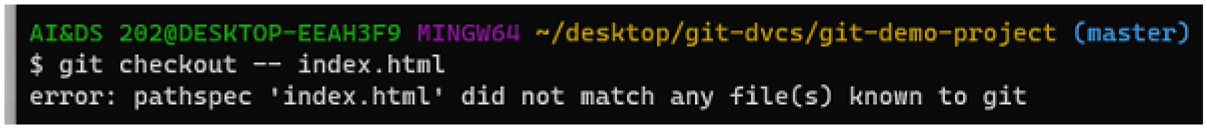
```
Lenovo@202-23 MINGW64 ~/git-dvcs/git-demo-project (master)
$ git log
commit 8bd5091ee125e8e461ee5b4355e853c60c890f4b (HEAD -> master)
Author: PriyaWaghela35 <priyawaghela35@gmail.com>
Date:   Wed Jan 17 09:11:10 2024 +0530

    Express Commit

commit cd7ba3b72464df2072daffbdd20dfdffb58762a7
Author: PriyaWaghela35 <priyawaghela35@gmail.com>
Date:   Wed Jan 17 09:01:27 2024 +0530

    FirstCommit

Lenovo@202-23 MINGW64 ~/git-dvcs/git-demo-project (master)
$ git log --oneline
8bd5091 (HEAD -> master) Express Commit
cd7ba3b FirstCommit
```

3. Branching and Merging:

- git branch: Lists, creates, or deletes branches within the repository.
- git checkout: Switches between branches or commits, allowing developers to navigate through the project's history.
- git merge: Combines changes from different branches, integrating modifications into a common branch.



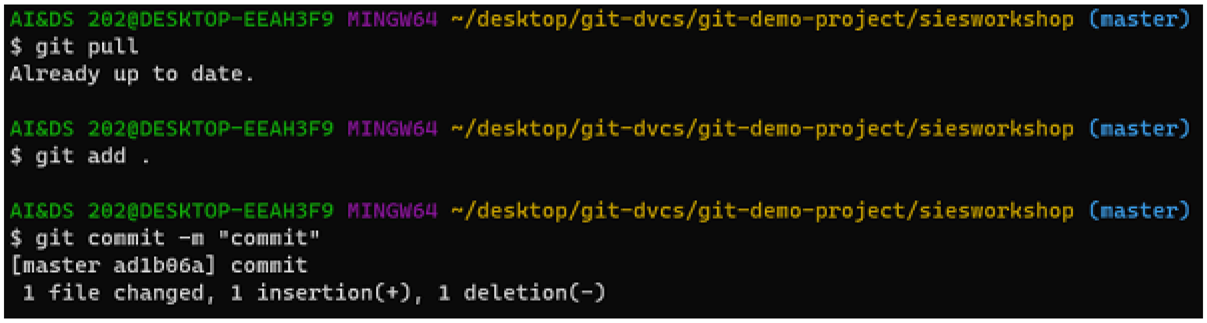
```
AI&DS 202@DESKTOP-EEAH3F9 MINGW64 ~/desktop/git-dvcs/git-demo-project (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    index.html.txt

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

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

AI&DS 202@DESKTOP-EEAH3F9 MINGW64 ~/desktop/git-dvcs/git-demo-project (master)
$ touch teststatus

AI&DS 202@DESKTOP-EEAH3F9 MINGW64 ~/desktop/git-dvcs/git-demo-project (master)
$ git checkout -- teststatus
error: pathspec 'teststatus' did not match any file(s) known to git
```



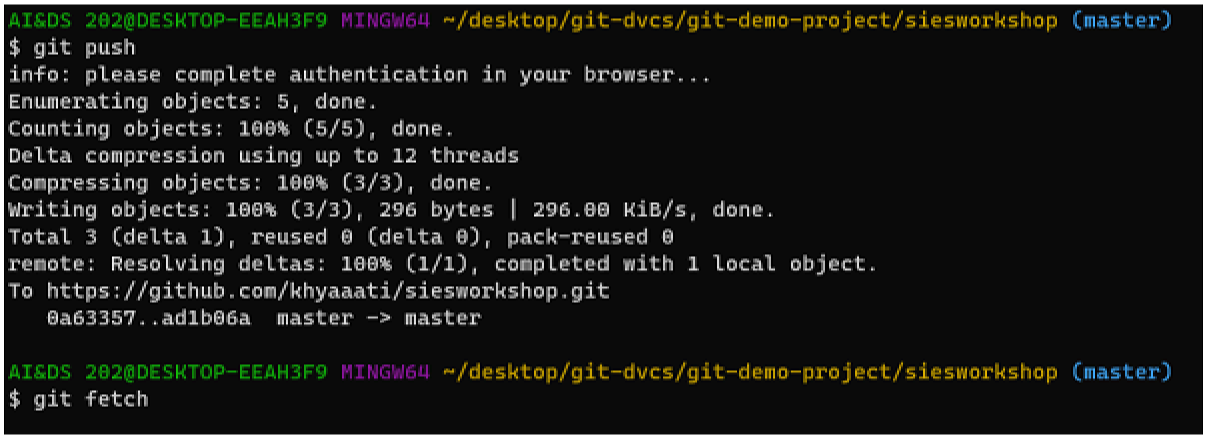
```
AI&DS 202@DESKTOP-EEAH3F9 MINGW64 ~/desktop/git-dvcs/git-demo-project (master)
$ git checkout -- index.html
error: pathspec 'index.html' did not match any file(s) known to git
```

4. Remote Repositories and Collaboration:

- git remote: Manages connections to remote repositories, such as GitHub or GitLab.
- git fetch: Retrieves changes from a remote repository, updating the local repository without modifying the working directory.
- git pull: Fetches changes and merges them into the current branch.
- git push: Uploads local commits to a remote repository.



```
AI&DS 202@DESKTOP-EEAH3F9 MINGW64 ~/desktop/git-dvcs/git-demo-project/siesworkshop (master)
$ git pull
Already up to date.

AI&DS 202@DESKTOP-EEAH3F9 MINGW64 ~/desktop/git-dvcs/git-demo-project/siesworkshop (master)
$ git add .

AI&DS 202@DESKTOP-EEAH3F9 MINGW64 ~/desktop/git-dvcs/git-demo-project/siesworkshop (master)
$ git commit -m "commit"
[master ad1b06a] commit
 1 file changed, 1 insertion(+), 1 deletion(-)
```



```
AI&DS 202@DESKTOP-EEAH3F9 MINGW64 ~/desktop/git-dvcs/git-demo-project/siesworkshop (master)
$ git push
info: please complete authentication in your browser...
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 296 bytes | 296.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/khyaaati/siesworkshop.git
   0a63357..ad1b06a  master -> master

AI&DS 202@DESKTOP-EEAH3F9 MINGW64 ~/desktop/git-dvcs/git-demo-project/siesworkshop (master)
$ git fetch
```

5. Undoing Changes:

- git reset: Resets the repository to a specific commit, optionally preserving changes in the working directory.
- git revert: Creates a new commit that undoes specific changes introduced by previous commits.
- git stash: Temporarily saves changes in a stack, allowing developers to switch branches without committing incomplete work.

CONCLUSION:

In summary, exploring Command Line (Cmd) version control commands provides valuable insights into efficiently managing and tracking changes in software projects. Command-line tools, such as Git, empower users to initiate repositories, stage changes, commit modifications, and collaborate seamlessly. Understanding these commands is essential for developers, enabling precise control over versioning and facilitating collaboration through platforms like GitHub. Mastering Cmd version control commands enhances a developer's ability to navigate and contribute to projects, fostering a streamlined and collaborative approach to software development.