EXPERIMENT No. 01: Setting Up and Basic Commands

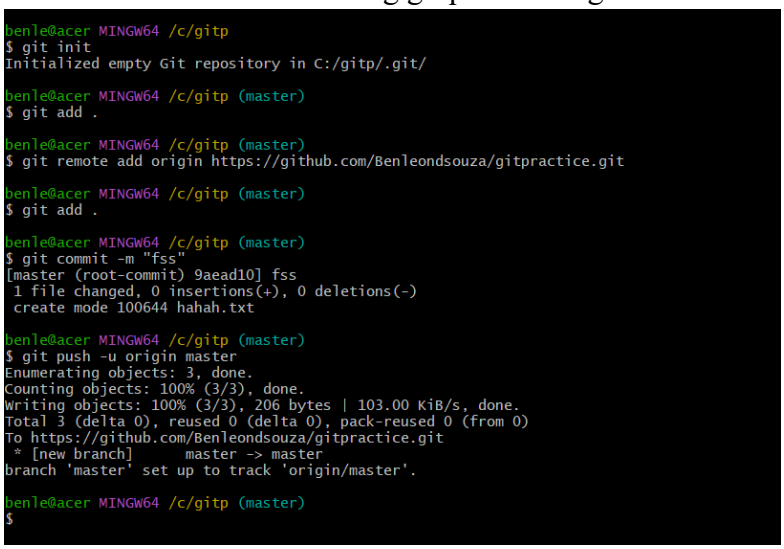
AIM:
Initialize a Git repository, create files, stage and commit changes, link with a remote repository, and push the changes.

COMMANDS:

1. git init
2. git add .
3. git commit -m "Initial"
4. git remote add origin <repo-link>
5. git remote -v
6. git push -u origin main

PROCEDURE AND RESULTS:

- Create a new repository on GitHub (just give the repo name and create it).
- Copy the repo link.
- Create a new folder locally and open it in VS Code.
- Inside the folder, create any file and add some content.
- Open the terminal and initialize Git using git init.
- Stage the file using git add ..
- Commit the changes using git commit -m "Initial".
- Add the remote origin using git remote add origin <repo-link>.
- Confirm the remote using git remote -v.
- Push the code to GitHub using git push -u origin main.



EXPERIMENT No. 02: Creating and Managing Branches

AIM:
Create and manage branches using Git. Perform merge operations and use stash to temporarily save and apply changes.

COMMANDS:

Branch Creation and Merge:

1. git branch feature-branch
2. git checkout feature-branch
3. git add .
4. git commit -m "Initial"
5. git checkout main
6. git merge feature-branch
7. git push

Stash Operation:

1. git stash
2. git checkout main
3. git stash apply

PROCEDURE AND RESULTS:

Branch Creation and Merge:

- Create a new branch using git branch feature-branch.
- Switch to the new branch with git checkout feature-branch.
- Make changes (e.g., create a file), then stage and commit using git add . and git commit -m "Initial".
- Switch back to the main branch using git checkout main.
- Merge the feature branch using git merge feature-branch.
- Push the updated main branch to GitHub using git push.

Stash Operation:

- While on the feature-branch, make some changes without committing.
- Use git stash to temporarily save the changes.
- Switch to the main branch using git checkout main.

- After performing other operations or creating files, run git stash apply to re-apply the stashed changes to the working directory.

```
benle@acer MINGW64 /c/gitp (master)
$ git branch feature-branch

benle@acer MINGW64 /c/gitp (master)
$ git checkout feature-branch
Switched to branch 'feature-branch'

benle@acer MINGW64 /c/gitp (feature-branch)
$ git add .

benle@acer MINGW64 /c/gitp (feature-branch)
$ git commit -m "ha2"
[feature-branch bb18cae] ha2
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 sdhbcibci.txt

benle@acer MINGW64 /c/gitp (feature-branch)
$ git checkout main
error: pathspec 'main' did not match any file(s) known to git

benle@acer MINGW64 /c/gitp (feature-branch)
$ ^C

benle@acer MINGW64 /c/gitp (feature-branch)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

benle@acer MINGW64 /c/gitp (master)
$ git merge feature-branch
Updating 9aead10..bb18cae
Fast-forward
 sdhbcibci.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 sdhbcibci.txt

benle@acer MINGW64 /c/gitp (master)
$ git push
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 238 bytes | 238.00 KiB/s, done.
Total 2 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Benleondsouza/gitpractice.git
   9aead10..bb18cae  master -> master
```

```
benle@acer MINGW64 /c/gitp (master)
$ git checkout feature-branch
Switched to branch 'feature-branch'
M       sdhbcibci.txt

benle@acer MINGW64 /c/gitp (feature-branch)
$ git stash
Saved working directory and index state WIP on feature-branch: bb18cae ha2

benle@acer MINGW64 /c/gitp (feature-branch)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

benle@acer MINGW64 /c/gitp (master)
$ git stash apply
On branch master
Your branch is up to date with 'origin/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:   hahah.txt
        modified:   sdhbcibci.txt
```

EXPERIMENT No. 03: Collaboration and Remote Repositories

AIM:
Clone remote repositories and manage updates with fetch, rebase, and merge.

COMMANDS:

1. git clone <link>
2. git fetch
3. git rebase origin/main
4. git merge feature-branch --no-ff -m "Merge"

PROCEDURE AND RESULTS:

- Clone an existing GitHub repository using git clone <link>.
- Fetch new updates from the remote using git fetch.
- Rebase the local branch using git rebase origin/main.
- Merge the feature branch with a message using git merge feature-branch --no-ff -m "Merge".

```
benle@acer MINGW64 /c/gitp2
$ git clone https://github.com/Benleondsouza/gitpractice.git
Cloning into 'gitpractice'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 5 (delta 0), reused 5 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (5/5), done.

benle@acer MINGW64 /c/gitp2
$ git fetch
```

```
benle@acer MINGW64 /c/gitp2/gitpractice (feature-branch)
$ git fetch

benle@acer MINGW64 /c/gitp2/gitpractice (feature-branch)
$ git rebase origin/main
fatal: invalid upstream 'origin/main'

benle@acer MINGW64 /c/gitp2/gitpractice (feature-branch)
$ git rebase origin/master
Current branch feature-branch is up to date.

benle@acer MINGW64 /c/gitp2/gitpractice (feature-branch)
$ git checkout main
error: pathspec 'main' did not match any file(s) known to git

benle@acer MINGW64 /c/gitp2/gitpractice (feature-branch)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

benle@acer MINGW64 /c/gitp2/gitpractice (master)
$ git merge feature-branch --no-ff -m "Merge"
Merge made by the 'ort' strategy.
 wefw.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 wefw.txt

benle@acer MINGW64 /c/gitp2/gitpractice (master)
$ git push origin master
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 301 bytes | 301.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To https://github.com/Benleondsouza/gitpractice.git
   bb18cae..11943f2  master -> master
```

EXPERIMENT No. 04: Git Tags and Releases

AIM:

Use Git tags to mark versions and create GitHub releases.

COMMANDS:

1. git tag v1.0

PROCEDURE AND RESULTS:

- Create a local tag using git tag v1.0.
- Go to the GitHub repository.
- Click on **Releases > Create a new release**.
- Set the tag version as v1.0 and give a release title.

```
benle@acer MINGW64 /c/gitp2/gitpractice (master)
$ git log
commit 11943f29a1b8093c32988c75e812276a0e84e1ed (HEAD -> master, tag: v1.0, origin/master, origin/HEAD)
Merge: bb18cae 724ef74
Author: ben10 <benleondsouza2@gmail.com>
Date:   Tue May 20 21:50:18 2025 +0530

    Merge

commit 724ef7411dc597b0aa4879209ea2b6487969f0a2 (feature-branch)
Author: ben10 <benleondsouza2@gmail.com>
Date:   Tue May 20 21:47:51 2025 +0530

    fs

commit bb18cae158f39a23973400cf29b7190950bb581b
Author: ben10 <benleondsouza2@gmail.com>
Date:   Tue May 20 21:25:09 2025 +0530

    ha2

commit 9aead107b3a1256680f33686ece72b94598338b4
Author: ben10 <benleondsouza2@gmail.com>
Date:   Tue May 20 21:12:02 2025 +0530

    fss

benle@acer MINGW64 /c/gitp2/gitpractice (master)
$ git tag v1.0
fatal: tag 'v1.0' already exists

benle@acer MINGW64 /c/gitp2/gitpractice (master)
$ git show v1.0
commit 11943f29a1b8093c32988c75e812276a0e84e1ed (HEAD -> master, tag: v1.0, origin/master, origin/HEAD)
Merge: bb18cae 724ef74
Author: ben10 <benleondsouza2@gmail.com>
Date:   Tue May 20 21:50:18 2025 +0530

    Merge


benle@acer MINGW64 /c/gitp2/gitpractice (master)
$ git push origin v1.0
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Benleondsouza/gitpractice.git
 * [new tag]         v1.0 -> v1.0
```

EXPERIMENT No. 05: Advanced Git Operations – Cherry Pick

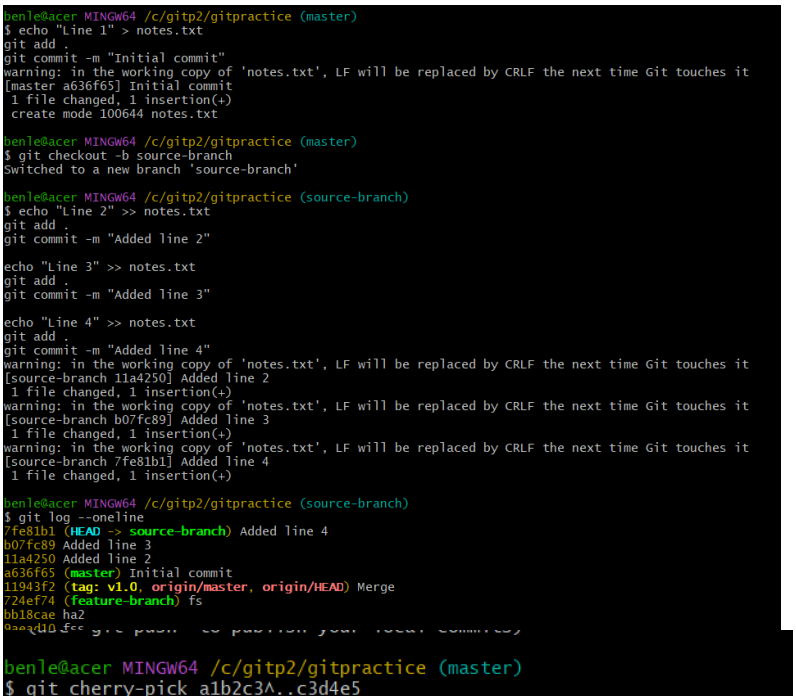
AIM:
Use cherry-pick to apply specific commits from one branch to another.

COMMANDS:

1. git log --oneline
2. git cherry-pick <commit-id>

PROCEDURE AND RESULTS:

- On the feature-branch, create a file and commit changes.
- Use git log --oneline to get the commit ID.
- Switch to the main branch using git checkout main.
- Use git cherry-pick <commit-id> to apply a specific commit.



EXPERIMENT No. 06: Analysing and Changing Git History

AIM:
Analyze commit logs and revert specific changes.

COMMANDS:

1. git show <commit-id>
2. git log --author="Varsha" --after="YYYY-MM-DD" --before="YYYY-MM-DD"
3. git log -n 5
4. git revert <commit-id>

PROCEDURE AND RESULTS:

- Use git show <commit-id> to view commit details.
- Use author and date filters to check commit history.
- View last five commits using git log -n 5.
- Revert a specific commit using git revert <commit-id>.