Experiment No 01: Setting Up and Basic Commands

1.1 Objective

Initialize a new Git repository in a directory. Create a new file and add it to the staging area and commit the changes with an appropriate commit message

1.2 System Configuration

Windows 10

1.3 Pre-Requisites:

Install Git bash and make required settings.

1.4: Introduction:

git init

This command is used to initialize a new Git repository in the current directory. It sets up the necessary data structures and files that Git needs to start tracking changes to your project.

git clone 'remote repository link'

This command is used to create a copy of an existing Git repository from a remote location (in this case, from the specified URL) to your local machine. It not only copies the files but also sets up a connection to the original repository so you can pull in updates later.

git add .

This command is used to stage all changes in the current directory for the next commit. Staging is the process of preparing files to be included in the next commit. The dot (.) indicates that all changes, including new files, modified files, and deleted files, should be staged.

git commit -m "<message>"

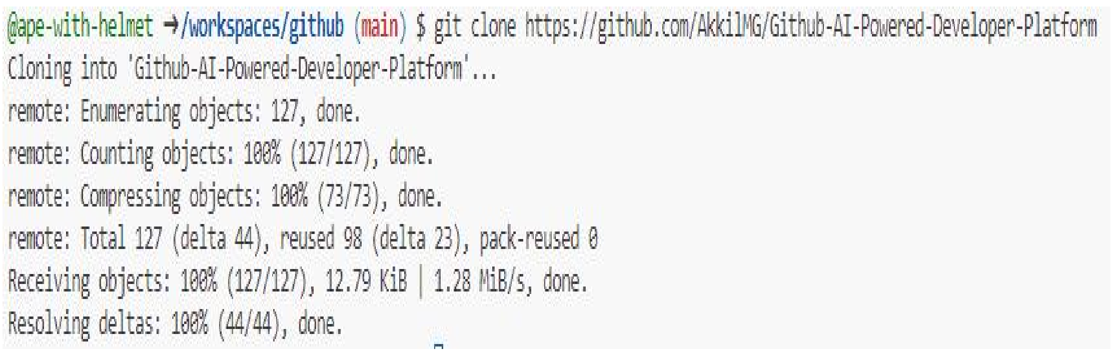
This command is used to record the changes that have been staged (using git add) in the repository. The -m flag allows you to include a short message that describes the changes made in this commit. It's good practice to write clear and concise commit messages that explain the purpose of the changes.

git push

This command is used to upload local repository content to a remote repository. In the context of GitHub, it typically sends committed changes from your local repository to the remote repository on GitHub. This allows others to see the changes you've made and collaborate with you. Depending on your Git configuration, you may need to specify the remote repository and branch name, but if you've cloned a repository, Git usually sets up the default remote and branch for you.

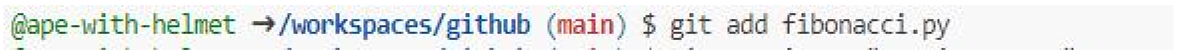
1.5 Procedure and Result:

1.git clone ‘remote repository link’



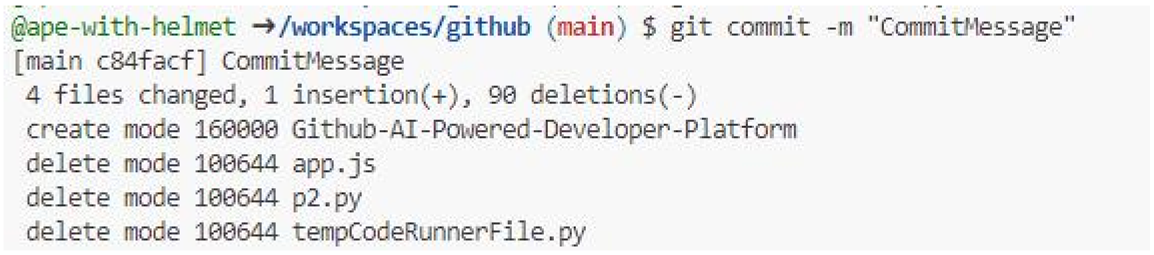
```
@ape-with-helmet ➜ /workspaces/github (main) $ git clone https://github.com/AkkilMG/Github-AI-Powered-Developer-Platform
Cloning into 'Github-AI-Powered-Developer-Platform'...
remote: Enumerating objects: 127, done.
remote: Counting objects: 100% (127/127), done.
remote: Compressing objects: 100% (73/73), done.
remote: Total 127 (delta 44), reused 98 (delta 23), pack-reused 0
Receiving objects: 100% (127/127), 12.79 KiB | 1.28 MiB/s, done.
Resolving deltas: 100% (44/44), done.
```

2. git add .



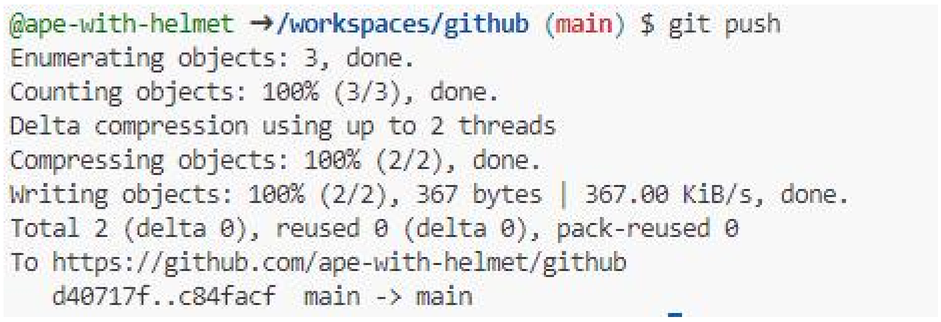
```
@ape-with-helmet →/workspaces/github (main) $ git add fibonacci.py
```

3. git commit -m "<message>"



```
@ape-with-helmet →/workspaces/github (main) $ git commit -m "CommitMessage"
[main c84facf] CommitMessage
 4 files changed, 1 insertion(+), 90 deletions(-)
 create mode 160000 Github-AI-Powered-Developer-Platform
 delete mode 100644 app.js
 delete mode 100644 p2.py
 delete mode 100644 tempCodeRunnerFile.py
```

4. git push



```
@ape-with-helmet →/workspaces/github (main) $ git push
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 367 bytes | 367.00 KiB/s, done.
Total 2 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/ape-with-helmet/github
   d40717f..c84facf  main -> main
```

Experiment No 02: Creating and Managing Branches

2.1Objective

a) Create a new branch named “feature-branch”. Switch to the “master” branch. Merge the “feature-branch” into “master”.

b) Write the commands to stash your changes, switch branches, and then apply the stashed changes.

2.2 System Configuration

Windows 10

2.3 Pre-Requisites:

1. Git is installed on your system.
2. You have a Git repository initialized and have some changes made to the files.
3. You have at least two branches: **master/main** and **feature-branch**.

2.3 Introduction:

git branch feature-branch

This command creates a new branch named "feature-branch" but doesn't switch to it.

git checkout master/main or

git switch main

This command switches to the "master" branch.

git merge feature-branch

This command merges changes from "feature-branch" into the "master" branch

git stash:

Stash your changes

git checkout feature-branch

Switch branches (example: from "master" to "feature-branch")

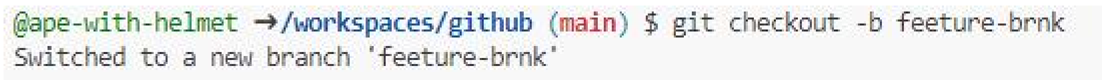
git stash apply

Apply the stashed changes

2.5 Procedure and Result:

git branch feature-branch

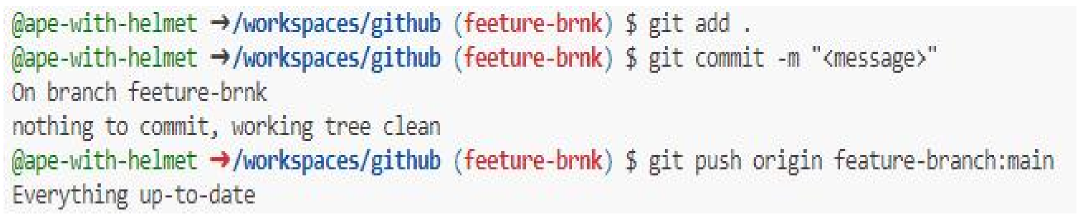
git checkout feature-branch



```
@ape-with-helmet →/workspaces/github (main) $ git checkout -b feeture-brnk
Switched to a new branch 'feeture-brnk'
```

git branch master

git checkout master



```
@ape-with-helmet →/workspaces/github (feeture-brnk) $ git add .
@ape-with-helmet →/workspaces/github (feeture-brnk) $ git commit -m "<message>"
On branch feeture-brnk
nothing to commit, working tree clean
@ape-with-helmet →/workspaces/github (feeture-brnk) $ git push origin feature-branch:main
Everything up-to-date
```

Experiment No 03: Collaboration and Remote Repositories

3.1 Objectives:

a) Clone a remote Git repository to your local machine.

b) Fetch the latest changes from a remote repository and rebase your local branch onto the updated remote branch.

c) Write a command to merge “feature-branch” into “master” while providing a custom commit message for the merge.

3.2 System Configuration:

Windows 10

3.3 Pre requisite:

- Git bash should be installed
- Visual studio code editor should be installed

3.4 Introduction:

a) Clone a remote Git repository to your local machine:

   git clone <repository_url>

   This command clones a remote Git repository onto your local machine, creating a new directory with the same name as the repository.

b) Fetch the latest changes from a remote repository and rebase your local branch onto the updated remote branch:

   git checkout -b feature-branch

   git fetch origin

   git rebase origin/<branch_name>

   The first command (git fetch origin) fetches the latest changes from the remote repository named "origin" without merging them into your local branches. The second command (git rebase origin/<branch_name>) rebases your current local branch onto the updated remote branch, integrating the latest changes from the remote repository while maintaining the commit history.

c) Write a command to merge "feature-branch" into "master" while providing a custom commit message for the merge:

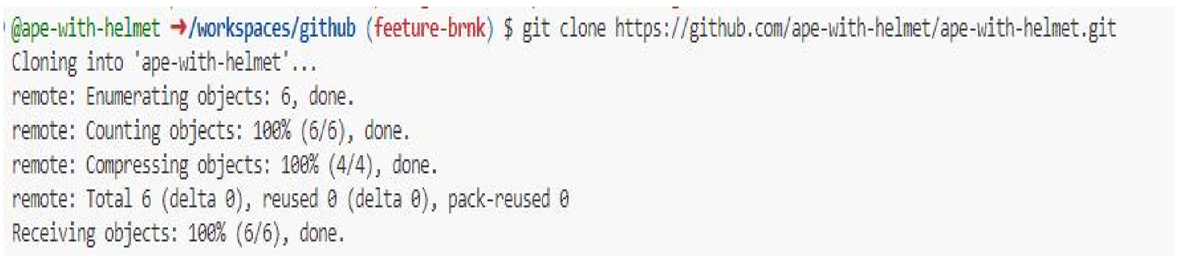
git checkout master

git merge --no-ff feature-branch -m "Custom commit message"

The first command (git checkout master) switches to the "master" branch. The second command (git merge --no-ff feature-branch -m "Custom commit message") merges the changes from the "feature-branch" into the "master" branch, creating a merge commit with the provided custom commit message. The --no-ff option ensures that a merge commit is always created, even if the merge could be performed with a fast-forward.

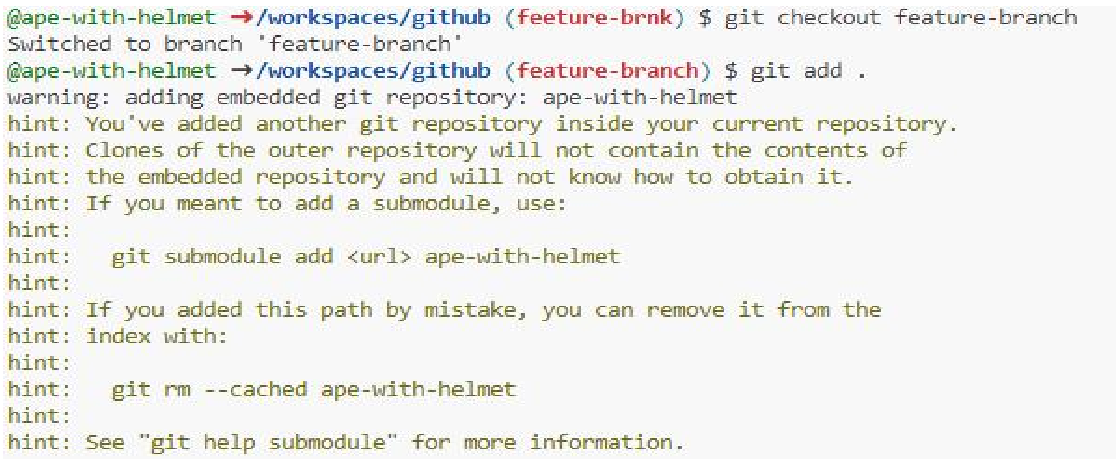
3.5 Procedure & Results:

git clone <repository_url>



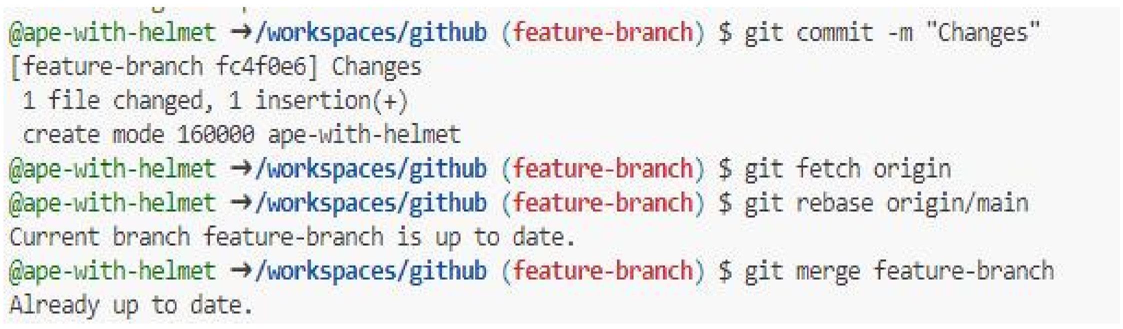
```
@ape-with-helmet →/workspaces/github (feeture-brnk) $ git clone https://github.com/ape-with-helmet/ape-with-helmet.git
Cloning into 'ape-with-helmet'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (6/6), done.
```

git checkout -b feature-branch



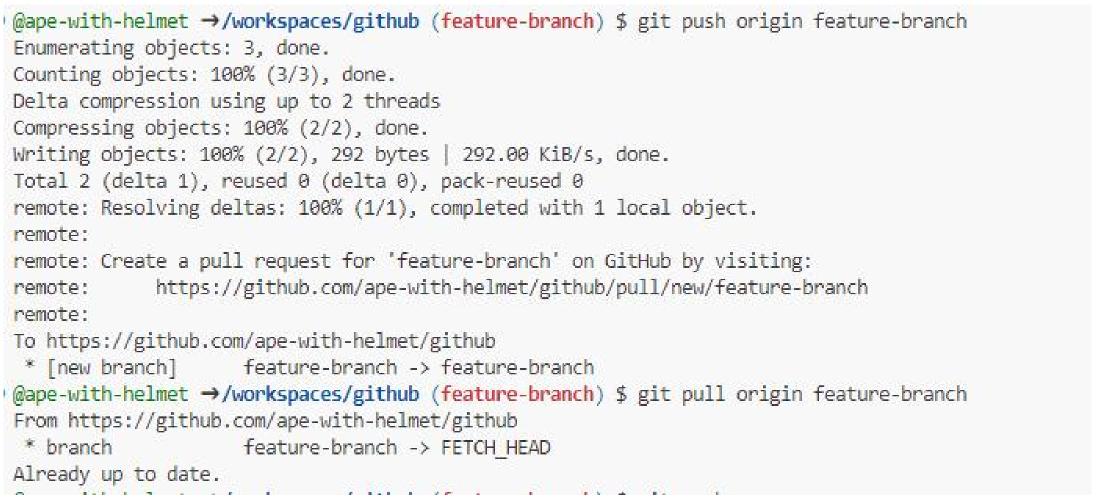
```
@ape-with-helmet →/workspaces/github (feeture-brnk) $ git checkout feature-branch
Switched to branch 'feature-branch'
@ape-with-helmet →/workspaces/github (feature-branch) $ git add .
warning: adding embedded git repository: ape-with-helmet
hint: You've added another git repository inside your current repository.
hint: Clones of the outer repository will not contain the contents of
hint: the embedded repository and will not know how to obtain it.
hint: If you meant to add a submodule, use:
hint:
hint:   git submodule add <url> ape-with-helmet
hint:
hint: If you added this path by mistake, you can remove it from the
hint: index with:
hint:
hint:   git rm --cached ape-with-helmet
hint:
hint: See "git help submodule" for more information.
```

git fetch origin



```
@ape-with-helmet →/workspaces/github (feature-branch) $ git commit -m "Changes"
[feature-branch fc4f0e6] Changes
 1 file changed, 1 insertion(+)
 create mode 160000 ape-with-helmet
@ape-with-helmet →/workspaces/github (feature-branch) $ git fetch origin
@ape-with-helmet →/workspaces/github (feature-branch) $ git rebase origin/main
Current branch feature-branch is up to date.
@ape-with-helmet →/workspaces/github (feature-branch) $ git merge feature-branch
Already up to date.
```

git rebase origin/<branch_name>



```
@ape-with-helmet →/workspaces/github (feature-branch) $ git push origin feature-branch
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 292 bytes | 292.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'feature-branch' on GitHub by visiting:
remote:      https://github.com/ape-with-helmet/github/pull/new/feature-branch
remote:
To https://github.com/ape-with-helmet/github
 * [new branch]      feature-branch -> feature-branch
@ape-with-helmet →/workspaces/github (feature-branch) $ git pull origin feature-branch
From https://github.com/ape-with-helmet/github
 * branch            feature-branch -> FETCH_HEAD
Already up to date.
```

Experiment No 04: Git Tags and Releases

4.1Objective

Write the command to create a lightweight Git tag named "v1.0" for a commit in your local repository.

4.2 System Configuration

Windows 10

4.3 Pre-Requisites:

Install Git bash and make required settings.

4.4: Introduction:

git log
to view the commit history and find the commit ID you want to tag. Each commit is identified by a unique hash.

git tag v1.0 <commit_id>
git tag followed by the tag name (e.g., "v1.0") and the commit ID to create a lightweight tag. Lightweight tags are simply pointers to specific commits and do not contain additional metadata like annotated tags.

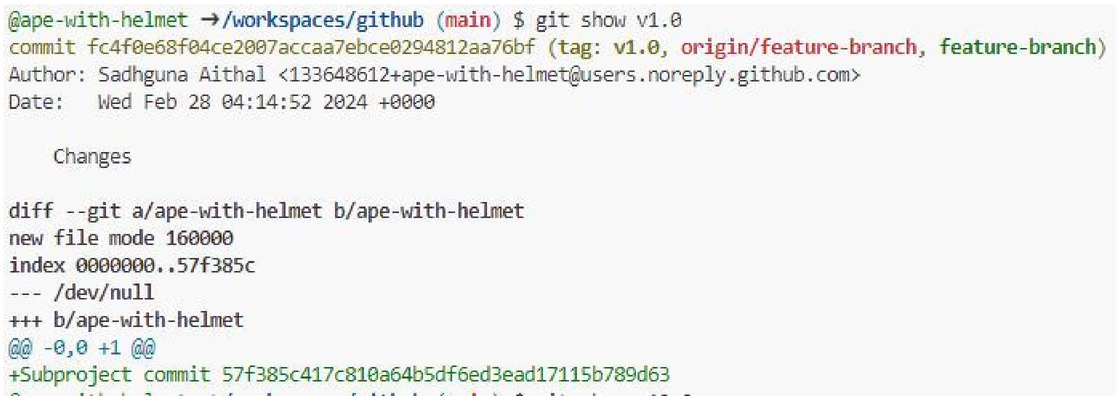
git show v1.0
Use git show followed by the tag name to verify that the tag was created correctly and is pointing to the desired commit. This command will display the details of the tag, including the commit it points to.

git push origin v1.0

If you want to share the tag with others, you can push it to a remote repository using git push. This step is optional and depends on your workflow and the need to share the tag with others..

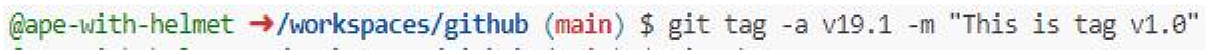
4.5 Procedure and Result:

git log



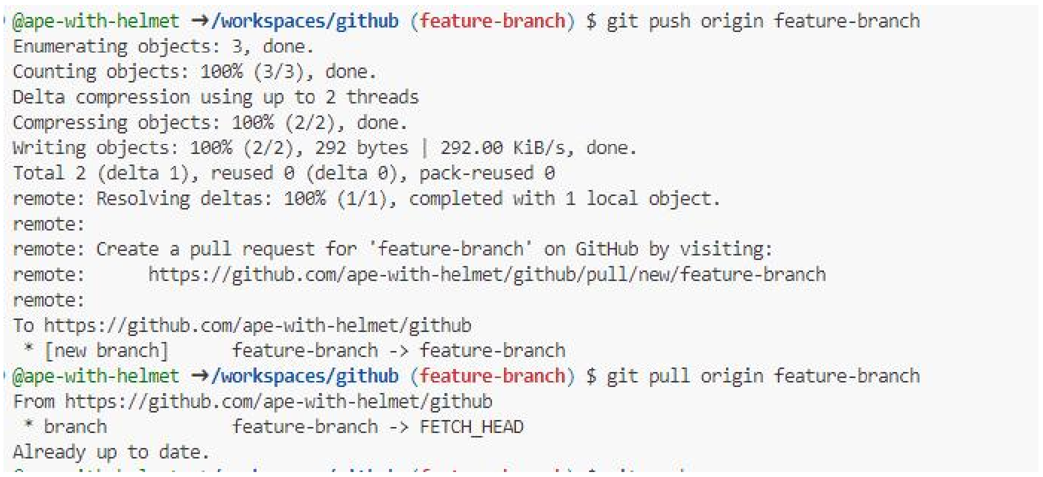
```
@ape-with-helmet ➜/workspaces/github (main) $ git show v1.0
commit fc4f0e68f04ce2007accaa7ebce0294812aa76bf (tag: v1.0, origin/feature-branch, feature-branch)
Author: Sadhguna Aithal <133648612+ape-with-helmet@users.noreply.github.com>
Date:   Wed Feb 28 04:14:52 2024 +0000

    Changes

diff --git a/ape-with-helmet b/ape-with-helmet
new file mode 160000
index 0000000..57f385c
--- /dev/null
+++ b/ape-with-helmet
@@ -0,0 +1 @@
+Subproject commit 57f385c417c810a64b5df6ed3ead17115b789d63
```

git tag v1.0 <commit_id>



```
@ape-with-helmet ➜/workspaces/github (main) $ git tag -a v19.1 -m "This is tag v1.0"
```

git push origin v1.0



```
@ape-with-helmet ➜/workspaces/github (feature-branch) $ git push origin feature-branch
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 292 bytes | 292.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'feature-branch' on GitHub by visiting:
remote:      https://github.com/ape-with-helmet/github/pull/new/feature-branch
remote:
To https://github.com/ape-with-helmet/github
 * [new branch]      feature-branch -> feature-branch
@ape-with-helmet ➜/workspaces/github (feature-branch) $ git pull origin feature-branch
From https://github.com/ape-with-helmet/github
 * branch            feature-branch -> FETCH_HEAD
Already up to date.
```

Experiment No 05: Advanced Git Operations

5.1 Objective

Write a command to cherry-pick a range of commits from "source-branch" to the current branch.

5.2 System Configuration

Windows 10

5.3 Pre-Requisites

Install Git bash and make required settings.

5.4 Introduction

git checkout -b source-branch:
Creates and switches to a new branch named "source-branch."

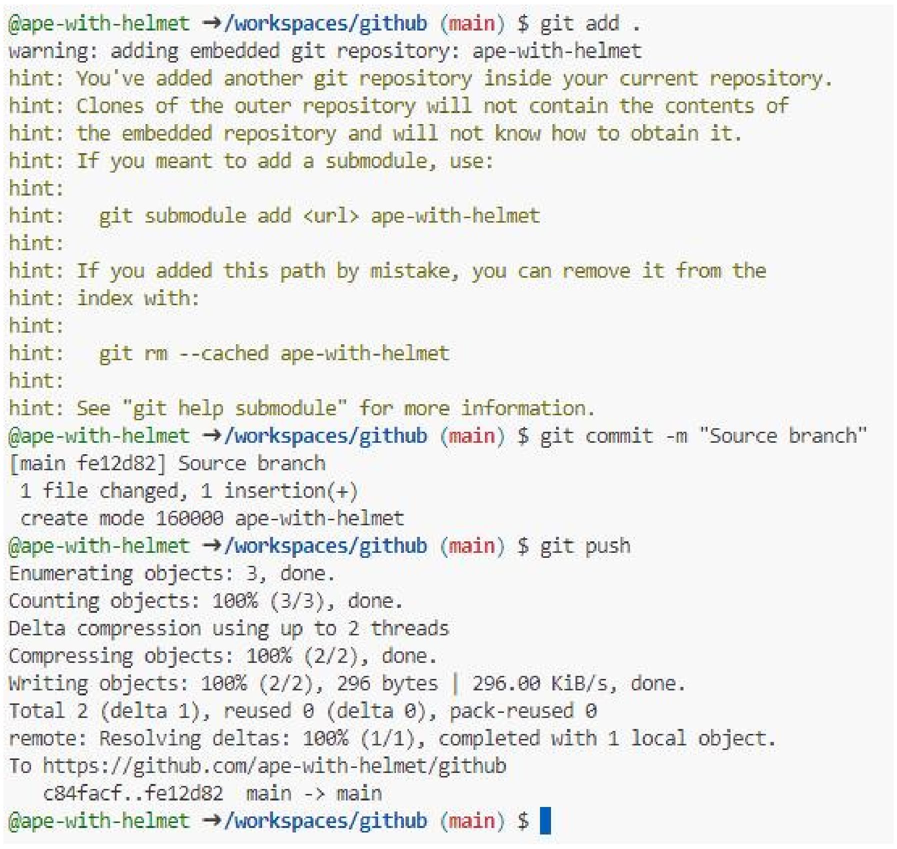
Makes some change filename:
Edits a file in the working directory.

git add .:
Stages all changes in the working directory for the next commit.

git push:
Pushes committed changes to the remote repository.

git commit -m "commit123":

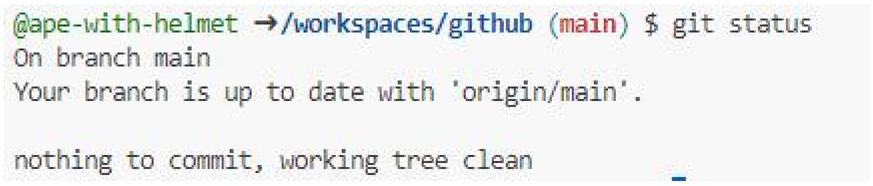
Commits the staged changes with the given commit message.



```
@ape-with-helmet ➜/workspaces/github (main) $ git add .
warning: adding embedded git repository: ape-with-helmet
hint: You've added another git repository inside your current repository.
hint: Clones of the outer repository will not contain the contents of
hint: the embedded repository and will not know how to obtain it.
hint: If you meant to add a submodule, use:
hint:
hint:   git submodule add <url> ape-with-helmet
hint:
hint: If you added this path by mistake, you can remove it from the
hint: index with:
hint:
hint:   git rm --cached ape-with-helmet
hint:
hint: See "git help submodule" for more information.
@ape-with-helmet ➜/workspaces/github (main) $ git commit -m "Source branch"
[main fe12d82] Source branch
 1 file changed, 1 insertion(+)
 create mode 160000 ape-with-helmet
@ape-with-helmet ➜/workspaces/github (main) $ git push
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 296 bytes | 296.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/ape-with-helmet/github
   c84facf..fe12d82  main -> main
@ape-with-helmet ➜/workspaces/github (main) $
```

git status:

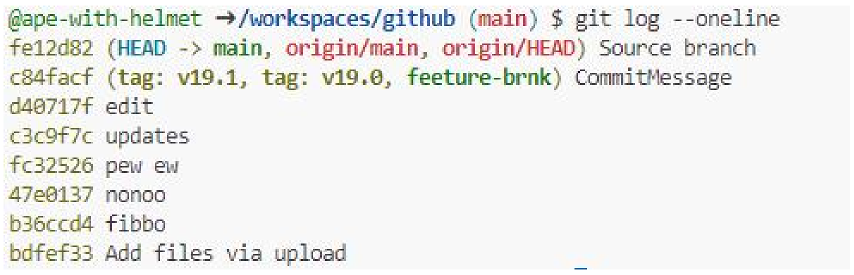
Displays the status of changes as untracked, modified, or staged.



```
@ape-with-helmet ➜/workspaces/github (main) $ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

git log --oneline:

Shows a concise log of commits with their hash and short descriptions.



```
@ape-with-helmet ➜/workspaces/github (main) $ git log --oneline
fe12d82 (HEAD -> main, origin/main, origin/HEAD) Source branch
c84facf (tag: v19.1, tag: v19.0, feeture-brnk) CommitMessage
d40717f edit
c3c9f7c updates
fc32526 pew ew
47e0137 nonoo
b36ccd4 fibbo
bdfef33 Add files via upload
```

Copy hash of the commit:

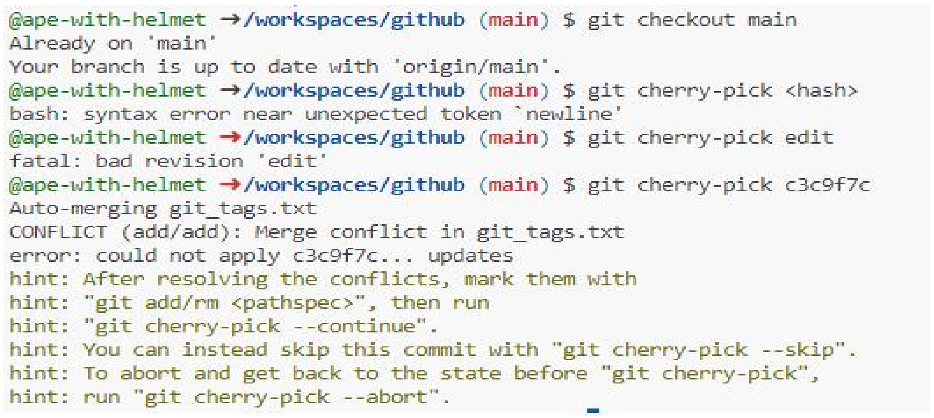
Manually record the commit hash for reference.

git checkout -m main:

Switches to the "main" branch, assuming it already exists.

git cherry-pick c3c9f7c:

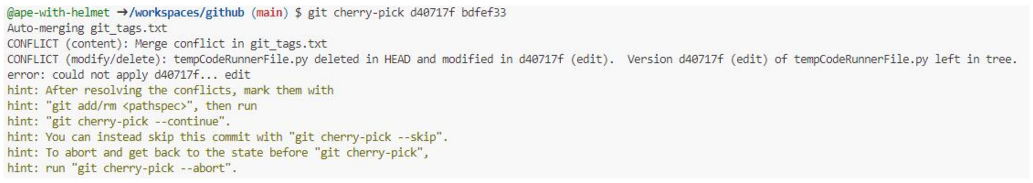
Applies the changes from a specific commit (**c3c9f7c**) to the current branch.



```
@ape-with-helmet →/workspaces/github (main) $ git checkout main
Already on 'main'
Your branch is up to date with 'origin/main'.
@ape-with-helmet →/workspaces/github (main) $ git cherry-pick <hash>
bash: syntax error near unexpected token `newline'
@ape-with-helmet →/workspaces/github (main) $ git cherry-pick edit
fatal: bad revision 'edit'
@ape-with-helmet →/workspaces/github (main) $ git cherry-pick c3c9f7c
Auto-merging git_tags.txt
CONFLICT (add/add): Merge conflict in git_tags.txt
error: could not apply c3c9f7c... updates
hint: After resolving the conflicts, mark them with
hint: "git add/rm <pathspec>", then run
hint: "git cherry-pick --continue".
hint: You can instead skip this commit with "git cherry-pick --skip".
hint: To abort and get back to the state before "git cherry-pick",
hint: run "git cherry-pick --abort".
```

git cherry-pick 5ei3ei...2b:

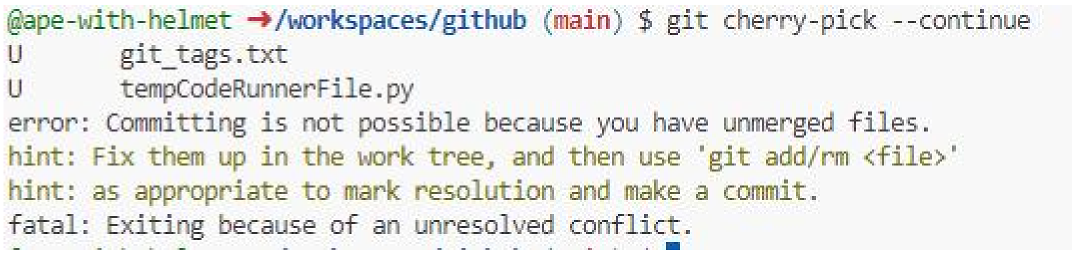
Picks a range of commits and applies them to the current branch.



```
@ape-with-helmet →/workspaces/github (main) $ git cherry-pick d40717f bdfef33
Auto-merging git_tags.txt
CONFLICT (content): Merge conflict in git_tags.txt
CONFLICT (modify/delete): tempCodeRunnerFile.py deleted in HEAD and modified in d40717f (edit).  Version d40717f (edit) of tempCodeRunnerFile.py left in tree.
error: could not apply d40717f... edit
hint: After resolving the conflicts, mark them with
hint: "git add/rm <pathspec>", then run
hint: "git cherry-pick --continue".
hint: You can instead skip this commit with "git cherry-pick --skip".
hint: To abort and get back to the state before "git cherry-pick",
hint: run "git cherry-pick --abort".
```

git cherry-pick --continue:

Continues the cherry-pick process after resolving conflicts.



```
@ape-with-helmet →/workspaces/github (main) $ git cherry-pick --continue
U       git_tags.txt
U       tempCodeRunnerFile.py
error: Committing is not possible because you have unmerged files.
hint: Fix them up in the work tree, and then use 'git add/rm <file>'
hint: as appropriate to mark resolution and make a commit.
fatal: Exiting because of an unresolved conflict.
```

git log --oneline:

Displays an updated log after cherry-picking, showing the new commit(s).

Experiment No.06: Analzing And Changing Git History

6.1 Objective

a) Given a commit ID, how would you use Git to view the details of that specific commit, including the author, date, and commit message?

b) Write the command to list all commits made by the author "JohnDoe" between "2023-01-01" and "2023-12-31."

c) Write the command to display the last five commits in the repository's history.

d) Write the command to undo the changes introduced by the commit with the ID "abc123".

6.2 System Configuration

Windows 10

6.3 Pre-Requisites:

Install Git bash and make required settings.

6.4 Introduction:

Git show <commit_id>

This command displays the details of a specific commit, including the commit message, author, date, and the changes made in the commit. It's useful for reviewing the details of a particular commit in your Git history.

git log --author=JohnDoe --after=2023-01-01 --before=2023-12-31

This command lists all commits made by the author "JohnDoe" between the dates "2023-01-01" and "2023-12-31". It's helpful for filtering the commit history based on authorship and date ranges.

git log -n 5

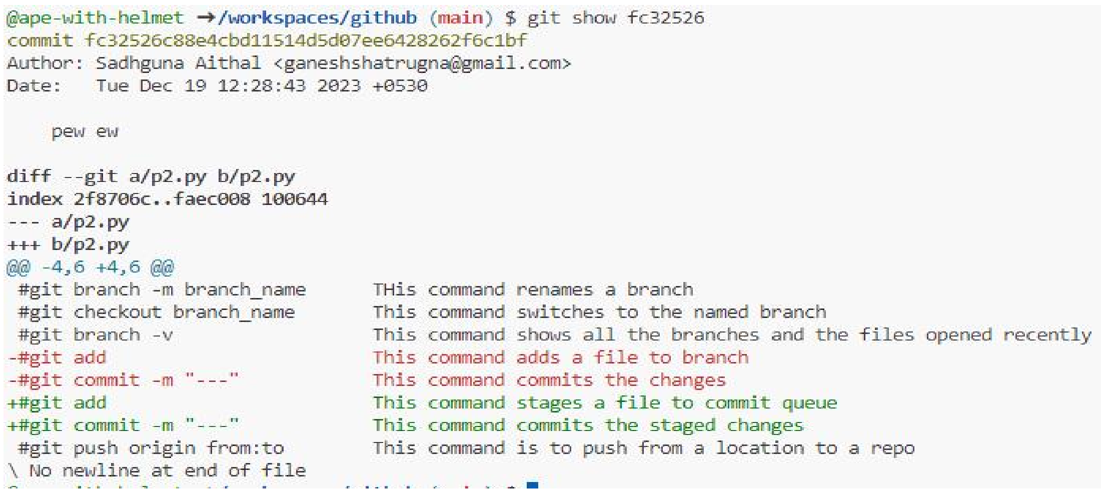
This command displays the last five commits in the repository's history. It's useful for quickly viewing recent changes and understanding the recent development activity in the repository.

git revert abc123

This command creates a new commit that undoes the changes introduced by the commit with the ID "abc123". It's a safe way to undo changes without altering the commit history, as it creates a new commit that reflects the changes being reverted.

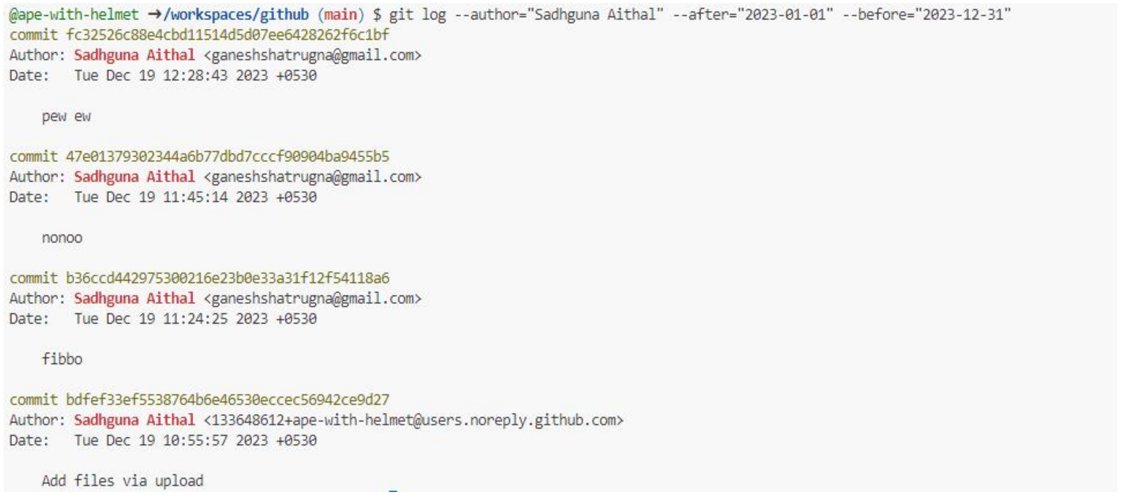
6.5 Procedure and Result:

1. Git show <commit_id>



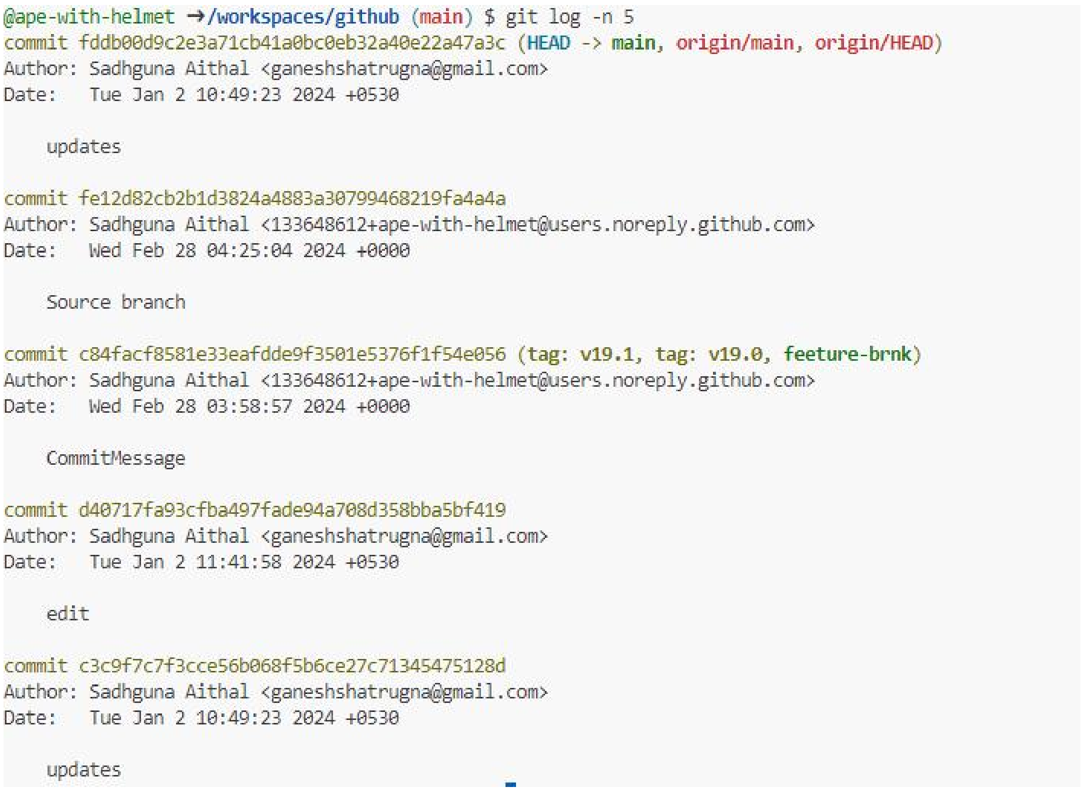
```
@ape-with-helmet →/workspaces/github (main) $ git show fc32526
commit fc32526c88e4cbd11514d5d07ee6428262f6c1bf
Author: Sadhguna Aithal <ganeshshatrugna@gmail.com>
Date:   Tue Dec 19 12:28:43 2023 +0530

    pew ew

diff --git a/p2.py b/p2.py
index 2f8706c..faec008 100644
--- a/p2.py
+++ b/p2.py
@@ -4,6 +4,6 @@
 #git branch -m branch_name      THis command renames a branch
 #git checkout branch_name       This command switches to the named branch
 #git branch -v                  This command shows all the branches and the files opened recently
-#git add                        This command adds a file to branch
-#git commit -m "---"            This command commits the changes
+#git add                        This command stages a file to commit queue
+#git commit -m "---"            This command commits the staged changes
 #git push origin from:to        This command is to push from a location to a repo
\ No newline at end of file
```

2. git log --author=JohnDoe --after=2023-01-01 --before=2023-12-31



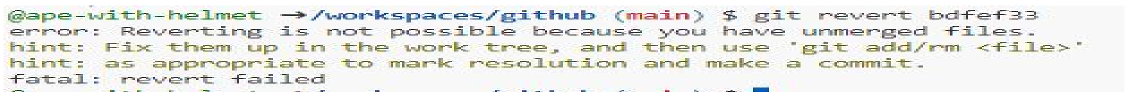
```
@ape-with-helmet →/workspaces/github (main) $ git log --author="Sadhguna Aithal" --after="2023-01-01" --before="2023-12-31"
commit fc32526c88e4cbd11514d5d07ee6428262f6c1bf
Author: Sadhguna Aithal <ganeshshatrugna@gmail.com>
Date:   Tue Dec 19 12:28:43 2023 +0530

    pew ew

commit 47e01379302344a6b77dbd7cccf90904ba9455b5
Author: Sadhguna Aithal <ganeshshatrugna@gmail.com>
Date:   Tue Dec 19 11:45:14 2023 +0530

    nonoo

commit b36ccd442975300216e23b0e33a31f12f54118a6
Author: Sadhguna Aithal <ganeshshatrugna@gmail.com>
Date:   Tue Dec 19 11:24:25 2023 +0530

    fibbo

commit bdfef33ef5538764b6e46530eccec56942ce9d27
Author: Sadhguna Aithal <133648612+ape-with-helmet@users.noreply.github.com>
Date:   Tue Dec 19 10:55:57 2023 +0530

    Add files via upload
```

3. git log -n 5



```
@ape-with-helmet →/workspaces/github (main) $ git log -n 5
commit fddb00d9c2e3a71cb41a0bc0eb32a40e22a47a3c (HEAD -> main, origin/main, origin/HEAD)
Author: Sadhguna Aithal <ganeshshatrugna@gmail.com>
Date:   Tue Jan 2 10:49:23 2024 +0530

    updates

commit fe12d82cb2b1d3824a4883a30799468219fa4a4a
Author: Sadhguna Aithal <133648612+ape-with-helmet@users.noreply.github.com>
Date:   Wed Feb 28 04:25:04 2024 +0000

    Source branch

commit c84facf8581e33eafdde9f3501e5376f1f54e056 (tag: v19.1, tag: v19.0, feeture-brnk)
Author: Sadhguna Aithal <133648612+ape-with-helmet@users.noreply.github.com>
Date:   Wed Feb 28 03:58:57 2024 +0000

    CommitMessage

commit d40717fa93cfba497fade94a708d358bba5bf419
Author: Sadhguna Aithal <ganeshshatrugna@gmail.com>
Date:   Tue Jan 2 11:41:58 2024 +0530

    edit

commit c3c9f7c7f3cce56b068f5b6ce27c71345475128d
Author: Sadhguna Aithal <ganeshshatrugna@gmail.com>
Date:   Tue Jan 2 10:49:23 2024 +0530

    updates
```

4. git revert abc123



```
@ape-with-helmet →/workspaces/github (main) $ git revert bdfef33
error: Reverting is not possible because you have unmerged files.
hint: Fix them up in the work tree, and then use 'git add/rm <file>'
hint: as appropriate to mark resolution and make a commit.
fatal: revert failed
```