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 stagingarea and commit the changes with an appropriate commit message

1.2 System Configuration

Windows 10, Linux,Mac

1.3 Pre-Requisite

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 setsup the necessary data structures and files that Git needs to start tracking changes to yourproject.

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 copiesthe files but also sets up a connection to the original repository so you can pull in updateslater.

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 thatexplain the purpose of the changes.

git push

This command is used to upload local repository content to a remote repository. In thecontext 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 Results

git clone ‘remote repository link’

```
PS C:\Users\asus\OneDrive\Desktop\git> git  clone https://github.com/shettyprathwish123/git-prathwish.git
Cloning into 'git-prathwish'...
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 3 (delta 0), pack-reused 0
Receiving objects: 100% (6/6), done.
```

git add .

```
PS C:\Users\asus\OneDrive\Desktop\git> git add .
PS C:\Users\asus\OneDrive\Desktop\git>
```

git commit -m "<message>"

```
PS C:\Users\asus\OneDrive\Desktop\git-prathwish> git commit -m "commit"
[main 6c265b3] commit
 1 file changed, 1 insertion(+)
 create mode 100644 ghgh.txt
```

git push

```
PS C:\Users\asus\OneDrive\Desktop\git-prathwish> git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 303 bytes | 151.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/shettyprathwish123/git-prathwish.git
   19e0353..6c265b3  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, Linux, Mac

2.3 Pre-Requisite

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.

2.4 Introduction

git branch -b feature-branch

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

git checkout master/main

This command switches to the "master" branch.

git merge feature-branch

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

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 Results

git branch feature-branch

git checkout feature-branch

```
PS C:\Users\asus\OneDrive\Desktop\github> git branch feature-branch1
PS C:\Users\asus\OneDrive\Desktop\github> git checkout feature-branch1
Switched to branch 'feature-branch1'
```

git add .

git commit -m “<message>”git checkout master git merge

```
PS C:\Users\asus\OneDrive\Desktop\github> git add .
PS C:\Users\asus\OneDrive\Desktop\github> git commit -m "branching1"
[feature-branch1 d1ccb79] branching1
 1 file changed, 2 insertions(+), 1 deletion(-)
```

```
PS C:\Users\asus\OneDrive\Desktop\github> git push origin feature-branch1:main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 263 bytes | 87.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/shettyprathwish123/github.git
   b47a985..d1ccb79  feature-branch1 -> main
```

git stash

git checkout feature-branch

gitstash apply

```
PS C:\Users\asus\OneDrive\Desktop\github> git stash
Saved working directory and index state WIP on feature-branch1: d1ccb79 branching1
PS C:\Users\asus\OneDrive\Desktop\github> git checkout main
Switched to branch 'main'
Your branch is behind 'origin/main' by 1 commit, and can be fast-forwarded.
  (use "git pull" to update your local branch)
```

```
PS C:\Users\asus\OneDrive\Desktop\github> git stash apply
Auto-merging p1.txt
CONFLICT (content): Merge conflict in p1.txt
On branch main
Your branch is behind 'origin/main' by 1 commit, and can be fast-forwarded.
  (use "git pull" to update your local branch)

Unmerged paths:
  (use "git restore --staged <file>..." to unstage)
  (use "git add <file>..." to mark resolution)
        both modified:   p1.txt

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

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 branchonto the updated remote branch.

c) Write a command to merge “feature-branch” into “master” while providing acustom 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 branchonto 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 remoterepository named "origin" without merging them into your local branches. Thesecond command (git rebase origin/<branch_name>) rebases your current localbranch onto the updated remote branch, integrating the latest changes from theremote repository while maintaining the commit

history.

c) Write a command to merge “feature-branch” into “master” while providing acustom 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 beperformed with a fast-forward.

3.5 Procedure and Results:

git clone <repository_url>

```
PS C:\Users\asus\git3> git  clone https://github.com/shettyprathwish123/git-prathwish.git
Cloning into 'git-prathwish'...
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 3 (delta 0), pack-reused 0
Receiving objects: 100% (6/6), done.
```

git checkout -b feature-branch

gitfetch origin

git rebase origin/<branch_name>

```
PS C:\Users\asus\prath> git branch -v
* feature-branch a470275 commit message
  master         a470275 commit message
PS C:\Users\asus\prath> git checkout -b feature-branch1
Switched to a new branch 'feature-branch1'
PS C:\Users\asus\prath> git branch -v
  feature-branch  a470275 commit message
* feature-branch1 a470275 commit message
  master          a470275 commit message
```

```
PS C:\Users\Admin\OneDrive\Desktop\gitexp3> git add .
PS C:\Users\Admin\OneDrive\Desktop\gitexp3> git commit -m "stashing"
[feature-branch c0d530f] stashing
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 something.txt
PS C:\Users\Admin\OneDrive\Desktop\gitexp3> git rebase origin
Current branch feature-branch is up to date.
```

EXPERIMENT N0 .04: Git Tags and Releases

4.1 Objective

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

4.2 System Configuration

Windows 10, Linux, Mac

4.3 Pre-Requisite

Install Git bash and make required settings.

4.4 Introduction

gitlog

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 lightweighttag. 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 andis 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 gitpush. This step is optional and depends on your workflow and the need to share the tagwith others.

4.5.Procedure and Results

git log –oneline

git tag v1.0 <commit_id>

git tag

git push origin v1.0

```
PS C:\Users\asus\OneDrive\Desktop\github> git log --oneline
b47a985 (HEAD -> main, feature3-branch, feature1-branch, feature-branch) commit
PS C:\Users\asus\OneDrive\Desktop\github> git tag v1.0 b47a985
PS C:\Users\asus\OneDrive\Desktop\github> git tag
v1.0
PS C:\Users\asus\OneDrive\Desktop\github> git push origin v1.0
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/shettyprathwish123/github.git
 * [new tag]         v1.0 -> v1.0
```

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, Linux, Mac

5.3 Pre-Requisite

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.git status:

Displays the status of changes as untracked, modified, or staged.
git log --oneline:

Shows a concise log of commits with their hash and short descriptions.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 cb0249:

Applies the changes from a specific commit (cb0249) to the current branch.git cherry-pick 5ei3ei...2b:

Picks a range of commits and applies them to the current branch.git cherry-pick --continue:

Continues the cherry-pick process after resolving conflicts.git log --oneline:

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

5.5 Procedure and Results

git log –oneline

git status

git checkout feature-branchgit cherry-pick <hash>

```
PS C:\Users\asus\OneDrive\Desktop\git5> git checkout -b source-branch
Switched to a new branch 'source-branch'
PS C:\Users\asus\OneDrive\Desktop\git5> git add .
PS C:\Users\asus\OneDrive\Desktop\git5> git commit -m "commit123"
[source-branch (root-commit) 8113996] commit123
 1 file changed, 6 insertions(+)
 create mode 100644 git5.t
PS C:\Users\asus\OneDrive\Desktop\git5> git status
On branch source-branch
nothing to commit, working tree clean
PS C:\Users\asus\OneDrive\Desktop\git5> git log --oneline
8113996 (HEAD -> source-branch) commit123
PS C:\Users\asus\OneDrive\Desktop\git5> git cherry-pick 8113996
On branch source-branch
You are currently cherry-picking commit 8113996.
  (all conflicts fixed: run "git cherry-pick --continue")
  (use "git cherry-pick --skip" to skip this patch)
  (use "git cherry-pick --abort" to cancel the cherry-pick operation)

nothing to commit, working tree clean
The previous cherry-pick is now empty, possibly due to conflict resolution.
If you wish to commit it anyway, use:

    git commit --allow-empty

Otherwise, please use 'git cherry-pick --skip'
```

EXPERIMENT No.06: Analyzing and Changing Git History

6.1 Objective

a) Given a commit ID, how would you use Git to view the details of that specificcommit, 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 Software Required

Windows 10, Linux, Mac

6.3 Pre-Requisite

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 forquickly 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 Results

git show <commit_id>

```
PS C:\Users\asus\OneDrive\Desktop\git5> git show 8113996
commit 8113996bd333e56594c5424fc9e1ca965661d6e9 (HEAD -> source-branch)
Author: shettyprathwish123 <shettyprathwish123@gmail.com>
Date:   Thu Jul 18 20:13:18 2024 +0530

    commit123

diff --git a/git5.t b/git5.t
new file mode 100644
index 0000000..c6725d9
--- /dev/null
+++ b/git5.t
@@ -0,0 +1,6 @@
+hffhfhf
+hgftfff
+hjgjhf
+hh
+hjhghg
+yfgffg
\ No newline at end of file
```

git log --author=”JohnDoe” --after=”2023-01-01” --before=”2023-12-31”
git log -n 5
git revert abc123

```
PS C:\Users\asus\OneDrive\Desktop\git5> git log --author="shettyprathwish123" --after="2023-01-12" --before="24-07-18"
PS C:\Users\asus\OneDrive\Desktop\git5> git log -n 5
commit 8113996bd333e56594c5424fc9e1ca965661d6e9 (HEAD -> source-branch)
Author: shettyprathwish123 <shettyprathwish123@gmail.com>
Date:   Thu Jul 18 20:13:18 2024 +0530

    commit123
```