Day – 5

Git-hub fundamentals

1)Local Repository

A **local repository** resides on a developer's local machine. It is the place where developers:

- Make changes to the code.
- Stage and commit changes.
- Test and debug before pushing changes to a remote repository.

Key Commands in Git:

- `git init`: Initializes a local repository.
- `git add`: Stages changes for a commit.
- `git commit`: Saves changes to the local repository

2)Distributed Repository

In a **distributed repository** model, every developer has a complete copy of the entire project history. This model is employed by Git, where:

- Developers can work offline.
- Changes can be merged across multiple repositories.

Advantages:

- No single point of failure.
- Better collaboration and flexibility.
- Full history available locally.

3)Centralized Repository

A **centralized repository** is a single, central location where all code resides. Developers:

- Pull updates from the central server.
- Push their changes back to the central server.

Features of Git

1. Distributed Version Control

- Every developer has a full copy of the repository, including the entire history.
- Enables offline work: developers can commit changes without needing a central server.
- Redundancy ensures no single point of failure.

---

2. Branching and Merging

- **Lightweight Branching**: Create, switch, and delete branches easily for feature development, bug fixes, or experiments.
  - `git branch`: Create a new branch.
  - `git checkout`: Switch between branches.
- **Merging**: Combine changes from different branches.
  - `git merge`: Merge changes from one branch into another.
  - `git rebase`: Reapply commits from one branch onto another.

---

3. Staging Area

- A unique **staging area** (or index) allows developers to prepare commits selectively.
  - `git add`: Stage changes.
  - `git reset`: Unstage changes.

---

4. Fast and Efficient

- Designed to handle large projects quickly.
- Optimized storage using compression and differential techniques.
- Commits and branching are nearly instantaneous.

---

5. History and Version Tracking

- Keeps a detailed history of changes (commits) with timestamps, authors, and messages.
  - `git log`: View commit history.
  - `git blame`: Show authorship of each line of a file.

---

6. Collaboration Tools

- Support for distributed workflows with **pull requests** and **code reviews**.
- Tools for resolving conflicts during merges.

---

7. Support for Remote Repositories

- Connect to remote repositories like GitHub, GitLab, or Bitbucket for collaboration.

- `git remote`: Manage remote connections.
- `git push`: Upload changes to a remote repository.
- `git pull`: Fetch and merge updates from a remote repository.

---

8. Undo Changes

- Flexible commands to undo mistakes:
  - `git revert`: Revert a specific commit.
  - `git reset`: Undo changes in the staging area or history.
  - `git stash`: Temporarily store changes without committing.

---

9. Security

- **SHA-1 Hashing**: Ensures data integrity by hashing content and references.
- Prevents accidental data loss with immutability of history.

---

10. Open Source and Extensible

- Free and open-source under the GNU General Public License (GPL).
- Extensible via custom hooks, aliases, and integrations with CI/CD tools.

Getting and creating project

Command	Description
`git init`	Initialize a local Git repository
`git clone ssh://git@github.com/[username]/[repository-name].git`	Create a local copy of a remote repository

Basic Snapshotting

Command	Description
`git status`	Check status
`git add [file-name.txt]`	Add a file to the staging area
`git add -A`	Add all new and changed files to the staging area
`git commit -m "[commit message]"`	Commit changes
`git rm -r [file-name.txt]`	Remove a file (or folder)
`git remote -v`	View the remote repository of the currently working file or directory

Branching & Merging

Command	Description
`git branch`	List branches (the asterisk denotes the current branch)
`git branch -a`	List all branches (local and remote)
`git branch [branch name]`	Create a new branch
`git branch -d [branch name]`	Delete a branch
`git push origin --delete [branch name]`	Delete a remote branch
`git checkout -b [branch name]`	Create a new branch and switch to it
`git checkout -b [branch name] origin/[branch name]`	Clone a remote branch and switch to it
`git branch -m [old branch name] [new branch name]`	Rename a local branch
`git checkout [branch name]`	Switch to a branch
`git checkout -`	Switch to the branch last checked out
`git checkout -- [file-name.txt]`	Discard changes to a file
`git merge [branch name]`	Merge a branch into the active branch
`git merge [source branch] [target branch]`	Merge a branch into a target branch
`git stash`	Stash changes in a dirty working directory
`git stash clear`	Remove all stashed entries
`git stash pop`	Apply latest stash to working directory

Sharing & Updating Projects

Command	Description
`git push origin [branch name]`	Push a branch to your remote repository
`git push -u origin [branch name]`	Push changes to remote repository (and remember the branch)
`git push`	Push changes to remote repository (remembered branch)
`git push origin --delete [branch name]`	Delete a remote branch
`git pull`	Update local repository to the newest commit
`git pull origin [branch name]`	Pull changes from remote repository
`git remote add origin ssh://git@github.com/[username]/[repository-name].git`	Add a remote repository
`git remote set-url origin ssh://git@github.com/[username]/[repository-name].git`	Set a repository's origin branch to SSH

Inspection & Comparison

Command	Description
`git log`	View changes
`git log --summary`	View changes (detailed)
`git log --oneline`	View changes (briefly)
`git diff [source branch] [target branch]`	Preview changes before merging

DAY -6

Restoring

git restore . for restoring from staging to working directory

git restore –workspace for restoring from local repository to working directory
git restore –staged for restoring from local repository to staging

Repository: It is a place to store and track the changes of files in software development

Commit: A commit is an action that saves changes made to the project's files to the repository. Each commit has a unique identifier and a message describing the changes

Branching: Creating separate line of development

Merging: merging allow multiple developers to work on different features or bug fixes simultaneously

Version History: Git keeps a detailed history of all changes made to the codebase. This helps you track progress, go back to previous states, and find bugs introduced in past commits.

Benefits of SCM:

Collaboration: Repositories enable multiple developers to work on the same project, making it easier to contribute and collaborate

Version Control: Tracks all code changes, allowing developers to go back to previous versions if needed.

Code Review: Changes can be reviewed before merging into the main codebase.

Risk Mitigation: The ability to track changes and revert to previous versions plays a crucial role in risk minimization. By maintaining a comprehensive log of modifications, SCM tools help prevent the loss of valuable work and ensure that code changes are well-documented and traceable.

Improved Productivity: Developers can work in parallel, switch between versions, and merge changes easily.

DAY-7

Commit: A commit is like a snapshot of your project at a specific point in time. Each commit in Git records changes to the files and is assigned a unique ID (commit hash)

Tag: Git tags are markers used to highlight specific commits in the history of your Git repository.

Hooks: It is a way to trigger other functions. Git comes with a set of script. Script can automatically run at every meaningful phase.

Index: File is where git stores staging area information

git commit –- amend This command helps to edit the commit message of the last message. The below screenshot displays the commit messaged modified.

Git push

```
1 file changed, 1 insertion(+)
create mode 100644 hello.txt

dministrator@37e9e1df41fc598 MINGW64 /c/User
 git remote add origin git@github.com:asnash

dministrator@37e9e1df41fc598 MINGW64 /c/User
 git push -u origin
atal: The current branch master has no upstr
o push the current branch and set the remote

   git push --set-upstream origin master

o have this happen automatically for branche
pstream, see 'push.autoSetupRemote' in 'git


dministrator@37e9e1df41fc598 MINGW64 /c/User
 git push -u origin master
numerating objects: 3, done.
ounting objects: 100% (3/3), done.
riting objects: 100% (3/3), 220 bytes | 9.00
otal 3 (delta 0), reused 0 (delta 0), pack-r
o github.com:asnashameel/Myust.git
* [new branch]      master -> master
ranch 'master' set up to track 'origin/maste

dministrator@37e9e1df41fc598 MINGW64 /c/User
 git checkout -b feature1
```

LFS

```
dministrator@37e9e1df41fc598 MINGW64 /learnLfs/lfs
 git init
nitialized empty Git repository in C:/Program Files/Git/learnLfs/lfs/.git/

dministrator@37e9e1df41fc598 MINGW64 /learnLfs/lfs (master)
 git lfs track "*.jpeg"
racking "*.jpeg"

dministrator@37e9e1df41fc598 MINGW64 /learnLfs/lfs (master)
 ls -a
/  ../  .git/  .gitattributes

dministrator@37e9e1df41fc598 MINGW64 /learnLfs/lfs (master)
 cat .gitattributes
.jpeg filter=lfs diff=lfs merge=lfs -text

dministrator@37e9e1df41fc598 MINGW64 /learnLfs/lfs (master)
 git lfs track "*.mov"
racking "*.mov"

dministrator@37e9e1df41fc598 MINGW64 /learnLfs/lfs (master)
 cat .gitattributes
.jpeg filter=lfs diff=lfs merge=lfs -text
.mov filter=lfs diff=lfs merge=lfs -text

dministrator@37e9e1df41fc598 MINGW64 /learnLfs/lfs (master)
 echo "" >> file1.jpeg

dministrator@37e9e1df41fc598 MINGW64 /learnLfs/lfs (master)
 cat file1.jpeg


dministrator@37e9e1df41fc598 MINGW64 /learnLfs/lfs (master)
 git add .

dministrator@37e9e1df41fc598 MINGW64 /learnLfs/lfs (master)
 git commit -m "image file"
master (root-commit) 506a8c0] image file
2 files changed, 5 insertions(+)
create mode 100644 .gitattributes
create mode 100644 file1.jpeg

dministrator@37e9e1df41fc598 MINGW64 /learnLfs/lfs (master)
 git remote add origin git@github.com:asnashameel/lfs.git
```

```
 cat .gitattributes
.jpeg filter=lfs diff=lfs merge=lfs -text
.mov filter=lfs diff=lfs merge=lfs -text

dministrator@37e9e1df41fc598 MINGW64 /learnLfs/lfs (master)
 echo "" >> file1.jpeg

dministrator@37e9e1df41fc598 MINGW64 /learnLfs/lfs (master)
 cat file1.jpeg


dministrator@37e9e1df41fc598 MINGW64 /learnLfs/lfs (master)
 git add .

dministrator@37e9e1df41fc598 MINGW64 /learnLfs/lfs (master)
 git commit -m "image file"
master (root-commit) 506a8c0] image file
2 files changed, 5 insertions(+)
create mode 100644 .gitattributes
create mode 100644 file1.jpeg

dministrator@37e9e1df41fc598 MINGW64 /learnLfs/lfs (master)
 git remote add origin git@github.com:asnashameel/lfs.git

dministrator@37e9e1df41fc598 MINGW64 /learnLfs/lfs (master)
 git push -u origin master
atch response: This repository is over its data quota. Account responsible for LFS bandwidth should purchase more data packs to restore access.
ploading LFS objects:   0% (0/1), 0 B | 0 B/s, done.
rror: failed to push some refs to 'github.com:asnashameel/lfs.git'

dministrator@37e9e1df41fc598 MINGW64 /learnLfs/lfs (master)
 git push -u origin master
ploading LFS objects: 100% (1/1), 1 B | 0 B/s, done.
numerating objects: 4, done.
ounting objects: 100% (4/4), done.
elta compression using up to 4 threads
ompressing objects: 100% (4/4), done.
riting objects: 100% (4/4), 413 bytes | 413.00 KiB/s, done.
otal 4 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
o github.com:asnashameel/lfs.git
 * [new branch]      master -> master
ranch 'master' set up to track 'origin/master'.

dministrator@37e9e1df41fc598 MINGW64 /learnLfs/lfs (master)
```

Reset

```
$ git add .
warning: in the working copy of 'test3.txt', LF will be replaced by CRLF the next time Git touches it

Administrator@37e9e1df41fc598 MINGW64 /c/Users/reset (master)
$ git commit -m "m4"
[master 69b667a] m4
 1 file changed, 1 insertion(+)
 create mode 100644 test3.txt

Administrator@37e9e1df41fc598 MINGW64 /c/Users/reset (master)
$ git log --oneline
69b667a (HEAD -> master) m4
96dcf1a m3
641b9f2 m2
75cd689 m1

Administrator@37e9e1df41fc598 MINGW64 /c/Users/reset (master)
$ git reset --soft HEAD

Administrator@37e9e1df41fc598 MINGW64 /c/Users/reset (master)
$ git status
On branch master
nothing to commit, working tree clean

Administrator@37e9e1df41fc598 MINGW64 /c/Users/reset (master)
$ git reset --soft HEAD~1

Administrator@37e9e1df41fc598 MINGW64 /c/Users/reset (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   test3.txt


Administrator@37e9e1df41fc598 MINGW64 /c/Users/reset (master)
$ git commit -m "m5"
[master 5c11c1f] m5
 1 file changed, 1 insertion(+)
 create mode 100644 test3.txt

Administrator@37e9e1df41fc598 MINGW64 /c/Users/reset (master)
$ git log --oneline
5c11c1f (HEAD -> master) m5
```

Git tag

```
\ No newline at end of file
diff --git a/abc.log.txt b/abc.log.txt
new file mode 100644

Administrator@37e9e1df41fc598 MINGW64 /c/Users/myust (master)
$ git tag -a @author d0ec2 -m asna
fatal: tag '@author' already exists

Administrator@37e9e1df41fc598 MINGW64 /c/Users/myust (master)
$ git show @author
tag @author
Tagger: asnashameel <2020.cse.asna@ekc.edu.in>
Date:   Thu Jan 16 10:35:06 2025 +0530

asna

commit d0ec2ffcad5a7b93e51e3cffe2adbabcdd56d47c (tag: @author)
Author: asnashameel <2020.cse.asna@ekc.edu.in>
Date:   Wed Jan 15 14:59:30 2025 +0530

    commit modified

diff --git a/.gitignore b/.gitignore
new file mode 100644
index 0000000..38105ec
--- /dev/null
+++ b/.gitignore
@@ -0,0 +1,24 @@
+# Compiled class file
+*.class
+
+# Log file
+*.log
+
+# BlueJ files
+*.ctxt
+
+# Mobile Tools for Java (J2ME)
+.mtj.tmp/
+
+# Package Files #
+*.jar
+*.war
+*.nar
+*.ear
+*.zip
```

Git ignore

```
ministrator@37e9e1df41fc598 MINGW64 /c/Users/NewIgnore (master)
git commit -m "class"
aster (root-commit) 8b1285b] class
 file changed, 1 insertion(+)
reate mode 100644 abc.class

ministrator@37e9e1df41fc598 MINGW64 /c/Users/NewIgnore (master)
ls
ew Text Document (2).txt'  'New Text Document.txt'   abc.class

ministrator@37e9e1df41fc598 MINGW64 /c/Users/NewIgnore (master)
ls -a
/   ../   .git/   .gitignore  'New Text Document (2).txt'  'New Text Document.txt'

ministrator@37e9e1df41fc598 MINGW64 /c/Users/NewIgnore (master)
ls -a
  ../  .git/  .gitignore  abc.class

ministrator@37e9e1df41fc598 MINGW64 /c/Users/NewIgnore (master)
cp abc.class newfile.class

ministrator@37e9e1df41fc598 MINGW64 /c/Users/NewIgnore (master)
ls
c.class  newfile.class

ministrator@37e9e1df41fc598 MINGW64 /c/Users/NewIgnore (master)
git status
 branch master
tracked files:
(use "git add <file>..." to include in what will be committed)
       .gitignore

thing added to commit but untracked files present (use "git add" to track)

ministrator@37e9e1df41fc598 MINGW64 /c/Users/NewIgnore (master)
echo "newline" >> abc.class

ministrator@37e9e1df41fc598 MINGW64 /c/Users/NewIgnore (master)
git status
 branch master
anges not staged for commit:
(use "git add <file>..." to update what will be committed)
```

Git branch

```
 git push -u origin master
numerating objects: 3, done.
ounting objects: 100% (3/3), done.
riting objects: 100% (3/3), 220 bytes | 9.00 KiB/s, done.
otal 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
o github.com:asnashameel/Myust.git
 * [new branch]      master -> master
ranch 'master' set up to track 'origin/master'.

dministrator@37e9e1df41fc598 MINGW64 /c/Users/Myust (master)
 git checkout -b feature1
witched to a new branch 'feature1'

dministrator@37e9e1df41fc598 MINGW64 /c/Users/Myust (feature1)
 echo "learning pull request" >> hello.txt

dministrator@37e9e1df41fc598 MINGW64 /c/Users/Myust (feature1)
 cat hello.txt
ello World
earning pull request

dministrator@37e9e1df41fc598 MINGW64 /c/Users/Myust (feature1)
 git add .
arning: in the working copy of 'hello.txt', LF will be replaced by CRLF the next time Git touches it

dministrator@37e9e1df41fc598 MINGW64 /c/Users/Myust (feature1)
 git commit -m "Learning pull request"
feature1 5bb9a8c] Learning pull request
 1 file changed, 1 insertion(+)

dministrator@37e9e1df41fc598 MINGW64 /c/Users/Myust (feature1)
 git push -u origin feature1
numerating objects: 5, done.
ounting objects: 100% (5/5), done.
riting objects: 100% (3/3), 287 bytes | 95.00 KiB/s, done.
otal 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
emote:
emote: Create a pull request for 'feature1' on GitHub by visiting:
emote:      https://github.com/asnashameel/Myust/pull/new/feature1
emote:
o github.com:asnashameel/Myust.git
 * [new branch]      feature1 -> feature1
ranch 'feature1' set up to track 'origin/feature1'.

dministrator@37e9e1df41fc598 MINGW64 /c/Users/Myust (feature1)
```