SOFTWARE ENGINEERING

UNIT – 2

TOPIC – 8

GIT COMMANDS: WORKING WITH LOCAL AND REMOTE REPOSITORIES - BRANCHES, CHECKOUT, MERGE, REVERT, LOG

Branches in Git

A **branch** in Git is like a **separate copy of your project** where you can make changes without affecting the main version. You might need different branches to work on new features, fix bugs, or prepare for a release.

- Imagine you have a main branch (called **master**) that has the latest stable version of your project. If you need to create a new feature (like adding a search function), you would create a **feature branch**. This allows you to work on the search function independently. Once the feature is complete, you can combine it (merge) with the master branch.

  Types of Branches:

  - **Main Branch (Master)**: The main project everyone uses.
  - **Feature Branch**: A branch to work on specific new features.
  - **Bug Fix Branch**: Specifically for fixing bugs.
  - **Release Branch**: Prepares and tests the project before it is released to users.

Important Commands:

- `git branch <branch_name>:` Creates a new branch, such as `git branch search-feature.`
- `git branch -a:` Shows a list of all branches, including the main branch, feature branches, and remote branches.

- `git branch -d <branch_name>`: Deletes a branch that you no longer need, such as `git branch -d bugfix`.

```
Madhu@DESKTOP-CLQ6BHJ MINGW64 ~/Desktop/AY-23-24-Sem-2/RKR21-SE/Tes
seract/Sample_Example (master)
$ git branch project_changes

Madhu@DESKTOP-CLQ6BHJ MINGW64 ~/Desktop/AY-23-24-Sem-2/RKR21-SE/Tesseract/Sample_E
xample (master)
$ git branch -a
* master
  project_changes

Madhu@DESKTOP-CLQ6BHJ MINGW64 ~/Desktop/AY-23-24-Sem-2/RKR21-SE/Tesseract/Sample_E
xample (master)
$ git branch -d project_changes
Deleted branch project_changes (was 4e87650).

Madhu@DESKTOP-CLQ6BHJ MINGW64 ~/Desktop/AY-23-24-Sem-2/RKR21-SE/Tesseract/Sample_E
xample (master)
$ git branch -a
* master
```

2. Switching Between Branches: Checkout

The **git checkout** command is used to **switch between branches**. This allows you to move from one task to another easily. For example, you may be working on a new feature in one branch but need to fix a bug in another branch.

- Let's say you're working on a new search feature in a branch called `search-feature`, but suddenly a critical bug is found in the main branch (`master`). You can **switch to the bug fix branch** using `git checkout` and fix the bug. Once the bug is fixed, you can switch back to the `search-feature` branch and continue working without losing your progress.

Important Commands:

- `git checkout <branch_name>`: Switches to an existing branch, like `git checkout bugfix` to fix the bug.
- `git checkout -b <new_branch>:` Creates a new branch and switches to it immediately. For example, `git checkout -b update-design` will create and move to a new branch where you can work on a design update.

```
Madhu@DESKTOP-CLQ6BHJ MINGW64 ~/Desktop/AY-23-24-Sem-2/RKR21-SE/Tesseract/Sample_E
xample (master)
$ git branch project_changes

Madhu@DESKTOP-CLQ6BHJ MINGW64 ~/Desktop/AY-23-24-Sem-2/RKR21-SE/Tesseract/Sample_E
xample (master)
$ git checkout project_changes
Switched to branch 'project_changes'

Madhu@DESKTOP-CLQ6BHJ MINGW64 ~/Desktop/AY-23-24-Sem-2/RKR21-SE/Tesseract/Sample_E
xample (project_changes)
$ git checkout -b project_bugs
Switched to a new branch 'project_bugs'

Madhu@DESKTOP-CLQ6BHJ MINGW64 ~/Desktop/AY-23-24-Sem-2/RKR21-SE/Tesseract/Sample_E
xample (project_bugs)
$ git branch -a
  master
* project_bugs
  project_changes
```

3. Combining Work: Merge

After working on a separate branch for a while (e.g., developing a new feature or fixing a bug), you will want to **combine your work** with the main project. This is where **git merge** comes in—it merges the changes from one branch into another, typically merging changes into the **master** branch.

- You've finished working on the `search-feature` branch, and it's time to **merge it into the master branch**. By using `git merge search-feature` while on the `master` branch, you combine all the changes from the search-feature branch into the master. Now, the master branch contains both the original stable version and the new feature you've developed.

Important Command:

- `git merge <branch_name>`: Merges the changes from the specified branch into the current branch. For example, `git merge search-feature` combines the search feature with the master branch.

```
Madhu@DESKTOP-CLQ6BHJ MINGW64 ~/Desktop/AY-23-24-Sem-2/RKR21-SE/Tesseract/Sample_E
xample (master)
$ git branch project_changes

Madhu@DESKTOP-CLQ6BHJ MINGW64 ~/Desktop/AY-23-24-Sem-2/RKR21-SE/Tesseract/Sample_E
xample (master)
$ git checkout project_changes
Switched to branch 'project_changes'

Madhu@DESKTOP-CLQ6BHJ MINGW64 ~/Desktop/AY-23-24-Sem-2/RKR21-SE/Tesseract/Sample_E
xample (project_changes)
$ ls
a.txt

Madhu@DESKTOP-CLQ6BHJ MINGW64 ~/Desktop/AY-23-24-Sem-2/RKR21-SE/Tesseract/Sample_E
xample (project_changes)
$ nano b.txt

Madhu@DESKTOP-CLQ6BHJ MINGW64 ~/Desktop/AY-23-24-Sem-2/RKR21-SE/Tesseract/Sample_E
xample (project_changes)
$ ls
a.txt  b.txt

Madhu@DESKTOP-CLQ6BHJ MINGW64 ~/Desktop/AY-23-24-Sem-2/RKR21-SE/Tesseract/Sample_E
xample (project_changes)
$ git add .
warning: LF will be replaced by CRLF in b.txt.
The file will have its original line endings in your working directory

Madhu@DESKTOP-CLQ6BHJ MINGW64 ~/Desktop/AY-23-24-Sem-2/RKR21-SE/Tesseract/Sample_E
xample (project_changes)
$ git commit -m "New file added in new branch project_changes"
[project_changes e525410] New file added in new branch project_changes
 1 file changed, 1 insertion(+)
 create mode 100644 b.txt

Madhu@DESKTOP-CLQ6BHJ MINGW64 ~/Desktop/AY-23-24-Sem-2/RKR21-SE/Tesseract/Sample_E
xample (project_changes)
$ ls
a.txt  b.txt
```

```
Madhu@DESKTOP-CLQ6BHJ MINGW64 ~/Desktop/AY-23-24-Sem-2/RKR21-SE/Tesseract/Sample_E
xample (project_changes)
$ git checkout master
Switched to branch 'master'

Madhu@DESKTOP-CLQ6BHJ MINGW64 ~/Desktop/AY-23-24-Sem-2/RKR21-SE/Tesseract/Sample_Example (master)
$ ls
a.txt

Madhu@DESKTOP-CLQ6BHJ MINGW64 ~/Desktop/AY-23-24-Sem-2/RKR21-SE/Tesseract/Sample_Example (master)
$ git merge project_changes
Updating 4e87650..e525410
Fast-forward
 b.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 b.txt

Madhu@DESKTOP-CLQ6BHJ MINGW64 ~/Desktop/AY-23-24-Sem-2/RKR21-SE/Tesseract/Sample_Example (master)
$ ls
a.txt  b.txt
```

4. Viewing History: Log

Git keeps a **detailed history** of all changes (called **commits**) made to the project. The **git log** command shows you this history, so you can see who made changes, when they made them, and what the changes were. This is useful for tracking the progress of the project and identifying when bugs were introduced.

- Suppose you want to know when a specific feature was added or who worked on a particular part of the code. By using the `git log` command, you can see a detailed history of all the changes. Each entry in the log will show:
  - **Commit Hash**: A unique identifier for each change.
  - **Author**: The person who made the change.
  - **Date**: When the change was made.
  - **Message**: A short description of the change, like "added search functionality."

```
Madhu@DESKTOP-CLQ6BHJ MINGW64 ~/Desktop/AY-23-24-Sem-2/RKR21-SE/Tesseract/Sample_Example (master)
$ git log
commit e525410c637e693b0b512bbb2fe67ca8c3763b14 (HEAD -> master)
Author: Madhurika, Budaraju <budarajumadhurika@gmail.com>
Date:   Wed Oct 16 00:16:39 2024 +0530

    New file added in new branch project_changes

commit 4e87650a0f49e0b6f82b53e693099f5987eeb972 (project_bugs)
Author: Madhurika, Budaraju <budarajumadhurika@gmail.com>
Date:   Tue Oct 15 14:37:32 2024 +0530

    Created new file
```

- **Simplified View**:
  If you don't want to see too much detail, you can use the `git log --oneline` command. This shows a summary of each commit, making it easier to quickly scan through the project's history.

```
Madhu@DESKTOP-CLQ6BHJ MINGW64 ~/Desktop/AY-23-24-Sem-2/RKR21-SE/Tesseract/Sample_Example (master)
$ git log --oneline
e525410 (HEAD -> master) New file added in new branch project_changes
4e87650 (project_bugs) Created new file
```

Important Commands:

- `git log`: Shows the complete history of commits in a branch.
- `git log --oneline`: Shows a simpler view with one line per commit, like `f3b9c2b Added search feature.`

5. Undoing Changes: Revert

If a mistake was made in a previous commit, Git provides a way to **undo those changes** using the `git revert` command. This doesn't delete the commit but creates a new commit that cancels out the earlier changes. This way, the history of the project is preserved.

- Imagine you've added a new feature but later discover that it introduced a bug. Instead of deleting that commit, you can use `git revert` to undo the changes made in that commit. For example, if the commit hash for the problematic commit is `f3b9c2b`, you would use `git revert f3b9c2b`. Git will then create a new commit that undoes the changes introduced by that commit without deleting it from the project history.

Important Command:

- `git revert <commit_hash>`: Reverts (undoes) the changes from a specific commit. For example, `git revert f3b9c2b` will undo the changes made in the commit with the ID `f3b9c2b`.

```
Madhu@DESKTOP-CLQ6BHJ MINGW64 ~/Desktop/AY-23-24-Sem-2/RKR21-SE/Tesseract/Sample_Example (master)
$ git checkout project_changes
Switched to branch 'project_changes'

Madhu@DESKTOP-CLQ6BHJ MINGW64 ~/Desktop/AY-23-24-Sem-2/RKR21-SE/Tesseract/Sample_Example (project_changes)
$ git log --oneline
e525410 (HEAD -> project_changes, master) New file added in new branch project_changes
4e87650 (project_bugs) Created new file

Madhu@DESKTOP-CLQ6BHJ MINGW64 ~/Desktop/AY-23-24-Sem-2/RKR21-SE/Tesseract/Sample_Example (project_changes)
$ git revert e525410
[project_changes 7c2e526] Revert "New file added in new branch project_changes"
 1 file changed, 1 deletion(-)
 delete mode 100644 b.txt

Madhu@DESKTOP-CLQ6BHJ MINGW64 ~/Desktop/AY-23-24-Sem-2/RKR21-SE/Tesseract/Sample_Example (project_changes)
$ git status
On branch project_changes
nothing to commit, working tree clean

Madhu@DESKTOP-CLQ6BHJ MINGW64 ~/Desktop/AY-23-24-Sem-2/RKR21-SE/Tesseract/Sample_Example (project_changes)
$ ls
a.txt
```

Flow of Using Git Commands (with Examples):

1. **Create a Branch**:

   You create a branch to work on a new feature:

```
git branch new-feature
```

2. **Switch to the Branch**:

   You switch to the new branch to start working on the feature:

   ```
   git checkout new-feature
   ```

3. **Develop Your Feature and Commit Changes**:

   You make changes, then commit them:

   ```
   git commit -m "Developed search functionality"
   ```

4. **Switch to Another Branch to Fix a Bug**:

   Suddenly, you need to fix a bug, so you switch to the bug fix branch:

   ```
   git checkout bugfix
   ```

5. **Merge the Bug Fix Into the Main Branch**:

   Once the bug is fixed, you merge it into the master branch:

   ```
   git checkout master
   git merge bugfix
   ```

6. **Check History Using Log**:

   You want to review the commits to make sure everything is correct:

   ```
   git log
   ```

7. **Revert a Mistake**:

   If you discover a mistake in a previous commit, you can revert it:

   ```
   git revert f3b9c2b
   ```