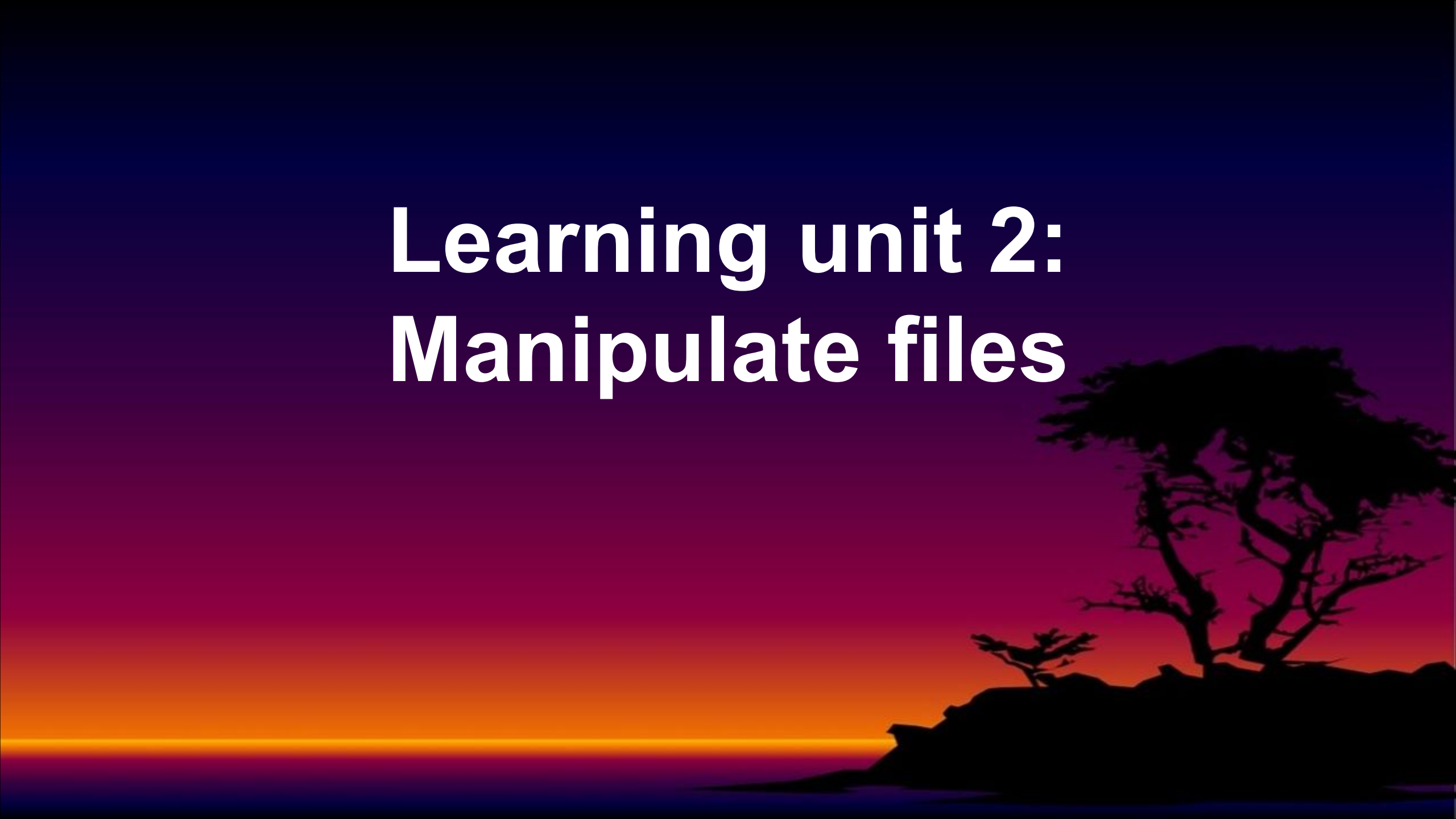
Learning unit 2: Manipulate files



2.2. Definition of general key terms

- Status: The "status" in Git refers to the current state of your working directory and staging area. When you run the "git status" command, Git provides information about the modified, added, or deleted files in your repository. It helps you track the changes made to your files and see which files are ready to be committed.
- Branch: A "branch" in Git is a parallel line of development. It represents an independent line of work within a repository. Each branch can have its own commits, allowing multiple people to work on different features or bug fixes simultaneously. The main branch, usually named "master" or "main," represents the default and most stable version of the project. Creating and switching between branches allows you to work on different features or experiments without affecting the main branch.

- Commit: A "commit" in Git represents a snapshot of your repository at a specific point in time. It captures the changes made to files and includes a unique identifier called a commit hash. When you make a commit, you are saving your changes permanently in the Git history.

Each commit contains a message describing the changes made, providing a concise summary of the modifications. Commits are essential for tracking the history of a project, allowing you to review changes, revert to previous versions, and collaborate with others.

✓ Add file change to git staging area

- **Operations on git status command**

› **git status:** Check the status of your repository and any files changes

› **View new untracked files:** The "git status" command will show any untracked files in your working directory. These are files that Git is not currently tracking.

git add <file-name>: To add a new untracked file to the staging area

Replace "<file-name>" with the name of the untracked file you want to add.

This command stages the file, making it ready to be committed.

› **View modified file**

The "git status" command also displays modified files in your working directory. These are files that have been changed since the last commit. To add a modified file to the staging area, use the same "git add" command mentioned above.

› **View deleted file**

- If you have deleted a file from your working directory, the "git status" command will show it as a deleted file. To stage the deletion, use the "git add" command as well:

git add <file-name>

Git will now consider the file changes you added using the "git add" command as part of the staging area.

› **Verify the staging area**

To check the status of the staging area and see which files are staged for the next commit, you can use the "git status" command again. It will display the files that have been added to the staging area and are ready to be committed.

- **Operation on git add command**

The "**git add**" command in Git is used to add file changes to the staging area.

› **Stage all files**: git add . The period (".") is used to represent the current directory.

› **Stage a file**: git add <file-name>

Replace "<file-name>" with the name of the specific file you want to stage.

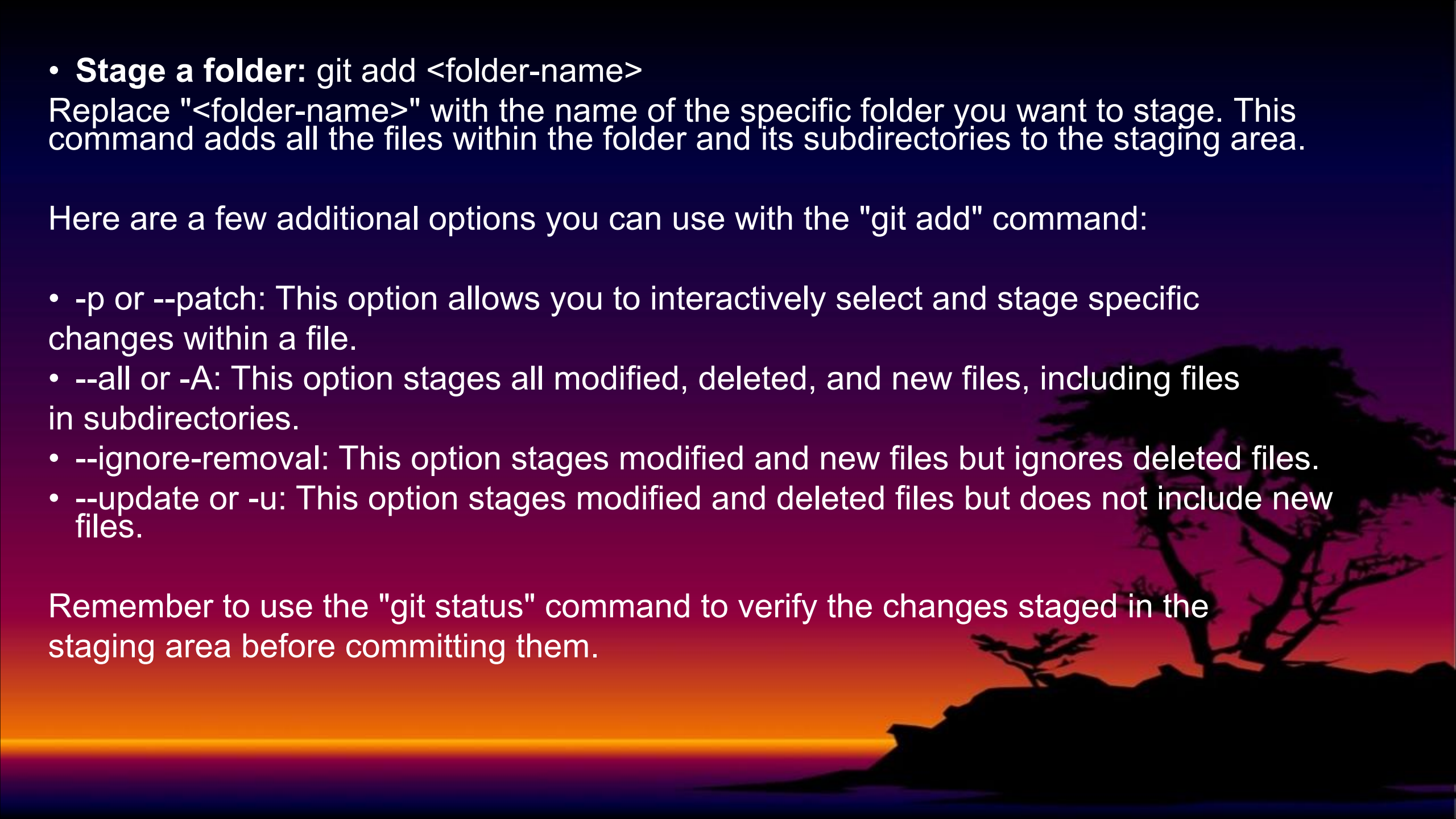
- **Stage a folder:** git add <folder-name>

Replace "<folder-name>" with the name of the specific folder you want to stage. This command adds all the files within the folder and its subdirectories to the staging area.

Here are a few additional options you can use with the "git add" command:

- -p or --patch: This option allows you to interactively select and stage specific changes within a file.
- --all or -A: This option stages all modified, deleted, and new files, including files in subdirectories.
- --ignore-removal: This option stages modified and new files but ignores deleted files.
- --update or -u: This option stages modified and deleted files but does not include new files.

Remember to use the "git status" command to verify the changes staged in the staging area before committing them.



Operations on git reset command for:

› **Unstage a file:** To unstage a file using git reset, you can follow these steps:

- Open your command line interface or terminal.
- Navigate to the root directory of your Git repository.
- Execute the following command to unstage the file:
- **git reset HEAD <file>**

- Replace with the path to the file you want to unstage. For example, if you want to unstage a file named "example.txt" located in the root directory of your repository, the command would be:

git reset HEAD example.txt

This command removes the file from the staging area, but it will still remain in your working directory.

› **Deleting and staging file/folder**

To delete a file or folder and stage the deletion using git reset, Navigate to the root directory of your Git repository. Execute the following command:

git rm <file/folder>

- Replace with the path to the file or folder you want to delete.

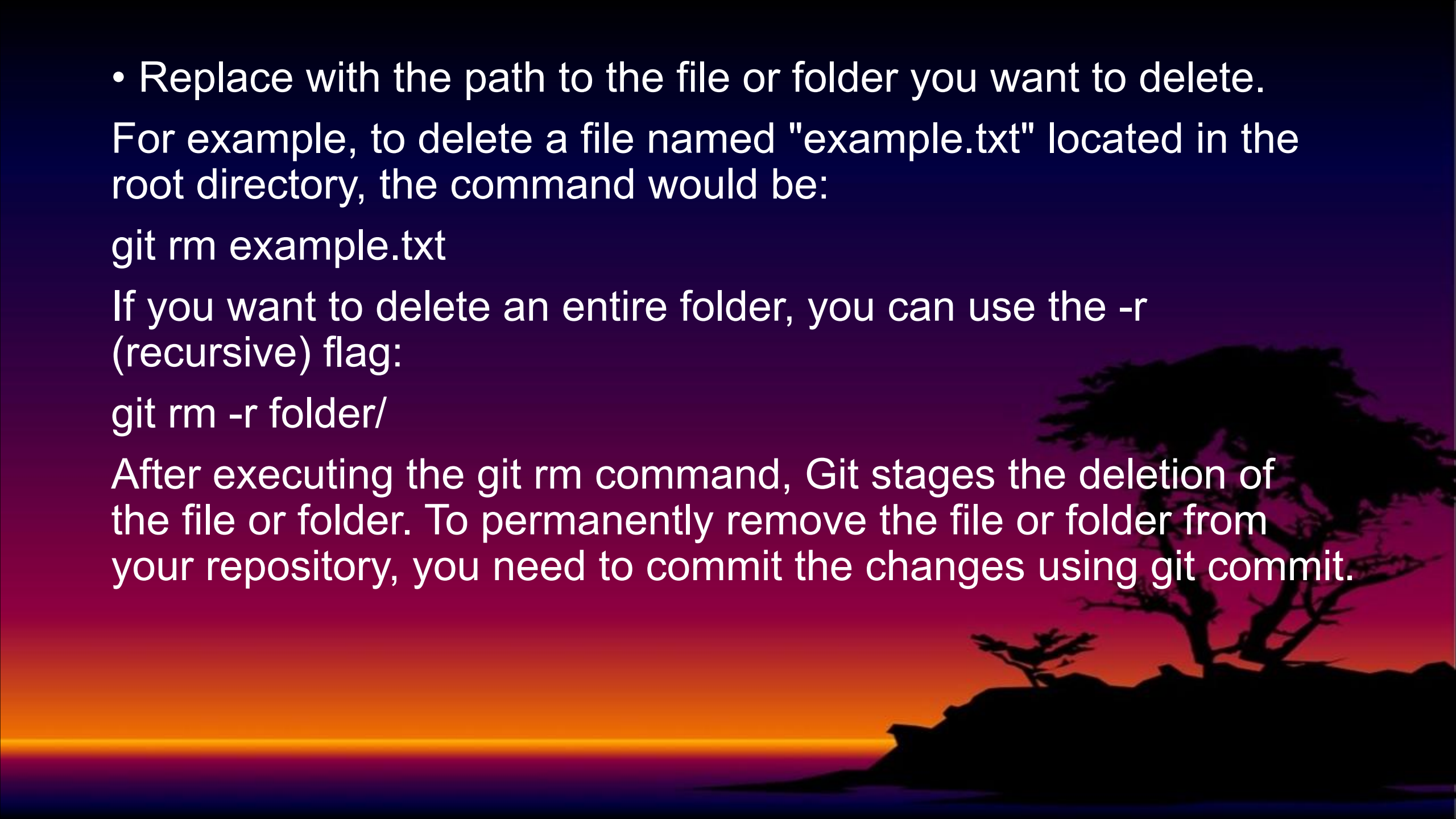
For example, to delete a file named "example.txt" located in the root directory, the command would be:

git rm example.txt

If you want to delete an entire folder, you can use the -r (recursive) flag:

git rm -r folder/

After executing the git rm command, Git stages the deletion of the file or folder. To permanently remove the file or folder from your repository, you need to commit the changes using git commit.



- Remember to be cautious when using the git reset and git rm commands, as they modify your repository's history. Always double-check the changes you're making and ensure they align with your intentions.

› **Commit File changes to git local repository**

✓ Best practice of creating a commit message

Operations on the git commit command like committing a file and editing the commit message, Here are the steps to commit file changes to a Git local repository:

- Open your command line interface or terminal.

- Navigate to the root directory of your Git repository.

- Use the following command to view the status of your repository and see the changes you've made:

git status

- You need to add a file to the staging area using the git add command.

For example, to stage a file named "example.txt" located in the root directory, use the following command:

git add example.txt

You can also use the git add command with wildcards or add multiple files at once.

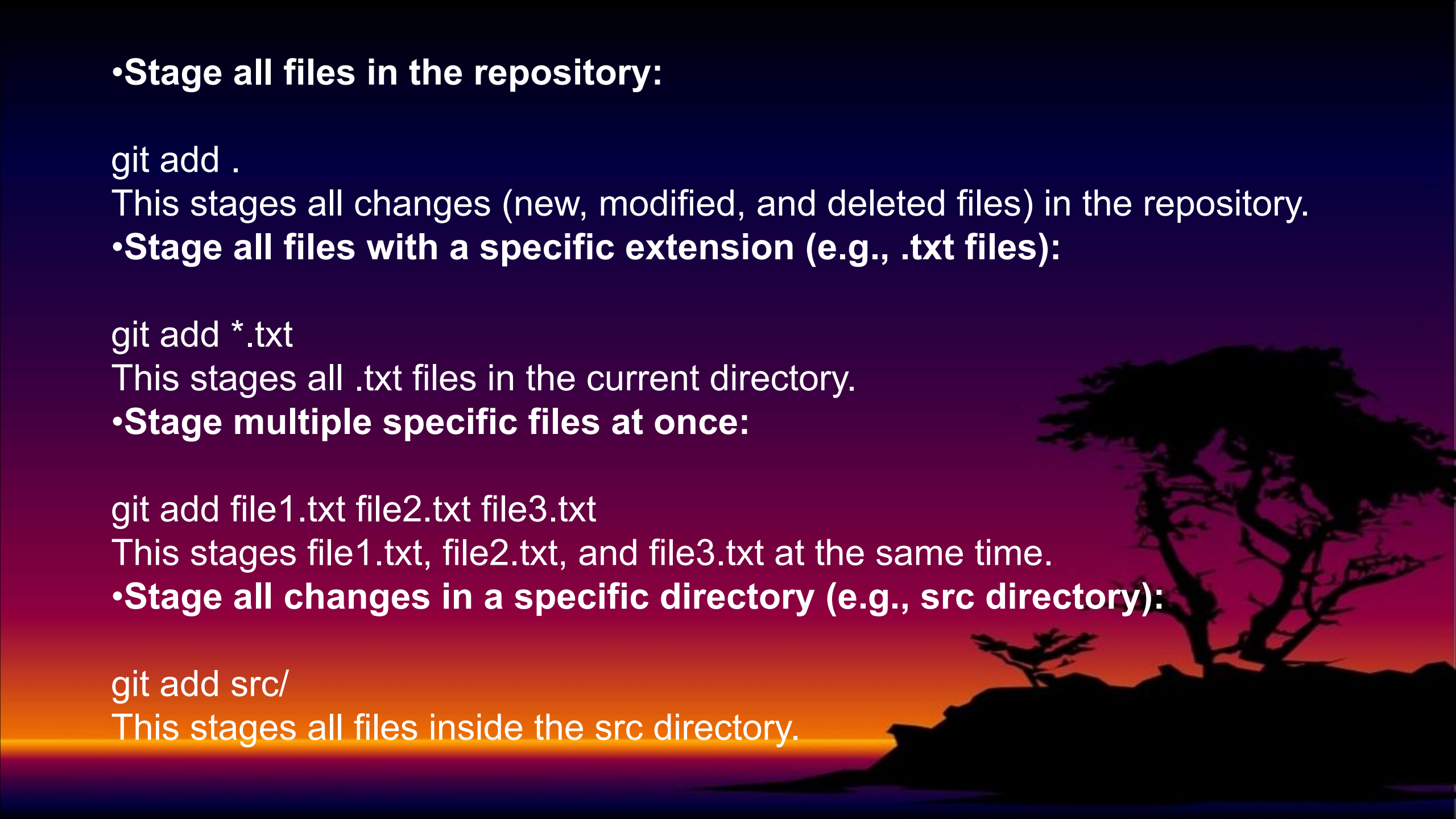
•**Stage all files in the repository:**

git add .
This stages all changes (new, modified, and deleted files) in the repository.
•**Stage all files with a specific extension (e.g., .txt files):**

git add *.txt
This stages all .txt files in the current directory.
•**Stage multiple specific files at once:**

git add file1.txt file2.txt file3.txt
This stages file1.txt, file2.txt, and file3.txt at the same time.
•**Stage all changes in a specific directory (e.g., src directory):**

git add src/
This stages all files inside the src directory.



- **Commit a file**

Once your files are staged, you can create a commit by using the git commit command. The commit message should be descriptive and concise, summarizing the changes you're making.

Best practice for commit message, follow these guidelines:

- Start the message with a capitalized verb in the imperative mood (e.g., "Add," "Fix," "Update").
- Keep the subject line (first line) to a maximum of 50-72 characters.
- Add a more detailed description in the body of the commit message, if necessary, using additional lines.

- To commit your changes with a message, execute the following command:

git commit -m "Your commit message here"

Replace "Your commit message here" with your actual commit message.

If you need to edit the commit message after committing, you can use the **git commit --amend** command.

Once you're done editing, save and close the text editor to finalize the changes. It's important to note that commit messages are essential for maintaining a clear history of your repository and understanding the changes made at a glance.

Operations on git log command

- **To see simplified list of commit**

git log command to view a simplified list of commits and a more detailed list of commits. The git log command is used to display the commit history of a Git repository.

Follow these steps: Open your command line interface or terminal. Navigate to the root directory of your Git repository. Execute the following command:

git log

- To see a more detailed list of commits, including the changes made in each commit, you can use additional options with the git log command. Here are a few commonly used options:
- git log -p: This option displays the commit history with the actual changes made in each commit, showing the diff for each commit.
- git log --stat: This option provides a concise summary of the changes made in each commit, including the number of files changed and the number of lines added or removed.
- git log --graph: This option visualizes the commit history as a text-based graph, showing the branching and merging of commits. You can combine these options as needed to get the level of detail you require. For example, git log -p --stat would show both the actual changes and a summary of each commit.

Manage branches

- A branch is a separate copy of the codebase that can be used for testing or development purposes.

Developers can create a new branch from the main codebase, make changes to the branch, and then merge those changes back into the main codebase when they are ready.

✓ **Operations on branches**

- Create a branch: To create a new branch in Git, you can use the following command:

git branch

- List branches: To list all the branches in your Git repository, you can use the following command:

git branch

This will display a list of all the local branches in your repository. The currently active branch will be marked with an asterisk (*).

- Delete local branch: To delete a local branch in Git, you can use the following command:

git branch -d

Replace with the name of the branch you want to delete.

- Delete remote branch: To delete a remote branch in Git, you can use the following command:

git push origin –delete <branch_name>

Replace with the name of the branch you want to delete on the remote repository.

- Switch branch:

To switch to a different branch in Git, you can use the following command:

git checkout <branch_name>

Replace with the name of the branch you want to switch to. This command will update your working directory to the specified branch.

- **Rename branch:** To rename a branch in Git, you can use the following command:

git branch -m <old_branch_name> <new_branch_name>

Replace with the current name of the branch you want to rename, and with the desired new name for the branch.