REPORT

Mar 4, 2025 P. Kurella

What is a staging area in Git?

In Git, a "staging area" is a temporary holding place where you selectively add changes to files from your working directory before committing them to your repository.

GIT Commands:

1. **`$ git config --global user.name "Your Name" :`** To enter the name/user ID of the user's github account.
2. **`$ git config --global user.email "your@email.com" :`** To enter the email id of the github account of the user.
3. **`$ git -- version:`** To check the version of the current running git on your personal computer.
4. **`$ git --global --list:`** To display the identities.
5. **`$ git init:`** This initializes a Git repo in your project folder.
6. **`$ git clone <repo-url>:`** To copy an existing repo from GitHub to your local system.
7. **`$ git status:`** Shows uncommitted changes.
8. **`$ git add <file>:`** Add a specific file.
9. **`$ git add . :`** Add all changes.
10. **`$ git commit -m "commit message":`** Saves your changes with a message.
11. **`$ git commit -a -m commit message":`** saves your new appended changes to an existing file.
12. **`$ git push origin main:`** Pushes changes to the main branch.
13. **`$ git push -u origin <branch-name>:`** Pushes the branch to the remote repository.
14. **`$ git remote add origin <repo-url>:`** Connects your git and github repo.
15. **`$ git remote -v:`** Shows the remote repositories linked to your project.
16. **`$ git branch:`** Lists all branches in the repository.
17. **`$ git branch <branch-name>:`** Creates a new branch.
18. **`$ git checkout <branch-name>:`** Switches to an existing branch.

19. **`$ git checkout -b <branch-name>:`** Creates a new branch and switches to it immediately.
20. **`$ git switch <branch-name>:`** Another way to switch branches (modern command).
21. **`$ git switch -c <branch-name>:`** Creates a new branch and switches to it.
22. **`$ git merge <branch-name>:`** Merges the specified branch into the current branch.
23. **`$ git pull origin <branch-name>:`** Fetches and merges changes from the remote repository.
24. **`$ git fetch:`** Downloads remote changes but does not merge them.
25. **`$ git log:`** Shows commit history.
26. **`$ git blame <file>:`** Shows who last modified each line of a file.
27. **`$ git branch -d <branch-name>:`** Deletes a merged branch locally.
28. **`$ git push origin --delete <branch-name>:`** Deletes a remote branch.
29. **`$ git diff:`** Shows differences between working directory and staging area.
30. **`$ git diff --staged:`** Shows differences between staged files and the last commit.

GIT Workflow:

Complete Git Workflow: From Initialization to Contribution

1. Initialize a Git Repository:

```
$ git init
```

- Creates a new Git repository in the current folder.
- Initializes a .git directory to start tracking changes.

2. Create & Add a New File:

```
$ touch index.html
```

- Creates a new file.
    - Check the status:

      ```
      $ git status
      ```

      Shows untracked files (files not yet tracked by Git).

- Stage the file:

  ```
  $ git add index.html
  ```

  - Moves the file to the **staging area**.

3. Commit the Changes:

```
$ git commit -m "Initial commit: Added index.html"
```

- Saves the staged changes with a commit message.

4. Connect to a Remote Repository (GitHub):

Create a new repository on **GitHub**, then link it to your local repo:

```
$ git remote add origin
https://github.com/your-username/repository-name.git
```

- To verify the remote:

  ```
  $ git remote -v
  ```

5. Push the Code to GitHub:

```
$ git push -u origin main
```

- Pushes the main branch to GitHub.
- -u sets origin/main as the default upstream branch.

6. Make More Changes & Commit Again

Edit index.html, then check the status:

```
$ git status
```

- Stage and commit the changes:
  ```
  $ git add .
  $ git commit -m "Updated index.html with new content"
  ```

- Push to GitHub:
  ```
  $ git push
  ```

7. Clone a Repository (Forking & Cloning):

Fork a Repo from GitHub

- Click "Fork" on the GitHub repo.
- Clone the Forked Repo
  `$ git clone https://github.com/your-username/forked-repo.git`

- Move into the project directory:

  `$ cd forked-repo`

- Check the remote:
  `$ git remote -v`

8. Pull the Latest Changes from the Original Repo:

Add Upstream Remote

`$ git remote add upstream`
`https://github.com/original-owner/original-repo.git`

- Fetch & merge changes:
  `$ git fetch upstream`
  `$ git merge upstream/main`

9. Pull Changes from GitHub:

`$ git pull origin main`

10. Create & Switch to a New Branch:

```
$ git branch feature-branch
$ git checkout feature-branch
```

or

`$ git switch -c feature-branch`

11. Merge a Branch into Main:

```
$ git checkout main
$ git merge feature-branch
```

12. Delete a Branch:

`$ git branch -d feature-branch`

- To delete a remote branch:

```
$ git push origin --delete feature-branch
```

13. Contribute Back to the Original Repo (Pull Request):

Push your changes to your forked repo:

```
$ git push origin feature-branch
```

- Go to GitHub, open your forked repo, and **create a Pull Request (PR)**.
- Once merged, delete the branch.

14. Undoing Mistakes in Git:

- Undo the Last Commit (But Keep Changes)
  ```
  $ git reset --soft HEAD~1
  ```

- Undo the Last Commit & Discard Changes
  ```
  $ git reset --hard HEAD~1
  ```

- Revert a Commit (Without Deleting It)
  ```
  $ git revert <commit-hash>
  ```

15. View History & Logs:

- To See Commit History
  ```
  $ git log --oneline --graph --all
  ```

- See Who Edited a File
  ```
  $ git blame index.html
  ```