## **Git Cheatsheet**

This cheatsheet provides a quick reference for commonly used Git commands.

**Basic Commands:**

* git init: Initializes a new Git repository in the current directory.
* git status: Shows the status of tracked files (modified, staged, etc.).
* git add <file>: Adds a file to the staging area for the next commit.
* git commit -m "<message>": Creates a commit with a descriptive message.
* git log: Shows the commit history of the repository.

**Branching and Merging:**

* git branch -a: Shows all branch.
* git branch <branch\_name>: Creates a new branch.
* git checkout <branch\_name>: Switches to a different branch.
* git merge <branch\_name>: Merges another branch into the current branch.
* git branch -d <branch\_name>: Deletes a branch (after merging or ensuring it's not needed).

**Remote Repositories:**

* git clone <url>: Clones a remote repository to your local machine.
* git remote add <name> <url>: Adds a remote repository to your local repository.
* git fetch <remote\_name>: Downloads the latest changes from the remote repository.
* git push <remote\_name> <branch\_name>: Pushes your local commits to the remote repository.

**Undoing Changes:**

* git checkout .: Discards all changes made to tracked files in the working directory.
* git checkout <file>: Discards changes made to a specific file.
* git reset HEAD <file>: Unstages a file from the staging area. (careful, removes changes from working directory too)
* git revert <commit\_hash>: Creates a new commit that reverses the changes introduced in a specific commit.

**Additional Tips:**

* Use git config --global --list to view your current Git configuration settings.
* Use git --help or git <command> --help for more detailed information on a specific command.

## **Git Feature Workflow**

Steps:

1. Clone and Update:  
   A developer clones the central repository to obtain the latest project code. They regularly run git pull to keep their local copy synchronized with the remote repository.
   * git clone <url>
2. or
   * git init: Initializes a new Git repository in the current directory.
   * git remote add <name> <url>: Adds a remote repository to your local repository.  
     Example: git remote add origin https://github.com/repo-name
3. Create a Feature Branch:  
   For each new feature or bug fix, the developer creates a new branch with a descriptive name (e.g., feature/add-search or fix/login-issue). This branch diverges from a stable branch (like develop or main).
   * git checkout -b <branch-name>
4. Develop in Isolation:  
   The developer works on their changes within the feature branch. They can commit their work incrementally with meaningful commit messages for better tracking.
   * git status: Shows the status of tracked files (modified, staged, etc.).
   * git add <file>: Adds a file to the staging area for the next commit.
   * git commit -m "<message>": Creates a commit with a descriptive message.
   * git log: Shows the commit history of the repository.
5. Pull Latest Changes:  
   Before pushing to remote repo, the developer must pull any updates from the main or the develop branch to ensure their own branch is up-to-date.
   * git checkout <branch-name>: Make sure you are on your branch (not main or develop).
   * git fetch origin: Fetch the latest changes from the main branch
   * git merge origin/main: Merge the main branch into your current branch
   * Resolve any conflicts that may arise during the merge.
   * git commit -m "Merged main branch into <current-branch>": Commit the merged changes
6. Push to your branch in remote repo and create a Pull Request:
   * git push origin feature/new-login
   * git push -u origin feature/new-login: for first push, use -u (set-upstream) flag to configure your local branch to track the remote branch of the same name
7. Merge to Develop Branch  
   Once the changes are deemed ready, the kead developer merges the feature branch into the develop branch, typically after code review or approval.
8. Delete Feature Branch (Optional):  
   After successful merging, the feature branch may be deleted locally and remotely as it's no longer needed for active development. However, some teams prefer to keep feature branches for future reference.
9. Merge Develop into Main Branch (Production):  
   When a set of features or bug fixes in develop are ready for deployment, they are merged into the main branch, marking them ready for release to production.

Git feature workflow is a branching model by creating separate branches for different features or tasks, which allows multiple developers to work on different aspects of the project simultaneously.

The flow is as follows:

1. Creates a new branch and switches to it.
2. This command adds a file to the staging area for the current branch. git add
3. git commit -m "" This command commits the changes in the staging area to the current branch.
4. git pull main

git push This command pushes the changes in the current branch to a remote repository.

git pull This command fetches and merges changes from a remote repository into the current branch.

git merge This command merges the changes from another branch into the current branch.

git branch -d This command deletes a branch.

git log This command shows the history of commits for the current branch.

git diff This command shows the differences between the current branch and another branch or commit.

Here's a simplified explanation of the git feature workflow:

Create a new branch: For each new feature or task, create a new branch from the main development branch (usually called 'master' or 'main'). This allows you to work on the new feature without affecting the main codebase.

Work on your feature branch: Make changes and commit them to your feature branch. You can push these changes to a remote repository so that other developers can access them.

Review and merge: Once your feature is complete, you can create a pull request to merge your changes back into the main development branch. This triggers a code review process, where other developers can review your changes and provide feedback.

Resolve conflicts: If there are any conflicts between your feature branch and the main branch, you'll need to resolve them before the merge can be completed.

Delete the feature branch: Once the merge is complete, you can delete the feature branch to keep your repository tidy.

The git feature workflow helps to keep your codebase organized and allows for better collaboration and code review. It also makes it easier to track and manage different features and tasks within a project.