Essential Git Functions

1. **Git Setup**
   - `git config --global user.name` *`name`* - sets a name for credit when reviewing version history
   - `git config --global user.email` *`email`* - sets an email association with history marker
2. **Repository Management**
   - `git init` – initializes a new Git repository.
   - `git clone` *`url`* - copies an existiing remote repository locally.
3. **Staging and Committing**
   - `git add` *`path`* – stages file(s) for commit.
   - `git commit -m "`*`message`*`"` – saves staged changes to the repository with a message.
   - `git status` – shows the current state of the working directory and staging area.
4. **Branching and Merging**
   - `git branch` – lists branches.
   - `git branch` *`name`* – creates a new branches.
   - `git checkout [`*`branch`*`]` – switches branches or restores working tree files.
   - `git merge` *`branch`* – merges another branch into the current branch.
   - `git switch` *`branch`* - switches to a branch.
5. **Remote Repositories**
   - `git remote add` *`repository url`* – connects local repo to a remote server.
   - `git push` – uploads commits to the remote branch.
   - `git pull` – fetches and merges changes from the remote.
   - `git fetch` – retrieves updates from remote without merging.
6. **History and Inspection**
   - `git log` – shows commit history.
   - `git diff` – shows changes between the working directory, staging area, or commits.
   - `git show [`*`commit`*`]` – displays a specific commit and its changes.
7. **Undoing Changes**
   - `git reset` – moves HEAD and optionally updates the staging area and working directory.
   - `git revert` – creates a new commit to undo changes from a previous commit.
   - `git restore` *`path`* – restores working tree files

Collaboration Practices

- Use branches for new features.
- Pull frequently and before modifications to stay updated.
- Commit often with clear messages.
- Resolve merge conflicts carefully.