Ojas Jawale

Git & GitHub Cheat-sheet

1. GIT BASICS

- `git init <directory>:` Create empty Git repo in specified directory.
- `git clone <repo>:` Clone repo located at <repo> onto local machine.
- `git config <user.name <name>:` Define author name to be used for all commits in current repo.
- `git add <directory>:` Stage all changes in <directory> for the next commit.
- `git commit -m "<message>" :` Commit the staged snapshot, but instead of launching a text editor, use <message> as the commit message.
- `git status :` List which files are staged, un-staged, and un-tracked.
- `git log :` Display the entire commit history using the default format.
- `git diff :` Show un-staged changes between your index and working directory.

---

2. UNDOING CHANGES

- `git revert <commit> :` Create new commit that undoes all of the changes made in <commit>, then apply it to the current branch.
- `git reset <file> :` Remove <file> from the staging area, but leave the working directory unchanged.
- `git clean -n :` Shows which files would be removed from working directory.

3. REWRITING GIT HISTORY

- `git commit --amend` : Replace the last commit with the staged changes and last commit combined.
- `git rebase <base>` : Rebase the current branch onto <base>. <base> can be a commit ID, branch name, a tag, or a relative reference to HEAD.
- `git reflog` : Show a log of changes to the local repository's HEAD.

---

4. GIT BRANCHES

- `git branch` : List all of the branches in your repo. Add a <branch> argument to create a new branch with the name <branch>.
- `git checkout -b <branch>` : Create and check out a new branch named <branch>.
- `git merge <branch>` : Merge <branch> into the current branch.

---

5. REMOTE REPOSITORIES

- `git remote add <name> <url>` : Create a new connection to a remote repo.
- `git fetch <remote> <branch>` : Fetches a specific <branch>, from the repo. Leave off <branch> to fetch all remote refs.
- `git pull <remote>` : Fetch the specified remote's copy of current branch and immediately merge it into the local copy.
- `git push <remote> <branch>` : Push the branch to <remote>, along with necessary commits and objects. Creates named branch in the remote repo if it doesn't exist.

6. GIT CONFIG

- `git config --global user.name <name>` : Define the author name to be used for all commits by the current user.
- `git config --global user.email <email>` : Define the author email to be used for all commits by the current user.
- `git config --global alias. <alias-name> <git-command>` : Create shortcut for a Git command. E.g. alias.glog "log --graph --oneline" will set "git glog" equivalent to "git log --graph --oneline.
- `git config --system core.editor <editor>` : Set text editor used by commands for all users on the machine. <editor> arg should be the command that launches the desired editor (e.g., vi).
- `git config --global --edit` : Open the global configuration file in a text editor for manual editing.

---

7. GIT LOG

- `git log -<limit>` : Limit number of commits by <limit>. E.g. "git log -5" will limit to 5 commits.
- `git log --oneline` : Condense each commit to a single line.
- `git log -p` : Display the full diff of each commit.
- `git log --stat` : Include which files were altered and the relative number of lines that were added or deleted from each of them.
- `git log --author="<pattern>"` : Search for commits by a particular author.
- `git log --grep="<pattern>"` : Search for commits with a commit message that matches <pattern>.

8. GIT DIFF

- `git diff HEAD` : Show difference between working directory and last commit.
- `git diff --cached` : Show difference between staged changes and last commit.

---

9. GIT RESET

- `git reset` : Reset staging area to match most recent commit, but leave the working directory unchanged.
- `git reset --hard` : Reset staging area and working directory to match most recent commit and overwrites all changes in the working directory.
- `git reset <commit>` : Move the current branch tip backward to <commit>, reset the staging area to match, but leave the working directory alone.
- `git reset --hard <commit>` : Same as previous, but resets both the staging area & working directory to match. Deletes uncommitted changes, and all commits after <commit>.

---

10. GIT REBASE

- `git rebase -I <base>` : Interactively rebase current branch onto <base>. Launches editor to enter commands for how each commit will be transferred to the new base.

---

11. GIT PULL

- `git pull --rebase <remote>` : Fetch the remote's copy of current branch and rebases it into the local copy.

12. GIT PUSH

- `git push <remote> --force` : Forces the git push even if it results in a non-fast-forward merge. Do not use the --force flag unless you're absolutely sure you know what you're doing.
- `git push <remote> --all` : Push all of your local branches to the specified remote.
- `git push <remote> --tags` : Tags aren't automatically pushed when you push a branch or use the --all flag. The --tags flag sends all of your local tags to the remote repo.

For more details read official docs.

Created by,
Ojas Jawale