git QUICK REFERENCE CARD

CONFIGURATION

git config --global user.name “<name>" ..... Sets the name you want attached to your commit transactions
git config --global user.email "<email address>" …. Sets the email you want attached to your commit transactions
git config --global color.ui auto ..... Enables helpful colorization of command line output
git config --system core.editor <editor> ….. Set text editor used by commands. <editor> is command to run editor
git config --global --edit ..... Open global configuration file in text editor for manual editing
git help <command> ..... Get help on the given command

CREATE REPOSITORIES -- INIT & CLONE

git init <project-name> ..... Creates a new local repository with the specified name
git clone <url> ..... Downloads a project and its entire version history
git clone ssh://user@domain.com/repo.git ..... Clone and existing repository

STATUS & LOG

git status ..... Shows the current state of the repository
git log ..... Lists all new or modified files to be committed
git log --stat ..... Include which files were altered and relative number of lines that were added/deleted
git log -p ..... Display full diff of each committed
git log -- <file> ..... Only display commits that have the specified file
git log --graph --decorate ..... --graph draws a text based graph of commits
..... --decorate adds names of branches or tags of commits shown

LOCAL CHANGES -- ADD, DIFF, & COMMIT

git add **.** ..... Add all current changes to the next commit
git add <file> ..... Snapshots the file in preparation for versioning
git reset <file> ..... Unstages the file, but preserve its contents
git diff ..... Shows file differences not yet staged
git diff --staged ..... Shows file differences between staging and the last file version
git commit ..... Commit previously staged changes
git commit -a ..... Commit all local chenges in tracked files
git commit -m "<descriptive message>" ..... Records file snapshots permanently in version history
git commit --amend ..... Change the last commit (Don't amend published commits!)
git reflog (--all) ..... Show log of changes to local repository's HEAD, --all shows all refs

GROUP CHANGES BRANCHES & TAGS

git branch ..... Lists all local branches in the current repository
git branch -av ..... List all existing branches
git branch <branch-name> ..... Creates a new branch on your current HEAD
git checkout <branch-name> ..... Switches to the specified branch and updates the working directory
git merge <branch> ..... Combines the specified branch’s history into the current branch
git branch -d <branch-name> ..... Deletes the specified branch
git checkout --track <remote/branch> ..... Create new tracking branch based on a remote branch
git tag <tag-name> ..... Mark the current commit with a tag

UPDATE & PUBLISH

git remote -v ..... List all currently configured remotes
git remote show <remote> ..... Show information about a remote
git remote add <shortname> <url> ..... Add new remote repository, named <remote>
git fetch <remote> ..... Download all changes from <remote>, but don't integrate into HEAD
git pull <remote> <branch> ..... Download changes and directly merge/integrate into HEAD
git push <remote> <branch> ..... Publish local changes on a remote
git branch -dr <remote/branch> ..... Delete a branch on a remote
git push --tags ..... Publish your tags

REMOVE FILES

git rm <file> ..... Deletes the file from the working directory and stages the deletion
git rm --cached <file> ..... Removes the file from version control but preserves the file locally
git mv <file-original> <file-renamed> ..... Changes the file name and prepares it for commit

SAVE FRAGMENTS

git stash ..... Temporarily stores all modified tracked files
git stash list ..... Lists all stashed change sets
git stash pop ..... Restores the most recently stashed files
git stash drop ..... Discards the most recently stashed change set

REDO COMMITS

git reset <commit> ..... Undoes all commits after <commit>, preserving changes locally
git reset --hard <commit> ..... Discards all history and changes back to the specified commit

REVIEW HISTORY

git log ..... Lists version history for the current branch, starting with newest
git log -p <file> ..... Show changes over time for a specific file
git log --follow <file> ..... Lists version history for a file, including renames
git diff <first-branch>...<second-branch> ..... Shows content differences between two branches
git show <commit> ..... Outputs metadata and content changes of the specified commit
git blame <file> ..... Who changes what and when in <file>

SUPPRESS TRACKING

.gitignore ..... Exclude temporary files and paths using .gitignore and suppress accidental versioning of files

```
*.log
build/
temp-*
```

git ls-files ..... Lists all files in this project
git ls-files --other ..... Lists other files in this directory
git ls-files --other --ignored --exclude-standard ..... Lists all ignored files in this project

SYNCHRONIZE CHANGES -- MERGE & REBASE

git merge <branch> ..... Merge <branch> into your current HEAD
git fetch <bookmark> ..... Downloads all history from the repository bookmark
git merge <bookmark>/<branch> ..... Combines bookmark's branch into current local branch
git push <alias> <branch> ..... Uploads all local branch commits to GitHub
git pull ..... Downloads bookmark history and incorporates changes
git rebase <branch> ..... Rebase your current HEAD into <brangh>. Don't rebase published commits!
git rebase --abort ..... Abort a rebase
git rebase --continue ..... Continue a rebase after resolving conflicts
git mergetool ..... Use your configured merge tool to solve conflicts
git add <resolved-file> ..... Use editor to manually solve conflicts and (after resolving) mark files as resolved
git rm <resolved-file> ..... Use editor to manually solve conflicts and (after resolving) mark files as resolved

UNDO

git reset --hard HEAD ..... Discard any local changes in your working directory
git checkout HEAD <file> ..... Discard local changes in a specific file-level
git revert <commit> ..... Revert commit (by producing a new commit with contrary changes)
git reset --hard <commit> ..... Reset HEAD pointer to previous commit …
..... & discard all changes as unstaged changes
git reset <commit> ..... … & preserve all changes as unstaged changes
git reset --keep <commit> ....and preserve uncommitted local changes
git clean -n ..... Shows which files would be removed from the working directory
git clean -f ..... Execute the clean