Getting Started with Git and GitHub

Module 2 Cheat Sheet: Git Commands and Managing GitHub Projects

Package/Method	Description	Code Example
git add	Used to move changes from the working directory to the staging area	`1. git add sample.md` Copied!
git add .	Allows to move the changed files into the staging area on GitHub repositories	`1. git add .` Copied!
git am	Used to apply patches emailed to the repository	`1. git am < patchfile.patch` Copied!
git branch	Allows to create an isolated environment within the repository to make changes	`1. git branch <new-branch>` Copied!
git checkout	Allows to see and change existing branches	`1. git checkout <existing-branch>` Copied!
git checkout main	Allows to switch to the main branch	`1. git checkout main` Copied!
git clone	Allows to create a copy of the remote repository	`1. git clone <repository-url>` Copied!
git commit	Allows you to take staged snapshots if changes and commit them to the project	`1. git commit -m "Your commit message here"` Copied!
git config --global user.email	Example 1: Sets a global email configuration for Git Example 2: Sets a global username configuration for Git	Example 1: `1. git config --global user.email "your.email@example.com"` Copied! Example 2: `1. git config --global user.name "Your Name"` Copied!
git daemon	Used to allow anonymous download from the repository	`1. git daemon --reuseaddr --verbose` Copied!
git diff	Helps others to review your code to identify and compare the changes	`1. git diff example.txt` Copied!
git fetch	Used to transfer the changes from the remote repo to your local repo	`1. git fetch <options> <remote name> <branch name>` Copied!
git fetch upstream/master	Used to grab upstream branches	`1. git fetch upstream master:upstream-master` Copied!
git format-patch	Generates or prepares e-mail submission if you adopt Linux kernel-style public forum workflow	`1. git format-patch -n <number_of_commits>` Copied!

Package/Method	Description	Code Example
git http-backend	Provides a server-side implementation of Git-over-HTTP, allowing both fetch and push services	1. 1 2. 2 3. 3 1. git clone --bare /path/to/repos/myrepo.git 2. cd myrepo.git 3. git update-server-info Copied!
git init	Used to clone an existing repository	1. 1 1. git init <directory> Copied!
git instaweb	Allows to set up web front-end to Git repositories	1. 1 1. git instaweb -p 8080 Copied!
git log	Enables to browse previous changes to a project	1. 1 1. git log -p filename Copied!
git merge	Used to merge changes in the active branch into another branch	1. 1 1. git merge feature_branch Copied!
git merge upstream/master	Merges changes from the 'upstream/master' branch to the current branch	1. 1 1. git merge upstream/master Copied!
git pull	Used to transfer the changes from the remote repo to your local repo, and merge them to a branch	1. 1 1. git pull origin main Copied!
git pull downstream	Pulls changes from a downstream repository, specifically from the master branch of that repository	1. 1 1. git pull downstream main Copied!
git pull upstream	Pulls changes from the "upstream" repository into the current branch	1. 1 1. git pull upstream main Copied!
git push	Used to push all the committed changes into the repository	1. 1 1. git push origin your_branch_name Copied!
git remote	A command to manage a set of tracked repositories	1. 1 1. git remote add upstream https://github.com/original/repo.git Copied!
git remote add origin <URL>	Adds a remote repository named "origin" with the specified URL	1. 1 1. git remote add origin https://github.com/yourusername/your-repo.git Copied!
git remote add upstream	Adds the original repository as a new remote repository labeled upstream	1. 1 1. git remote add upstream https://github.com/original/repo.git Copied!
git remote rename	The git remote rename command is followed by the name of the remote repository(origin) you want to rename and the new name(upstream) you want to give it	1. 1 1. git remote rename origin new-origin Copied!
git remote -v	Allows to view the remotes associated with the local repository	1. 1 1. git remote -v Copied!
git request-pull	Example 1: Creates a summary of changes for your upstream to pull Example 2: Generates a summary of pending changes for an email request	Example 1: 1. 1 1. git request-pull origin/main your-branch Copied!

Package/Method	Description	Code Example

		Example 2:
		`1. 1`
		`1. git request-pull <base> <head> <repository>`
		Copied!

git rerere	Reuses recorded resolution of previously resolved merge conflicts	`1. 1`
		`2. 2`
		`1. git rerere`
		`2. git rerere diff`
		Copied!

git reset	Undoes changes that were made to the files in your working directory	`1. 1`
		`1. git reset HEAD~1`
		Copied!

git revert	Used to undo botched commits	`1. 1`
		`1. git revert HEAD`
		Copied!

git send-email	Example 1: Sends your email submission without corruption by your MUA
	Example 2: Sends a collection of patches as emails
		Example 1:
		`1. 1`
		`2. 2`
		`1. git send-email --to=recipient@example.com`
		`2. path/to/patchfile.patch`
		Copied!
		Example 2:
		`1. 1`
		`2. 2`
		`1. git send-email --to recipient@example.com`
		`2. patches/*.patch`
		Copied!

git-shell	Used as a restricted login shell for shared central repository users	`1. 1`
		`1. sudo usermod -s /usr/bin/git-shell gituser`
		Copied!

git status	Allows to see the state of your working directory and the staged snapshot of the changes	`1. 1`
		`1. git status`
		Copied!

git version	Displays the current Git version installed on your system	`1. 1`
		`1. git --version`
		Copied!

git web	Provides a web front-end to Git repositories	`1. 1`
		`1. git instaweb --port=8080`
		Copied!

Skills Network