GIT Cheat Sheet

Use this handy GIT sheet to enhance your workflow.

1. Helping to Set up Your New Project:

- **git init**
  This command initializes a new git repository in your current working directory.
  Note: git init (project name)
  If (project name) is provided, git will create a new working directory with the (project name) and then initialize a repository inside it.

- **git add**
  This command moves your changes from the working folder/directory to the staging area. It adds all your directory files and prepares you to commit those changes to the official history.
  Note: git add.
  This command is often used to move all your changes from the working folder/directory to the staging area.
  Note: git add (file)
  This command is used to add a particular file and move that file to the staging area.

- **git commit**
  This command takes stages history when a user performs a git add and commits it to project history, which is your repository history. This command is always used after git add to create a simple workflow of git for all users.
  Note: git commit -m “your custom message”
  This command tracks the type of commit made by giving a meaningful custom message.

- **git remote**
  This command is used to administer remote connections. It can add, delete, and view connections to other repositories.
  Note: git remote add origin
  This command is used to create a new, empty remote repository on your remote server.

- **git push**
  This command is used to push the changes committed at staging from your local repository to your git remote repository. It lets you push multiple commits in the same git remote repository.
  Note: git push origin master/main
  This command is used to specify on which branch of the remote repository you want to upload the content of your local repository.

2. Downloading Existing Project:

git clone: This command is used to create a copy of an existing git repository.
Note: git clone (repository_url)
This command downloads a particular repository from that URL with its entire commit history from a remote repository.

3. Know Your Git Branches:

- **git branch**
  This command is a general branch tool for creating isolated development environments within a single repository.
  Note: git branch (branch_name)
  This command creates a new git branch regarding the current head.
- **git branch -a**
  This command lists all the local and remote branches in your repository.
- **git branch -r**
  This command is used to list all remote branches in your repository.
- **git checkout**
  This command is used to navigate between the existing branches. It is also used for checking commit history.
- **git branch - -delete [branchname]**
  This command is used to delete a particular local git branch.
- **git merge**
  This command is used to integrate changes from different branches.
  Note: git merge (branch_to_be_merged)
  This command merges the specified branch with your current working branch.

4. Everyday Know-How of Git:

- **git status**
  This command is used to display the status of the working directory and commit history. It will include all the detail about new, staged, and modified files. It will Provide current commit identifier, branch name, and changes after commit.
- **git log**
  This command provides several formatting options to see committed changes to your project.
- **git pull**
  This command is used to download a branch from a remote repository.
- **git reset**
  This command is used to undo the changes in the working directory/folder. It lets you clean up or remove the changes that have not been pushed to the public repository.

5. Git Configuration Options:

- **git config**
  This command is used to set configuration options for the git installations. Generally, used immediately after git installation on a new machine.
- **git config --global user.name “your_git_username”**
  This command sets the name that will be attached to your commits and tags.
- **git config --global user.email “you@example.com”**
  This command is used to set the email address that will be attached to your commits and tags.

6. Repository Synchronization:

- **git fetch <remote_repository_url>**
  This command lets you fetch changes from the given remote repository. It will notify you that changes are available to your remote repository but will not bring them to your local repository.
- **git pull <remote_repository_url>**
  This command lets you get the copy of your remote repository changes into your local repository, basically merging it.

7. Doing Changes:

- **git reset –hard**
  This command is used to throw away all your uncommitted changes. It will discard changes only in tracked files.
  Note: git reset [--hard] [target reference]
  This command switches the current branch to the target reference, leaving a difference as an uncommitted change. When --hard is used, all changes are discarded**.**

- **git revert**
  This command is used to undo the committed changes. If you find a wrong commit to your public repository, you can revert it from the repository.
  It creates a new commit that alters the wrong commit to your public repository.