

Git & GitHub: The Complete Walkthrough (with Branching Strategy)

Whether you're just starting out or looking to polish your Git & GitHub workflow, this guide is your one-stop shop for everything from initialization to advanced branching techniques. Let's break it all down in a practical, developer-friendly format.

What are Git & GitHub?

- **Git** is a version control system — like a time machine for your code.
- **GitHub** is a web platform that hosts Git repositories, adding collaboration tools like pull requests, issue tracking, and CI/CD pipelines.

Think of Git as your local brain, and GitHub as your team's collective memory in the cloud.

Getting Started: Installing and Configuring Git

Install Git (Linux)

```
sudo apt install git
```

Set your identity

```
git config --global user.name "My Name"
```

```
git config --global user.email "my@email.com"
```

This ensures every commit my make is properly credited.

```
(kali㉿kali)-[~]  
$ sudo apt install git  
git is already the newest version (1:2.45.2-1).  
git set to manually installed.
```

```
(kali㉿kali)-[~]  
$ git --version  
git version 2.45.2  
  
(kali㉿kali)-[~]  
$ git config --global user.name "Airborne167-hacker"  
  
(kali㉿kali)-[~]  
$ git config --global user.email "indirapaul033@gmail.com"  
  
(kali㉿kali)-[~]  
$ git config --global user.email "indirapaul003@gmail.com"  
  
(kali㉿kali)-[~]  
$ git config --global --list  
user.name=Airborne167-hacker  
user.email=indirapaul003@gmail.com
```

Initializing a New Git Project

mkdir my-portfolio

cd my-portfolio

git init

touch index.html

git add index.html

git commit -m "Initial commit"

just created my first Git repository and made my first commit.

```

(kali@kali)-[~]
$ mkdir my-portfolio167 #creates a new folder named my-portfolio

(kali@kali)-[~]
$ cd my-portfolio167 #changes into that folder

(kali@kali)-[~/my-portfolio167]
$ git init #create a hidden .git folder and sets up the current directory
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /home/kali/my-portfolio167/.git/

(kali@kali)-[~/my-portfolio167]
$ touch index.html #creates an empty HTML file.

(kali@kali)-[~/my-portfolio167]
$ git add index.html #This tells Git: "I want to track this file and prepare it for committing.

(kali@kali)-[~/my-portfolio167]
$ git commit -m "Initital commit" # saves the changes to Git's history with a message: "Initial commit
[master (root-commit) 7972232] Initital commit
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 index.html

(kali@kali)-[~/my-portfolio167]
$ git status #To confirm everything worked
On branch master
nothing to commit, working tree clean

(kali@kali)-[~/my-portfolio167]
$ git log #also check commit history
commit 7972232357877b589e736ff5fcbba706bdfa895 (HEAD -> master)
Author: Airborne167-hacker <indirapaul003@gmail.com>
Date:   Tue Jul 8 19:19:44 2025 +0530

    Initital commit

```

Connecting to GitHub

1st Step: Create a Repo on GitHub

Go to <https://github.com> → Click New repository

- Name: my-portfolio
- Leave it empty (don't add README, .gitignore, etc.)
- Click Create Repository

2nd step: Run These Commands in Kali Terminal Assuming I'm in my project folder (my-portfolio) and already ran git init, git add, and git commit.

Connect Local Repo to Remote

git remote add origin <https://github.com/myusername/my-portfolio.git>

git branch -M main

git push -u origin main

local code is now live on GitHub

```
(kali@kali)-[~/my-portfolio167]
$ git remote add origin https://github.com/Airborne167-hacker/my-portfolio167.git

(kali@kali)-[~/my-portfolio167]
$ git remote -v #confirm it worked
origin https://github.com/Airborne167-hacker/my-portfolio167.git (fetch)
origin https://github.com/Airborne167-hacker/my-portfolio167.git (push)

(kali@kali)-[~/my-portfolio167]
$ git branch -M main #Rename Branch to main

(kali@kali)-[~/my-portfolio167]
$ git push -u origin main #u sets origin main as the default upstream so next time you can just do git push
Username for 'https://github.com': Airborne167-hacker
Password for 'https://Airborne167-hacker@github.com':
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 223 bytes | 223.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Airborne167-hacker/my-portfolio167.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.

(kali@kali)-[~/my-portfolio167]
$ #Use GitHub Personal Access Token (instead of password),using HTTPS,can't use your GitHub password anymore

(kali@kali)-[~/my-portfolio167]
$ #Generate new token,Copy the token,When asks for password -> paste the token
```

Essential Git Commands (Use Every Day)

<i>Command</i>	<i>What It Does</i>	<i>Real Example</i>
<i>git status</i>	<i>Shows which files are staged or not</i>	<i>git status</i>
<i>git add</i>	<i>Stages all changed files</i>	<i>git add .</i>
<i>git commit -m "msg"</i>	<i>Saves changes with a commit message</i>	<i>git commit -m "Added navbar"</i>
<i>git log</i>	<i>Displays commit history</i>	<i>git log --oneline</i>
<i>git diff</i>	<i>Shows what has changed</i>	<i>git diff index.html</i>

<i>Command</i>	<i>What It Does</i>	<i>Real Example</i>
<i>git checkout -b feature-x</i>	<i>Creates and switches to a new branch</i>	<i>git checkout -b contact-form</i>
<i>git merge branch-name</i>	<i>Merges another branch into current</i>	<i>git merge contact-form</i>
<i>git push origin main</i>	<i>Sends your changes to GitHub (remote)</i>	<i>git push origin main</i>
<i>git pull origin main</i>	<i>Pulls latest code from GitHub</i>	<i>git pull origin main</i>
<i>git fetch origin</i>	<i>Downloads latest changes (doesn't merge)</i>	<i>git fetch origin</i>
<i>git init</i>	<i>Initializes a new Git repository in your folder</i>	<i>git init</i>

Use *git pull* when you want to get and apply the latest updates

Use *git fetch* if you want to review updates before merging them

```

(kali@kali)-[~/my-portfolio167/my-project/my-project1/my-project167]
$ git log --oneline #short view
bbaa1eb (HEAD → master) Initial commit

SimpleScre...
(kali@kali)-[~/my-portfolio167/my-project/my-project1/my-project167]
$ git diff index.html # Shows what code has changed (not yet staged or committed)

(kali@kali)-[~/my-portfolio167/my-project/my-project1/my-project167]
$ git checkout -b contact-form #Creates and switches to a new branch
Switched to a new branch 'contact-form'

```

```

(kali㉿kali)-[~/my-portfolio167/my-project/my-project1/my-project167]
$ git branch
* contact-form
  master

(kali㉿kali)-[~/my-portfolio167/my-project/my-project1/my-project167]
$ git checkout master
Switched to branch 'master'

(kali㉿kali)-[~/my-portfolio167/my-project/my-project1/my-project167]
$ git checkout -b main #If your branch is main but missing, it might be called master instead.
Switched to a new branch 'main'

(kali㉿kali)-[~/my-portfolio167/my-project/my-project1/my-project167]
$ git merge contact-form #Merges another branch into the current one
Already up to date.

(kali㉿kali)-[~/my-portfolio167/my-project/my-project1/my-project167]
$ git branch -a #Bonus Tip: See All Branches (including remote)
  contact-form
* main
  master

```

```

(kali㉿kali)-[~/my-project167 ..]
$ git remote add origin https://github.com/Airborne167-hacker/my-project167 ... git

(kali㉿kali)-[~/my-project167 ..]
$ git remote -v #confirm it worked
origin https://github.com/Airborne167-hacker/my-project167 ... git (fetch)
origin https://github.com/Airborne167-hacker/my-project167 ... git (push)

(kali㉿kali)-[~/my-project167 ..]
$ git branch -M main #Rename Branch to main

(kali㉿kali)-[~/my-project167 ..]
$ git push -u origin main #-u sets origin main as the default upstream so next time you can just do git push
Username for 'https://github.com': Airborne167-hacker
Password for 'https://Airborne167-hacker@github.com':
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 222 bytes | 222.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Airborne167-hacker/my-project167 ... git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.

```

```

(kali㉿kali)-[~/my-project167 ..]
$ git pull origin main #Pulls latest changes from GitHub's main into your local main
From https://github.com/Airborne167-hacker/my-project167..
 * branch            main           -> FETCH_HEAD
Already up to date.

(kali㉿kali)-[~/my-project167 ..]
$ git fetch origin #Fetches changes from GitHub without merging

(kali㉿kali)-[~/my-project167 ..]
$ git merge origin/main # can manually merge or inspect
Already up to date.

```

Git Branching : A branch lets you work on features independently

Creating & Switching Branches

git checkout -b feature/login

Make changes

git add

git commit -m "Add login feature"

git checkout main

git merge feature/login

just worked in isolation, and then brought changes back into the main codebase.

Git Branching Strategy: Feature-Driven Workflow

A simple, powerful Git branching model:

main



Main Branch (main)

- *Always production-ready*
- *Protected (no direct commits)*

Development Branch (dev)

- *Integration branch for all features*

Feature Branches (feature/*)

- *One branch per feature*

- Merge into dev when done

Hotfix Branches (hotfix/*)

- Emergency fixes off main

Use Pull Requests (PRs) for code reviews before merging to main.

Viewing Changes & Undoing Mistakes

git log --oneline

git show <commit-id>

git diff

git restore file.txt # Undo file changes

git reset HEAD~1 # Undo last commit (keep changes)

git revert <commit-id> # Undo via new commit (Creates a **new commit** that undoes the changes of a previous commit ---safe for shared branches)

```
(kali@kali)-[~/my-project167 ..]
└─$ git log --oneline #short view
12a0f47 (HEAD → main, origin/main) Initial commit

(kali@kali)-[~/my-project167 ..]
└─$ git show 12a0f47
commit 12a0f4768501be1b327b879aadcd90c053b911ef (HEAD → main, origin/main)
Author: Airborne167-hacker <indirapaul003@gmail.com>
Date:   Wed Jul 9 01:34:39 2025 +0530

    Initial commit

diff --git a/index.html b/index.html
new file mode 100644
index 0000000..e69de29

(kali@kali)-[~/my-project167 ..]
└─$ git diff #Show unstaged changes

(kali@kali)-[~/my-project167 ..]
└─$ git diff --staged #Show staged changes

(kali@kali)-[~/my-project167 ..]
└─$ git diff index.html #Show changes in a specific file

(kali@kali)-[~/my-project167 ..]
└─$ git restore index.html # This will erase all unsaved changes in index.html
```



```

(kali@kali)-[~/my-project167 .. ]
$ git log
commit 12a0f4768501be1b327b879aadcd90c053b911ef (HEAD → main, origin/main)
Author: Airborne167-hacker <indirapaul003@gmail.com>
Date: Wed Jul 9 01:34:39 2025 +0530

    Initial commit

(kali@kali)-[~/my-project167 .. ]
$ git reset #This will unstage files (undo git add) but keep the changes in your working directory

(kali@kali)-[~/my-project167 .. ]
$ git init
Reinitialized existing Git repository in /home/kali/my-project167 ../.git/

(kali@kali)-[~/my-project167 .. ]
$ echo "test" > file.txt

(kali@kali)-[~/my-project167 .. ]
$ git add file.txt

(kali@kali)-[~/my-project167 .. ]
$ git commit -m "first commit"
[main 0c4b1b6] first commit
1 file changed, 1 insertion(+)
create mode 100644 file.txt

(kali@kali)-[~/my-project167 .. ]
$ git log
commit 0c4b1b603cdababe916da6059f37827d7e174cdc (HEAD → main)
Author: Airborne167-hacker <indirapaul003@gmail.com>
Date: Wed Jul 9 02:34:02 2025 +0530

    first commit

commit 12a0f4768501be1b327b879aadcd90c053b911ef (origin/main)
Author: Airborne167-hacker <indirapaul003@gmail.com>
Date: Wed Jul 9 01:34:39 2025 +0530

    Initial commit

```

```

(kali@kali)-[~/my-project167 .. ]
$ git reset HEAD~1

(kali@kali)-[~/my-project167 .. ]
$ #Uncommits the last commit, but keeps your changes in the working directory

(kali@kali)-[~/my-project167 .. ]
$ git revert 12a0f47
[main b348ba8] Revert "Initial commit"
1 file changed, 0 insertions(+), 0 deletions(-)
delete mode 100644 index.html

```

Advanced: Rewriting History & Stashing

<i>git commit --amend</i>	<i># Edit last commit</i>
<i>git rebase <branch></i>	<i># Replay commits over another branch</i>
<i>git stash -u</i>	<i># Temporarily shelve changes [If I have untracked files (like new.js or .txt files), and want to stash them]</i>
<i>git stash pop</i>	<i># Reapply stashed work</i>

```
(kali㉿kali)-[~/my-project167 .. ]
$ touch forget-file.js

(kali㉿kali)-[~/my-project167 .. ]
$ git add forget-file.js

(kali㉿kali)-[~/my-project167 .. ]
$ git commit --amend #Replaces the last commit without creating a new one
[main 46181be] Revert "Initital commit"
Date: Wed Jul 9 02:37:09 2025 +0530
2 files changed, 1 insertion(+)
create mode 100644 file.txt
rename index.html => forget-file.js (100%)

(kali㉿kali)-[~/my-project167 .. ]
$ #Avoid amending commits that have already been pushed/shared
```

```
(kali㉿kali)-[~/my-project167 .. ]
$ git branch
* main

(kali㉿kali)-[~/my-project167 .. ]
$ git checkout main #This reapplies commits from feature-x on top of main
Already on 'main'
Your branch and 'origin/main' have diverged,
and have 1 and 1 different commits each, respectively.
(use "git pull" if you want to integrate the remote branch with yours)

(kali㉿kali)-[~/my-project167 .. ]
$ git rebase main #Keeps history linear (unlike merge)
Current branch main is up to date.

(kali㉿kali)-[~/my-project167 .. ]
$ git stash -u #-u means "include untracked files"
No local changes to save

(kali㉿kali)-[~/my-project167 .. ]
$ git status
On branch main
Your branch and 'origin/main' have diverged,
and have 1 and 1 different commits each, respectively.
(use "git pull" if you want to integrate the remote branch with yours)

nothing to commit, working tree clean

(kali㉿kali)-[~/my-project167 .. ]
$ echo "change" >> app.js
```

```

(kali㉿kali)-[~/my-project167..]
└─$ git status
On branch main
Your branch and 'origin/main' have diverged,
and have 1 and 1 different commits each, respectively.
(use "git pull" if you want to integrate the remote branch with yours)

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    app.js

nothing added to commit but untracked files present (use "git add" to track)

(kali㉿kali)-[~/my-project167..]
└─$ touch app.js

(kali㉿kali)-[~/my-project167..]
└─$ git add app.js

(kali㉿kali)-[~/my-project167..]
└─$ git status
On branch main
Your branch and 'origin/main' have diverged,
and have 1 and 1 different commits each, respectively.
(use "git pull" if you want to integrate the remote branch with yours)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   app.js

(kali㉿kali)-[~/my-project167..]
└─$ git stash ## clean now
Saved working directory and index state WIP on main: 46181be Revert "Initital commit"

(kali㉿kali)-[~/my-project167..]
└─$ git stash list #see saved stash
stash@{0}: WIP on main: 46181be Revert "Initital commit"

```

```

(kali㉿kali)-[~/my-project167..]
└─$ git stash apply #bring the change back
On branch main
Your branch and 'origin/main' have diverged,
and have 1 and 1 different commits each, respectively.
(use "git pull" if you want to integrate the remote branch with yours)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   app.js

```

```

(kali㉿kali)-[~/my-project167.. ]
└─$ git stash pop #get changes back, even after switching branche
On branch main
Your branch and 'origin/main' have diverged,
and have 1 and 1 different commits each, respectively.
(use "git pull" if you want to integrate the remote branch with yours)

Changes to be committed:
  (use "git restore --staged <file> ..." to unstage)
        new file:   app.js

Dropped refs/stash@{0} (f1726247688216cb61cf54a0a2a4a1049d85c750)

```

Team Collaboration with GitHub

<i>GitHub Feature</i>	<i>Purpose</i>
<i>Pull Requests</i>	<i>Propose & review code changes</i>
<i>Issues</i>	<i>Track bugs and tasks</i>
<i>Projects</i>	<i>Kanban boards for planning</i>
<i>Wiki</i>	<i>Share documentation</i>
<i>GitHub Pages</i>	<i>Host static sites</i>
<i>GitHub Actions</i>	<i>Automate testing & deployment</i>

Real-World Workflow Example

git checkout -b feature/search-bar

Make changes

git add .

git commit -m "Add search bar"

git push origin feature/search-bar

Go to GitHub → Open a Pull Request → Team reviews → Merge to dev or main.

Git Cheat Sheet

Basics

git init

git add .

git commit -m "message"

git status

git log

Branching

git checkout -b feature-x

git merge feature-x

git branch -d feature-x (Use after merging the branch)

```
(kali㉿kali)-[~/my-project167 .. ]
$ git checkout -b contact-form
Switched to a new branch 'contact-form'
```

```
(kali㉿kali)-[~/my-project167 .. ]
$ git branch
* contact-form
  feature-x
  feature-x1
  feature-x2
  main
  main1
  master
```

```
(kali㉿kali)-[~/my-project167 .. ]
$ git checkout master
A       app.js
Switched to branch 'master'
```

```

(kali㉿kali)-[~/my-project167..]
└─$ git merge contact-form #Merges another branch into the current one
Already up to date.

(kali㉿kali)-[~/my-project167..]
└─$ git branch -D contact-form #delete it
Deleted branch contact-form (was 46181be).

(kali㉿kali)-[~/my-project167..]
└─$ git branch
  feature-x
  feature-x1
  feature-x2
  main
  main1
* master

```

Remote

git remote add origin <url>

git push origin main

git pull origin main

Undo

git reset --soft HEAD~1

git revert <commit>

1. Always branch from dev or main

2. Never commit directly to main

3. Use .gitignore to exclude unwanted files

FOR GITIGNORE

```

(kali㉿kali)-[~]
$ mkdir github11

(kali㉿kali)-[~]
$ cd github11

(kali㉿kali)-[~/github11]
$ git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:     git config --global init.defaultBranch <name>
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:     git branch -m <name>
Initialized empty Git repository in /home/kali/github11/.git/

(kali㉿kali)-[~/github11]
$ touch .gitignore

(kali㉿kali)-[~/github11]
$ vim .gitignore

```

```

# Ignore node_modules
node_modules/

# Ignore compiled Python files
*.pyc
__pycache__/

# Ignore log files
*.log

# Ignore system files
.DS_Store
Thumbs.db

# Ignore build files
dist/
build/

```



```

(kali㉿kali)-[~/github11]
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore

nothing added to commit but untracked files present (use "git add" to track)

(kali㉿kali)-[~/github11]
$ git commit -m "Removed ignore file"
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore

nothing added to commit but untracked files present (use "git add" to track)

(kali㉿kali)-[~/github11]
$ git add .gitignore

(kali㉿kali)-[~/github11]
$ git commit -m "Add .gitignore file"
[master (root-commit) c6cc6fb] Add .gitignore file
1 file changed, 18 insertions(+)
create mode 100644 .gitignore

(kali㉿kali)-[~/github11]
$ git remote -v #confirm it worked

(kali㉿kali)-[~/github11]
$ git log
commit c6cc6fbddcaa0f8cf7650a08d2820434c03a4151 (HEAD -> master)
Author: Airborne167-hacker <indirapaul003@gmail.com>
Date:   Wed Jul 9 17:54:33 2025 +0530

    Add .gitignore file

```

```

(kali@kali)-[~/github11]
$ git remote add origin https://github.com/Airborne167-hacker/github11.git

(kali@kali)-[~/github11]
$ git remote -v #confirm it worked
origin https://github.com/Airborne167-hacker/github11.git (fetch)
origin https://github.com/Airborne167-hacker/github11.git (push)

(kali@kali)-[~/github11]
$ git branch -M main #Rename Branch to main

(kali@kali)-[~/github11]
$ git push -u origin main #-u sets origin main as the default upstream so next time you can just do git push
Username for 'https://github.com': Airborne167-hacker
Password for 'https://Airborne167-hacker@github.com':
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 345 bytes | 345.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Airborne167-hacker/github11.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.

(kali@kali)-[~/github11]
$ ls -l
total 0

(kali@kali)-[~/github11]
$ echo "my_secret_value" > secret.txt

(kali@kali)-[~/github11]
$ ls -l
total 4
-rw-rw-r-- 1 kali kali 16 Jul  9 17:59 secret.txt

(kali@kali)-[~/github11]
$ echo "secret.txt" >> gitignore

```

```

(kali㉿kali)-[~/github11]
$ cat .gitignore
# Ignore node_modules
node_modules/

# Ignore compiled Python files
*.pyc
__pycache__/

# Ignore log files
*.log

# Ignore system files
.DS_Store
Thumbs.db

# Ignore build files
dist/
build/

(kali㉿kali)-[~/github11]
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file> ..." to include in what will be committed)
  gitignore
  secret.txt

nothing added to commit but untracked files present (use "git add" to track)

(kali㉿kali)-[~/github11]
$ ls -l
total 8
-rw-rw-r-- 1 kali kali 11 Jul  9 18:00 gitignore
-rw-rw-r-- 1 kali kali 16 Jul  9 17:59 secret.txt

```

```

(kali㉿kali)-[~/github11]
$ echo "secret.txt" >> .gitignore

(kali㉿kali)-[~/github11]
$ cat .gitignore
# Ignore node_modules
node_modules/

# Ignore compiled Python files
*.pyc
__pycache__/_

# Ignore log files
*.log

# Ignore system files
.DS_Store
Thumbs.db

# Ignore build files
dist/
build/
secret.txt

(kali㉿kali)-[~/github11]
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   .gitignore

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        gitignore

no changes added to commit (use "git add" and/or "git commit -a")

```

Untracked files: (use "git add <file>..." to include in what will be committed)

gitignore

[That means Git is not tracking secret.txt anymore — success]

4. Write clear, descriptive commit messages

5. Pull before you push

Git & GitHub is about understanding how version control makes your development process safer, smarter, and more collaborative.