**🧠 Complete Git Mastery Notes: From Noob to Legend**

**🌱 Stage 1: Git Baby Steps (Noob Level)**

**🎯 Goal:**

Understand the **basic building blocks** of Git and how it works locally.

**🔸 Core Concepts**

| **Concept** | **Description** |
| --- | --- |
| Git | A distributed version control system. It tracks changes in source code and allows multiple developers to collaborate. |
| Git vs GitHub | Git is local version control; GitHub is a cloud-based hosting service for Git repositories. |
| Working Directory | The actual files/folders on your machine. |
| Staging Area | A temporary area where changes are listed before committing. |
| Local Repository | The Git database on your machine. |
| Commit | A snapshot of changes with a message. |

**🔹 Commands with Examples**

# Set your name/email once

git config --global user.name "Tony Stark"

git config --global user.email "tony@starkindustries.com"

# Start a Git repo

git init

# See file status

git status

# Track a file

git add index.html

# Track everything

git add .

# Save changes with a message

git commit -m "Initial commit"

# Check history

git log

**🧪 Example**

You’re working on a Python script:

# calculator.py

print("Hello Calculator")

You git init, add, and commit this file. Now Git remembers this version. If you make changes and commit again, you can **roll back** to this point anytime.

**🌿 Stage 2: Git Garden (Beginner ➝ Intermediate)**

**🎯 Goal:**

Learn to **connect to GitHub**, use **branches**, and understand basic collaboration.

**🔸 Core Concepts**

| **Concept** | **Description** |
| --- | --- |
| Remote | A Git repo hosted on a service like GitHub |
| origin | Default name for your GitHub repo |
| push | Send changes from local to GitHub |
| pull | Fetch and merge changes from GitHub |
| Branch | A separate line of work |
| Merge | Combine branches |
| .gitignore | File that lists things Git should ignore |

**🔹 Commands with Examples**

# Add GitHub repo as remote

git remote add origin https://github.com/yourusername/repo.git

# Upload local branch

git push -u origin main

# Fetch & merge updates from GitHub

git pull origin main

# Create and switch to a new branch

git checkout -b feature/header

# Merge into main

git checkout main

git merge feature/header

**🧪 Branch Naming Convention**

feature/header # Working on a new UI header

bugfix/login-issue # Fixing a login error

hotfix/typo # Immediate fix for typo in prod

**🧪 Example**

You create a feature/header branch, work on it, commit your changes, and merge it into main. Git helps you avoid conflict by isolating work into branches.

**🌳 Stage 3: Git Tree Master (Intermediate ➝ Pro)**

**🎯 Goal:**

**Clean your commit history**, use **advanced commands**, and **recover from mistakes**.

**🔸 Core Concepts**

| **Concept** | **Description** |
| --- | --- |
| Rebase | Reapply commits on top of another branch |
| Interactive Rebase | Edit/squash/reword commits |
| Stash | Temporarily save changes without committing |
| Amend | Change your last commit |
| Tag | Label specific commits (v1.0.0) |
| Semantic Commits | Naming convention for clean commit messages |

**🔹 Commands with Examples**

# Rebase feature branch onto latest main

git checkout feature/expense

git rebase main

# Interactive rebase last 3 commits

git rebase -i HEAD~3

# Temporarily stash changes

git stash

git stash pop

# Modify last commit

git commit --amend

# Tag a version

git tag v1.0.0

git push origin v1.0.0

**🧪 Example**

You made 5 messy commits:

fix: bug

update text

add function

remove logs

final changes

Rebase interactively and squash into one:

feat: Add user profile feature

Now your Git history is clean for PRs.

**🌲 Stage 4: Git Forest (Pro Level)**

**🎯 Goal:**

**Automate**, **optimize**, and **enforce team standards** with advanced tools.

**🔸 Core Concepts**

| **Concept** | **Description** |
| --- | --- |
| Git Aliases | Shortcuts for commands |
| Git Hooks | Scripts that run before/after Git events |
| Husky | Modern Git hook manager for JS projects |
| GitHub Actions | Cloud automation tool |
| Squashing | Merge multiple commits into one clean one |

**🔹 Git Aliases**

git config --global alias.co checkout

git config --global alias.cm commit

git config --global alias.st status

**🔹 Husky + Prettier + Lint Setup**

npx husky-init && npm install

npm install prettier eslint --save-dev

npx husky add .husky/pre-commit "npm run lint"

**🔹 GitHub Actions Example**

Create .github/workflows/ci.yml:

name: CI Test

on: [push]

jobs:

test:

runs-on: ubuntu-latest

steps:

- uses: actions/checkout@v2

- run: npm install

- run: npm test

**🧪 Example**

Before every push, Husky checks for lint errors. GitHub Actions auto-tests your React app on every PR. You prevent broken code from reaching main.

**🌌 Stage 5: Git Absolute Legend (Advanced Guru)**

**🎯 Goal:**

Master **Git internals**, **advanced debugging**, and **open source contribution**.

**🔸 Core Concepts**

| **Concept** | **Description** |
| --- | --- |
| Git Internals | Understand blobs, trees, commits |
| Bisect | Pinpoint the commit that caused a bug |
| Blame | Find out who edited what |
| Fork → PR | Open source contribution flow |
| Custom Scripts | Git automation (tags, changelogs, etc.) |

**🔹 Advanced Commands**

# Git internals

git cat-file -p HEAD

# Git bisect

git bisect start

git bisect bad

git bisect good <commit>

git bisect reset

# Git blame

git blame src/App.js

**🔹 Open Source Flow**

# Fork repo

git clone https://github.com/some-project.git

# Create a branch

git checkout -b fix/bug-readme

# Push and create pull request

git push origin fix/bug-readme

**🔹 Release Script (Bash)**

#!/bin/bash

VERSION=$1

git tag -a "v$VERSION" -m "Release v$VERSION"

git push origin "v$VERSION"

Run it:

./release.sh 3.0.0

**🧪 Example**

You find a typo in an open source project. You fork it, fix it on a new branch, push, and create a Pull Request (PR) with a proper commit message and reference to the issue.

**🏁 Final Recap: Git Stages Summary**

| **Stage** | **Name** | **Goal** | **You Learned** |
| --- | --- | --- | --- |
| 1 | Git Baby Steps | Local Git usage | Init, add, commit, status, log |
| 2 | Git Garden | GitHub and branches | Remote, push/pull, merge, conflicts |
| 3 | Git Tree Master | History control | Rebase, stash, amend, squash |
| 4 | Git Forest | Automation | Aliases, Husky, GitHub Actions |
| 5 | Git Absolute Legend | Internals + open source | Bisect, blame, internals, PRs |