Complete Git Beginner-Friendly Notes (Interview Ready)

PHASE 1 – INTRODUCTION

What is VCS?

- VCS = Version Control System
- Saves different versions of code
- Tracks who changed what
- Allows teamwork and rollback

Why We Need VCS?

- Prevents overwriting files
- Maintains full history
- Safe collaboration
- Easy backup and comparison

Types of VCS

- Centralized VCS: CVS, SVN, Perforce (single central server)
- Distributed VCS: Git, BitKeeper (every developer has full copy)
- Git is Distributed VCS

PHASE 2 – INSTALLATION (Ubuntu)

- sudo apt-get update
- sudo apt-get install git
- Verify: which git, git --version
- Uninstall: sudo apt-get remove git

PHASE 3 – BASIC CONFIGURATION

- git config --global user.name "YourName"
- git config --global user.email "your@email.com"
- git config --global credential.helper cache
- git config --list
- Configuration file: ~/.gitconfig

PHASE 4 – GIT ARCHITECTURE (Very Important)

- Working Directory – where you create/modify files
- Staging Area – temporary area before commit (git add)
- Local Repository (.git) – created using git init
- Remote Repository – GitHub server copy

PHASE 5 – CREATING REPOSITORY

- Clone: git clone
- Local Repo: mkdir project → git init → ls -a (see .git)

BASIC WORKFLOW

- git status – check modified/untracked files
- git add filename or git add .
- git commit -m "message"
- git log / git log --oneline
- git remote add origin
- git push -u origin master

GIT DIFF

- git diff – compare unstaged changes
- git diff --staged – compare staged changes
- git diff master origin/patch1 – compare branches

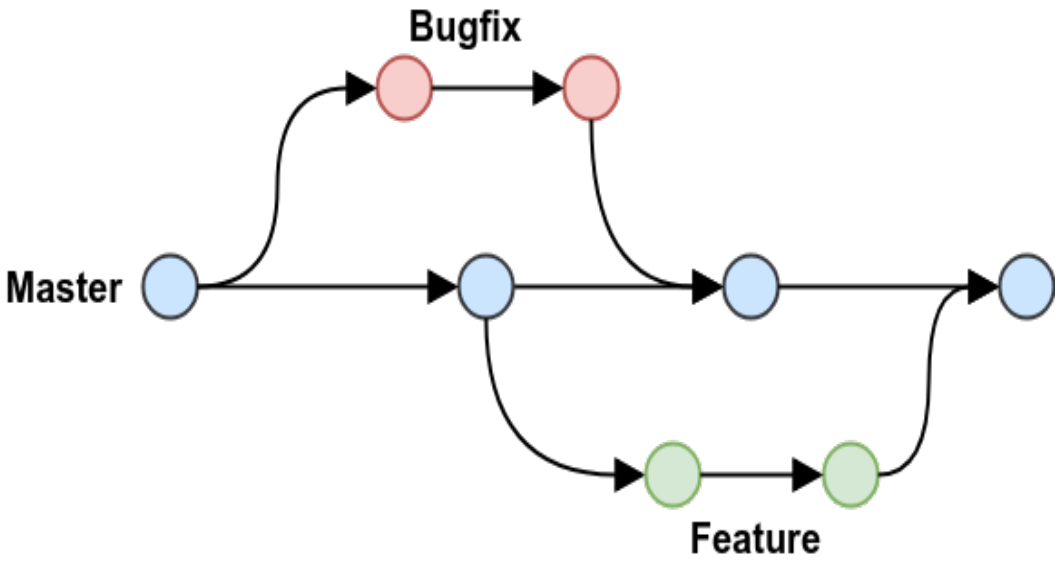
GIT FETCH & PULL

- git fetch – download only (no merge)
- git fetch origin
- git branch -r – see remote branches
- git pull – download + merge

BRANCHING

- Branch = separate line of development
- git branch dev
- git checkout dev
- git checkout -b dev
- git branch / -r / -a
- git push -u origin dev

Git Branching Diagram Example:



MERGING

- Switch to target branch first
- git checkout master
- git merge dev
- Fast-forward merge (no conflict)
- Merge conflict occurs when same line modified differently
- Resolve conflict → git add → git commit → git push

RESET (Undo Changes)

- git reset HEAD filename (unstage)
- git reset --soft HEAD~1 (keep changes)
- git reset --hard HEAD~1 (delete changes)

DELETE & RENAME

- git rm filename → git commit
- git mv old new → git commit

HEAD

- HEAD is a pointer
- Points to current branch
- Points to latest commit

PRACTICE TASKS

- Create repo → Add → Commit → Push
- Create branch → Add change → Merge
- Create conflict → Resolve
- Compare branches using git diff

FINAL COMPLETE FLOW

- Working Directory → git add → Staging Area
- Staging Area → git commit → Local Repository (.git)
- Local Repository → git push → Remote Repository (GitHub)

INTERVIEW READY SUMMARY

- Git Areas: Working Directory, Staging Area, Local Repo, Remote Repo
- Important Commands: git init, clone, add, commit, push, pull, fetch, branch, merge, diff