Date: 20.06.2025

Git

Contents

1. What is Git?

Git is a fast, scalable, distributed revision control system with an unusually rich command set that provides both high-level operations and full access to internals.

Git is an Open Source project covered by the GNU General Public License version 2 (some parts of it are under different licenses, compatible with the GPLv2). It was originally written by Linus Torvalds with help of a group of hackers around the net.

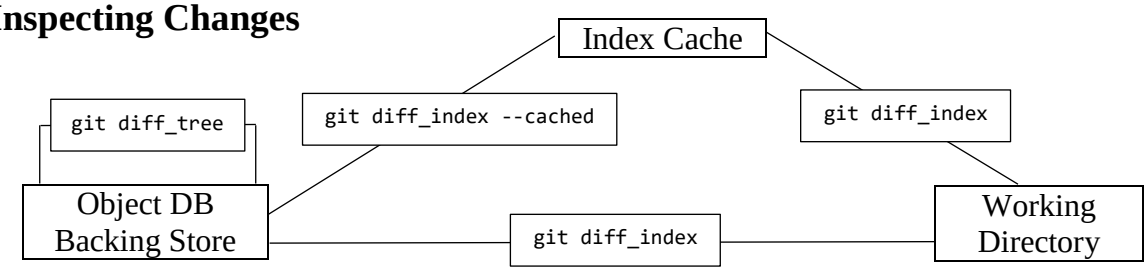
2. Set up Git

```
git config –-global user.name “Your Name”
git config --global user.email you@example.com
```

3. Basic Git Commands for Local Repository

`git init`	Initialize repository (creates .git/)
`git add .`	Stores snapshot in temporary staging area (.git/index)
`git commit –m “msg”`	Permanently stores contents of index in the repository Creates a commit object with “msg” description
`git commit –a`	Combines git add with git commit Automatically identifies any modified (but not new) files and adds them to index and commits in one step

3. Inspecting Changes



`gitk`	Graaphical Representation of Commit History

4. View Commit History

`git log`	History of commits
`git log -p`	Complete diff at each step
`git log –stat --summary`	Overview of changes at each step

5. Tagging a version

Light Tag	Annotated Tag
• Technically nothing more than a branch • But it is placed in `.git/refs/tags/` instead of calling it head • `$ git tag <tagname>` • Simply writes current HEAD to `.git/refs/tags/<tagname>` • Useful for marking certain points using private tag	• Real Git object • Contains pointer to state you want to tag, but also a small tagname and message along with optional PGP signature • Created with `-a` or `-s` flag • `$ git tag –s <tagname> <thing>` • Default – signs current HEAD Optional argument that specifies thing to tag • Useful to tagging major releases

6. Creating Branches

`git branch`	List branches
`git branch <branchname>`	Creates branch
`git switch <branchname>`	Switches branch

7. Merging Branches

`git merge <branchname>`	Merge branch with current branch (HEAD)
If conflicts arise, markers will be left in those problematic files `git diff –` shows the problematic files `gitk –` graphical representation of resulting history Resolve conflicts manually then git commit	

8. Deleting Branches

`git branch –d <branchname>`	Deletes branch but ensures that the changes made in that branch are already in current branch
`git branch –D <branchname>`	Deletes branch but does not save those changes anywhere

9. Basic Commands to work with Remote Repository

Specified Source	
SSH	`remote.machine/path/to/repo.git/` `ssh://remote.machine/path/to/repo.git`
Local	`/path/to/repo.git`
Git	`git://remote.machine/path/to/repo.git`
HTTP(S)	`http://remote.machine/path/to/repo.git`

`git remote add <name> <source>`	Defines remote repository shorthand

`git clone <remote-repo>`	Clones remote repository from specified source
`git pull <remote-repo>`	Fetches changes from a remote branch then merges them to current branch
`git fetch <remote-repo>`	git pull may result in conflicts, so better approach is to git fetch first, then git merge

Comparing for conflicts before merging and after fetching if remote not added	
`git log –p HEAD..FETCH_HEAD`	Show everything that is reachable from the FETCH_HEAD but exclude everything that is reachable from HEAD
`gitk HEAD..FETCH_HEAD`	Same command but visual representation
`gitk HEAD...FETCH_HEAD`	Show everything that is reachable from the either one but exclude everything that is reachable from both

Comparing for conflicts before merging and after fetching if remote added	
`git log –p master..<name>/master`	Same but when remote repository is added using the git add remote command, the fetched changes are stored in a separate remote tracking branch
Then, git merge <name>/master	