Git is a source control management system, or a distributed version control management.

0. Install GIT: **sudo yum install git** ; **git version ; git status ; git help**

1. Initialize a GIT repository: **git init [filename if you don’t have a directory]**

2. Configure Files To Track: **git add [list of files] / git add . (adds all files)**

3. Git Tracks Changes: **git status**

4. Stage Changes: **git add [files] / git add . (add all)**

5. Commiting Changes: **git commit –m „initial commit”** (-m=message)

Before commiting, git needs to know who you are, because all commits are rercorder by the name of the author, so before commiting we have to use:

git config user.name „username”

git config user.email „mail@...”

A commit stores a copy of the file in it’s current state within the .git folder

**Squash add and commit command: git commit –am „message”**

git restore <file> to restore a file to it’s previous state in the Working Area

git restore --staged <file> to restore a file to it’s previous state in the Staging Area

To restore a file from staging area tu working area-untracked: git rm --cached <file>/ git rm –f --<file> (-f=delete)

In order to avoid adding or commiting undesired files, add the filename to .gitignore (example notes.txt file):

echo „notes.txt” >> .gitignore

To see what commits were made to the project: git log / git log --oneline / git log --name-only(prints the file commited) / git log --name-only -3 (last three)

**Branches** = a pointer to a specific commint in git – master by default

Create new branch: git branch <branchname> / git checkout –branchname

Switch branch: git checkout <branch>

Delete branch: git branch –d <branch>

To see a list of branches: git branch (-a)

HEAD = where u are at that moment in the git repository

git log --graph --decorate to see previous commit history along with the branch they were committed on.

Merge: git merge <name of the branch we want to merge into our current branch>

Fast forward merge – when the current branch has no extra commits compared to the branch that we are merging.

No-fast-forward merge – if you commit a change on the current branch that the branch we want to merge does not have, git will perform a no-fast-forward merge – git creates a new merging commit on the active branch

Remote GIT repository (GitHub, GitLab, Bitbucket) – connnection string [https://.../.../[name].git](https://.../.../%5bname%5d.git):

1. Host a GIT server – git daemon (a simple TCP server)

2. Publicly hosted services – Github/Gitlab

Transfer the code from the local repository from my laptop to the remote repository in github.

3. Connecting local repository to remote repository: **git remote add <name(origin)> <url>**

4**. List all remote repositories: git remote –v** (origin= default name given to a remote repo when cloned = original repository)

5. Push the code: **git push –u(connects local to remote branch) <name> <branch> (default branch is master)**

git push origin(alias of the remote repo) master(current branch)

6. Retrieve the code from the remote repository, if you do not have a local copy of the repository you must clone the repo first: **git clone <url> / git clone <ssh link>**

5. Update the local repo from the remote repo**.** After you have merged the origin master branch after review, you have to update the local repository as well:

**git fetch origin master and git merge origin/master**

**but it is recommended to use: git pull origin master (fetch+merge)**

**6. Rebasing: git rebase master (copy comits and hashing is updated – modifies git history)**

**7. Interactive rebasing: git rebase –i HEAD~4(rebase last 4 commits) (squash)**

**8. Cherry pick a commit: git cherry-pick <commit hash>**

**Revert a commit: git revert <commit hash>**

**Reset a commit:**

**git reset --soft HEAD~1 (we keep all the changes that were made on that commit)**

**git reset --hard HEAD ~1 (you loose all the changes that were made to that commit)**

Stash - to keep a version of a file in a working area (not commited): **git stash** and **git stash pop**

**See all the stashes: git stash list**

**See content of a certain stash: git stash show <stash>**

**If you want to pop a specific stash instead of the last one added: git stash pop <stash>**

**Reflog – shows all the actions that have been taken on a repo: git reflog**

**Let’s say you have reseted a commit with hard, you print git reflog, take the deleted commit hash and:**

**git reset --hard <hash>**

**Git is a key value store**

**Porcelain commands: git add/status/commit/stash...**

**Plumbing commands (you can create hash): git hash-object / git ls-files / git rev-parse / git ls-remote ...**

**Quick recap:** [**https://kodekloud.com/topic/conclusion/**](https://kodekloud.com/topic/conclusion/)