Git

Introduction

- Git is a distributed version control system (vcs)
- Free and open source
- https://git-scm.com
- Cheap local branching
- Terminal client and GUI clients
- IDE integration

Git configuration

git --version – prints git version

git config – git configuration (you can find all properties here https://git-scm.com/docs/git-config)

Starting repository

- Any directory on a computer can be become a Git repository
- All git related data is stored in hidden .get subdirectory
- From local: 'git init' command initialize current directory as a git repository
- From remote repository: 'git clone <URL>' pulls remote repository to the current directory
- git status – prints current repository status: branch, changes and untracked files

Making changes

- All changes to a repository must be registered in staging

'git add <file>' command registers changes

- To save changes call 'git commit' command and then provide message that describes the commit
- 'git diff' and 'git diff –staged' shows changes

History and comparison

git log – prints commits history

Commit hash – identifier of a commit that you can use as a reference

HEAD – is the latest commit

git diff HEAD~1 – compares the current state with the state before the latest commit

Use HEAD~n to refer to a previous state relative to the latest commit

git diff <commit_hash> – compares the latest commit with the particular commit

Restoring state

git restore <file> – discards changes to the last known state

git restore --source <commit_hash|relative_position> <file> - restores the state of a file to a specific commit. You must commit these changes if you want to save them.

git checkout <commit_hash| relative_position> <file> - does the same as 'git restore'

git checkout <commit_hash| relative_position> - switches whole directory to the state, you can restore by calling 'git switch main' or 'git checkout main'

Ignoring files and directories

- '.gitignore' file in a repo directory
- Contains patterns that are used to exclude files and directories from git
- Exact file or directory name
- Mask
- By conversion you should specify '.gitignore' file at the top-level directory but you can add '.gitignore' to any directory in your repository
- Rules inside '.gitignore' file are relative to the directory in which the file is located

Remote repository

- 'git clone <URL>' pulls and sets up a remote repository to the current directory
- By convention remote repo is referred as 'origin' which is a shortcut for the repository URL
- git remote add <name> <URL> - adds remote repo with the given name and URL
- git remote –v – prints all remote repos
- Local --push--> Remote
- Local <--pull—Remote
- git push origin main – pushes 'main' branch to the remote repo

Remote repository

- You must explicitly sync local and remote versions of a repo by calling 'git push' and 'git pull' commands
- If remote repo has changes that you don't have in your local repo, then you must pull these changes before pushing your local content to the remote repo
- If both local and remote repos contain changes of the same location, then you must resolve this conflict manually
- If it's possible git performs auto merging

Working with branches

- git branch <branch_name> - creates new branch
- 'git checkout -b <branch_name>' and 'git switch –c <branch_name>' - creates and switches to new branch
- 'git switch <branch_name>' or 'git checkout <branch_name>' - switches current branch
- git branch -a – lists all branches
- git branch -d <branch_name> - deletes the branch
- git branch -D <branch_name> - forces delete of the branch

Merging branches

- git switch <dest_branch_name> - switch to destination branch and then call 'git merge <src_branch_name>' to merge the source branch to the destination one.
- On the source branch call 'git rebase <dest_branch_name>'
- git rebase – incorporates changes from a branch to the current branch by rewriting the history of a commit. All commits from <dest_branch_name> are prepended to commits on the current directory
- It's recommended to do merging or rebasing on branch copies, so you always can just delete a branch
- Use rebase when you are still working, and you need to incorporate the latest changes from the main branch

Stash changes

- Sometimes you need to switch to another branch before you have finished your work
- You cannot switch to another branch without committing your current changes
- git stash – saves not finished work as a temporary commit
- git stash list – prints all stashes for the current branch
- git stash apply – restores the last temporary commit
- git stash clear – clean up your stashes