**HomeTask: [SCM Git] Basics**

**The contexts of the task is refer to GitLab (documentations, guides, etc). But you can use GiHub as well (the only thing that changes isweb interface).**

1. Install GIT on your local machine in case it is not installed yet.
2. Config git: change your global configs (*add name and email, setup core text editor*).
3. Provide access to GitLab, i.e., generate **ssh-key** and integrate it with Gitlab. (Use <https://docs.gitlab.com/ee/gitlab-basics/create-your-ssh-keys.html> fot the help).
4. **Create new "gitlab\_practise" project on GitLab.**
5. **Clone project to your local machine.**
6. Provide access for Anna (@annamuravska) to your repository on Gitlab (**as Developer**).
7. Create ".gitignore" file. Include ".gitignore" in its own list. Try to use different patterns in .gitignore file to make invisible for git some local files that should not be commited (see git official documentation). Check that files mentioned in .gitignore file shouldn’t be visible via executing git status command. (Make some experiments).
8. Open terminal or git console in root directory of your cloned project **"gitlab\_practise"** and do next.
9. Do all actions in folder **“git\_task”.**
10. Create empty README.txt file.
11. Make init commit to the repo **"gitlab\_practise".**
12. Create **develop** branch and **checkout on it.**
13. Create **common.html** empty file. Commit.
14. Create branch “**books**”. Checkout on it. Add some **books folder** with whatever .pdf (or so) files inside it. Commit.
15. Change **common.html**. Add some book source inside it ()or some text inside common.html). Commit.
16. Return to **develop** branch.
17. Create **branch** “**docs**”. Add some documentation folder with documents inside it (e.g., some .md, .txt files). Commit.
18. Change **common.html** (as in 15 step). Commit.
19. Return to **develop** branch.
20. **Merge two created branches into develop** with **git merge** command. Try to resolve conflict in case it appears. Follow the next steps:

* merge “books” into “develop”
* merge “docs” into “develop”

NOTE: Do not delete any branches!

1. **Merge develop into master.**
2. Checkout to **develop** branch and repeat **14-21** items one more time (use names “**new\_books**”, “**new\_docs**” for new branches). This time use **git rebase** command instead of git merge in step 20. Follow rebase according next steps:
3. rebase “new\_books” onto “develop”
4. rebase (fast-forward) “develop” onto “new\_books”
5. rebase “new\_docs” onto ‘’develop’
6. rebase (fast-forward) “develop” onto “new\_docs”
7. **Merge develop into master.**

NOTE: As a result master/develop branches should contain all commits that were done in process. (You can check it via **git log** or via UI tools - <https://tortoisegit.org/about/>).

1. Push all these changes with all existing branches to origin (**git push origin all**).
2. Do "**git reflog**" and save it content on local machine (not in repository) with filename “SCM\_GIT\_Basics\_HT.txt”.

**Advanced GIT (for those who want to play with interactive rebase).**

1. Checkout on ”develop”.
2. Do all actions in folder “**git\_task\_pro**”.
3. Create any file in new folder “**git\_task\_pro**” and commit it.
4. Create new branch “**feature\_branch1**”.
5. Create another new branch “**feature\_brach2**”.
6. Do ~3 commits to “develop” (e.g., add to repository some files, change them).
7. Checkout on the “feature\_branch1” branch and do ~5 commits.
8. Checkout on the “feature\_branch2” branch and do ~3 commits. Make sure that you have at least one commit, which contains changes in **more than one file**.
9. Perform **interactive rebasing**. I.e., your commits from the “feature\_branch2” branch should appear on the top of “develop” branch. During interactive rebasing:

* Divide one of commits (so you should have two commits instead of one);
* Change content of other commit and change commit message for it.

Use <https://git-scm.com/book/en/v2/Git-Tools-Rewriting-History> for help.

1. Merge “feature\_branch2” branch into develop (fast-forward). “Develop” and “feature\_branch2” branches should point to the same commit.
2. Checkout to the “feature\_branch1” branch.
3. Do interactive rebasing. I.e., your commits from the “feature\_branch1” branch should appear on the top of “develop” branch. During interactive rebasing:

* Squash three commits into one;
* Drop other commit.

1. Merge “feature\_branch1” branch into develop (fast-forward). Both branches should point to the same commit.
2. Merge the “develop” branch into “master” branch.
3. Push all your local branches to origin.
4. Do "**git reflog**" and save it content on local machine (not in repository) with filename “SCM\_GIT\_Pro\_HT.txt”.