Git Advanced

Estimated time: 2 days

2019

# Theory

1. What is Git?
2. What is the difference between Git and Svn?
3. Learn the most useful Git commands:

<https://davidwalsh.name/git-commands>

<https://orga.cat/posts/most-useful-git-commands>

1. *Optional*. Learn how to use **Git** with [Code School's interactive course](https://try.github.io/levels/1/challenges/1)

# Terms and Conditions of ESM program

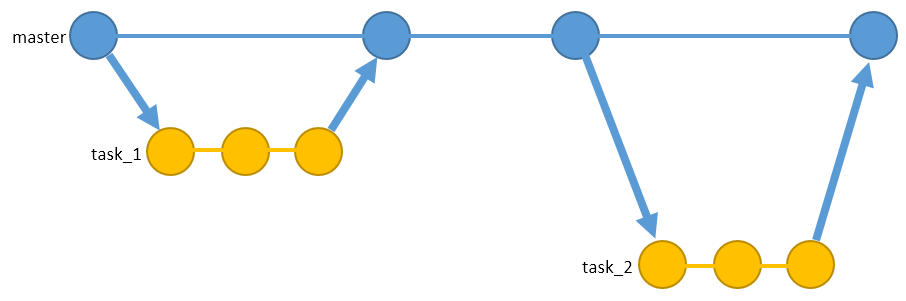
Use git bash for this task **only**.

Install TortoiseGit or any other well-known GUI for following tasks.

**Every day you should commit your changes to git repository (develop branch) for tracking your dev progress.**

## Branching strategy

The whole development **must** take place at git.epam.com. Create a repository and give Master access to all mentors (including consultants) and a coordinator.



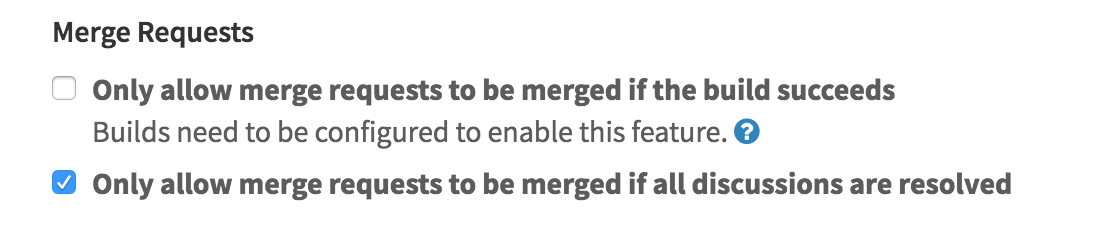
All merge requests should illustrate progress of a current task.

## Pull requests

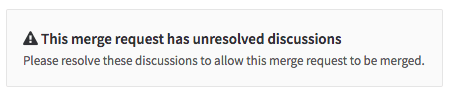
When you finish a task, you should create a pull request (for Gitlab – merge request) from your **current branch** to **master**.

When you reply to a mentor’s comment in GitLab, mention a mentor with **@MentorName** tag, except comments such as “Done” / ”OK” – in this case a discussion should be simply marked as Resolved.

Turn on the following setting in your repository:

****

**And do not leave unresolved discussions before merging.**

****

## SonarQube

**All** pull requests (merge requests) **must** contain attached SonarQube statistics with passed quality goals, starting from Task 1.

For more details: <https://www.sonarsource.com/products/codeanalyzers/sonarjava.html>

# Task 0

For this task, you should create two separate branches **git\_0** and **git\_1** inyour remote repository and local tracked branches with the same names.

Show the following steps to your mentor during a demo:

1. **git\_0**: Add and commit 1.txt file with 10 lines.
2. **git\_0**: Add and commit 2.txt file with 10 lines.
3. Merge branch **git\_0** to **git\_1**
4. **git\_1:** Update and commit any **two** lines in 2.txt.
5. **git\_0:** Updateand commit **the same** 2 lines with the different info in 2.txt
6. Merge branch **git\_1** to **git\_0,** resolve conflict. Left **all** (4) modified lines. Commit.
7. **git\_0**: Update and commit 1.txt file, modify two lines.
8. **git\_0**: Update and commit 1.txt file, modify **another** two lines.
9. Transfer changes of commit from Step 7 only to **git\_1,** using format patch.
10. Transfer changes of commit from Step 8 only to **git\_1,** using cherrypick command.
11. **git\_1:** Concatenate the last two commits using *reset + commit* commands.
12. **git\_1:** Change date, author and message of the last commit and add non-empty 3.txt file to it.
13. **git\_1**: Create a **new** commit that *reverts* changes of the last one.
14. **git\_1:** Updateand commit 3.txt file.
15. **git\_1**: Run command that removes all changes of the last two commits.
16. Synchronize **git\_0** and **git\_1** with a remote repository.
17. Clone your project to another folder.
18. **folder2: git\_0:** Change two lines in 1.txt. Commit + Push.
19. **folder1: git\_0:** Change **another** two lines in 1.txt.
20. **folder1: git\_0:** 
    1. Change **another** line in 1.txt (not the same as in 18, 19).
    2. Merge changes from Step 18 (pull) **without** committing changes from Step 19 and any additional commits.
    3. Push.
    4. Return local state of Step 19. (*stash*)