# Installing sourcetree

You can download sourceTree for Windows on <https://www.sourcetreeapp.com/>. They develop software both for windows and OS system. Installation is straightforward.

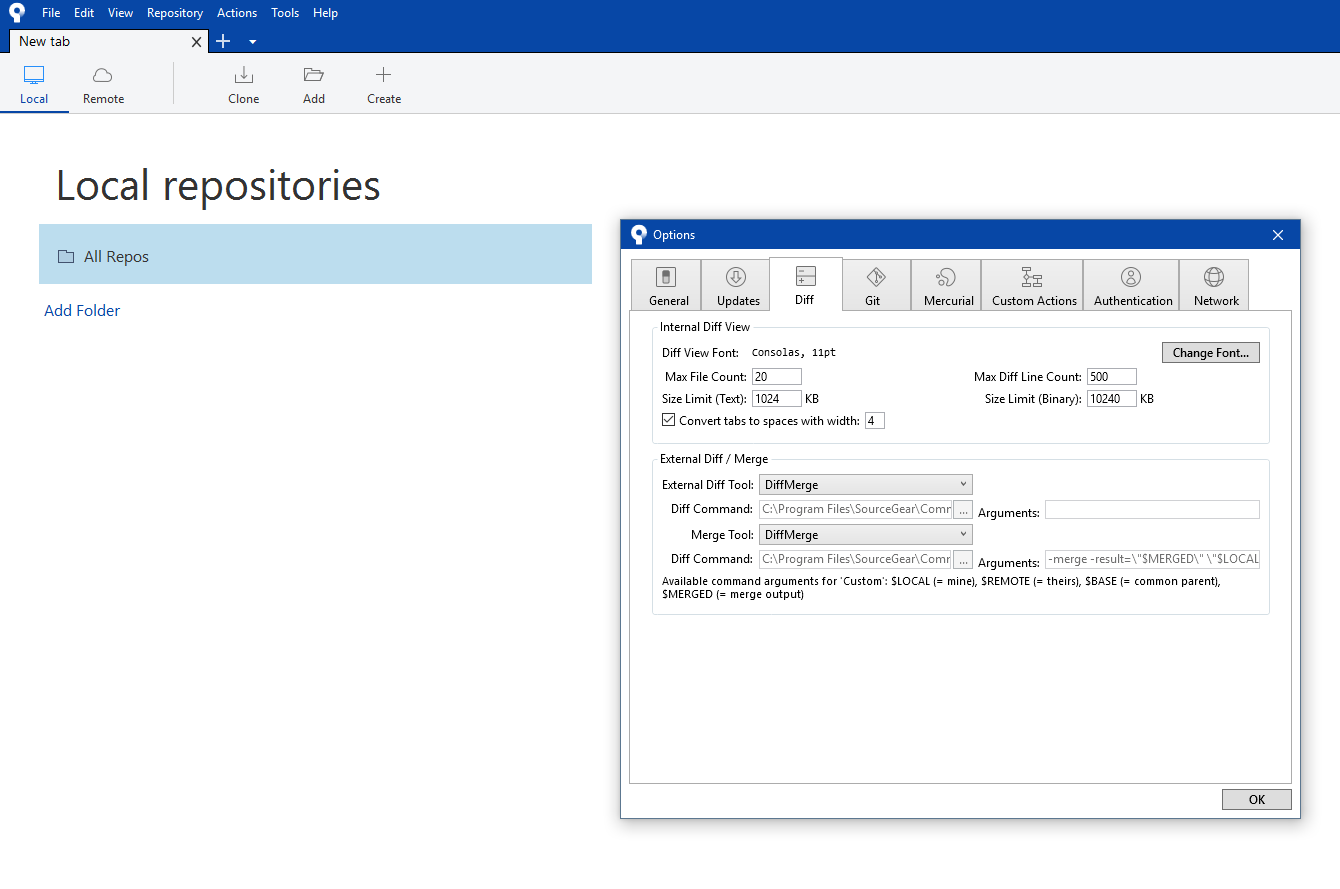
# Download diffMerge to manage conflicts

You can download diffMerge from following link: <https://sourcegear.com/diffmerge/downloads.php>

Installations are straightforward.

# Setup diffmerge tool for manage conflicts

To manage conflicts (we cover that in the further chapters), you need to set up repository on the GitHub. You can do this by following steps from picture bellow:

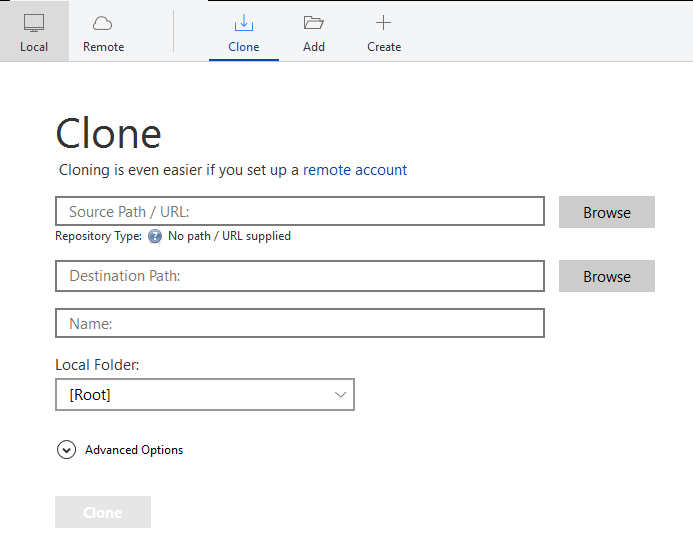


* Choose tools -> options from menu
* In options window go to Diff card
* Set ExternalDiff Tool and Merge tool to DiffMerge plugin

# Cloning project from GitHub

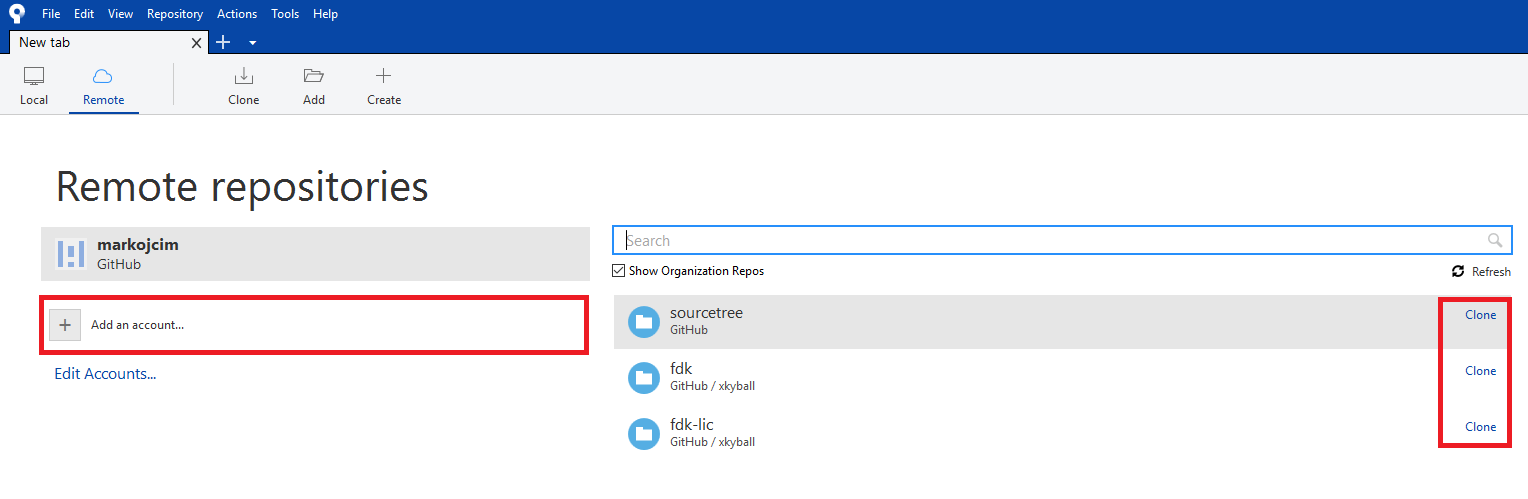
We probably want to work on some real project. We can create new project, or clone existing one from GitHub. In our case, we will clone projects from GitHub.

We have 2 options to do that. First is to manually clone project from GitHub. The other is to do it using sourceTree:



* Click on Clone button on sourceTree main menu
* Set source path, destination path and name
* Click on clone. It will create folder on the destination path and create new tab in sourceTree with existing branch and commits in that repository

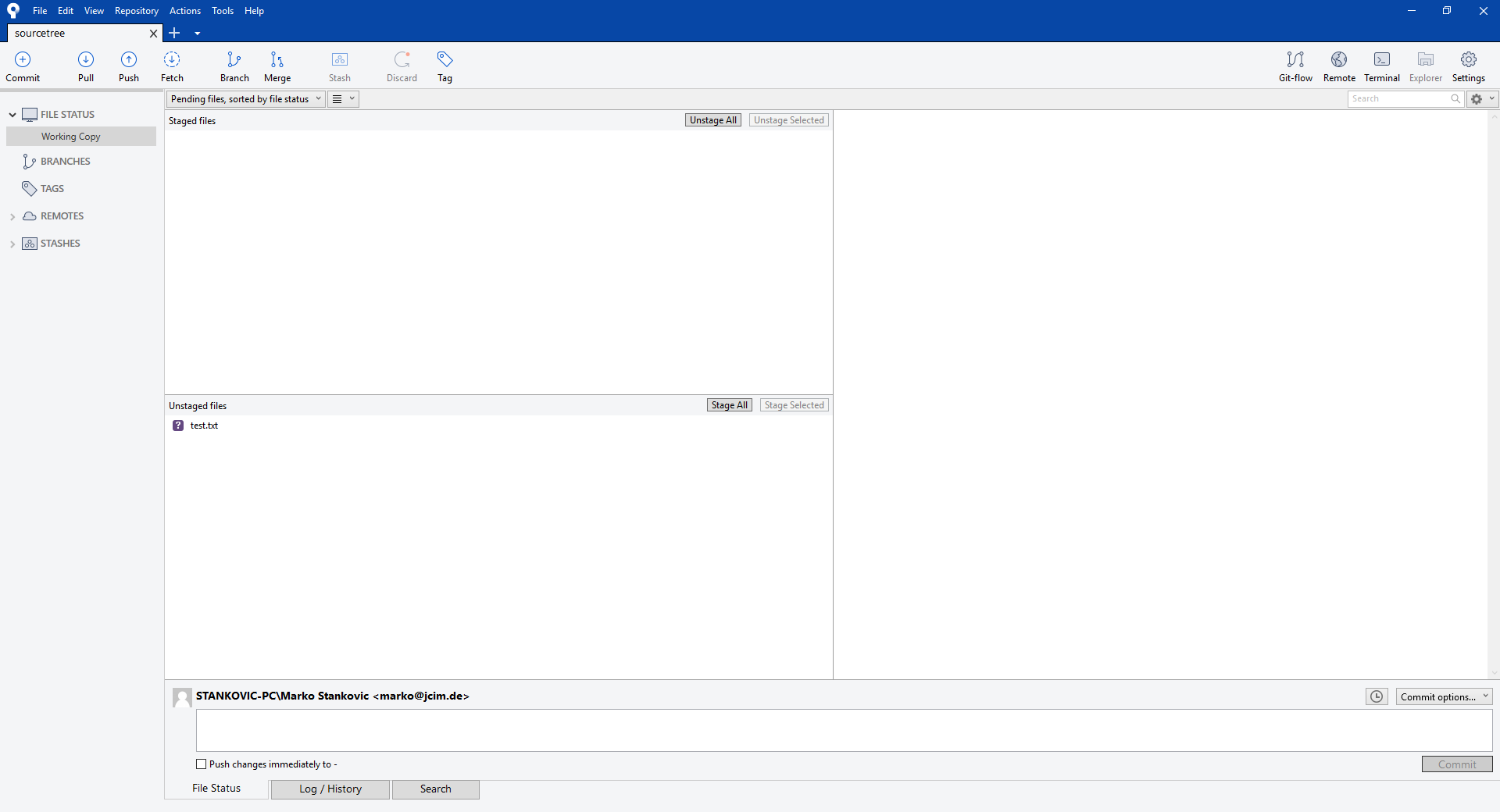
We can also setup a remote account, and after verifying we can clone all projects that belong to our account in one click.



* Add an account to sourceTree
* On the right side appear all projects belongs to user
* Clone whatever you want

# Initial commit

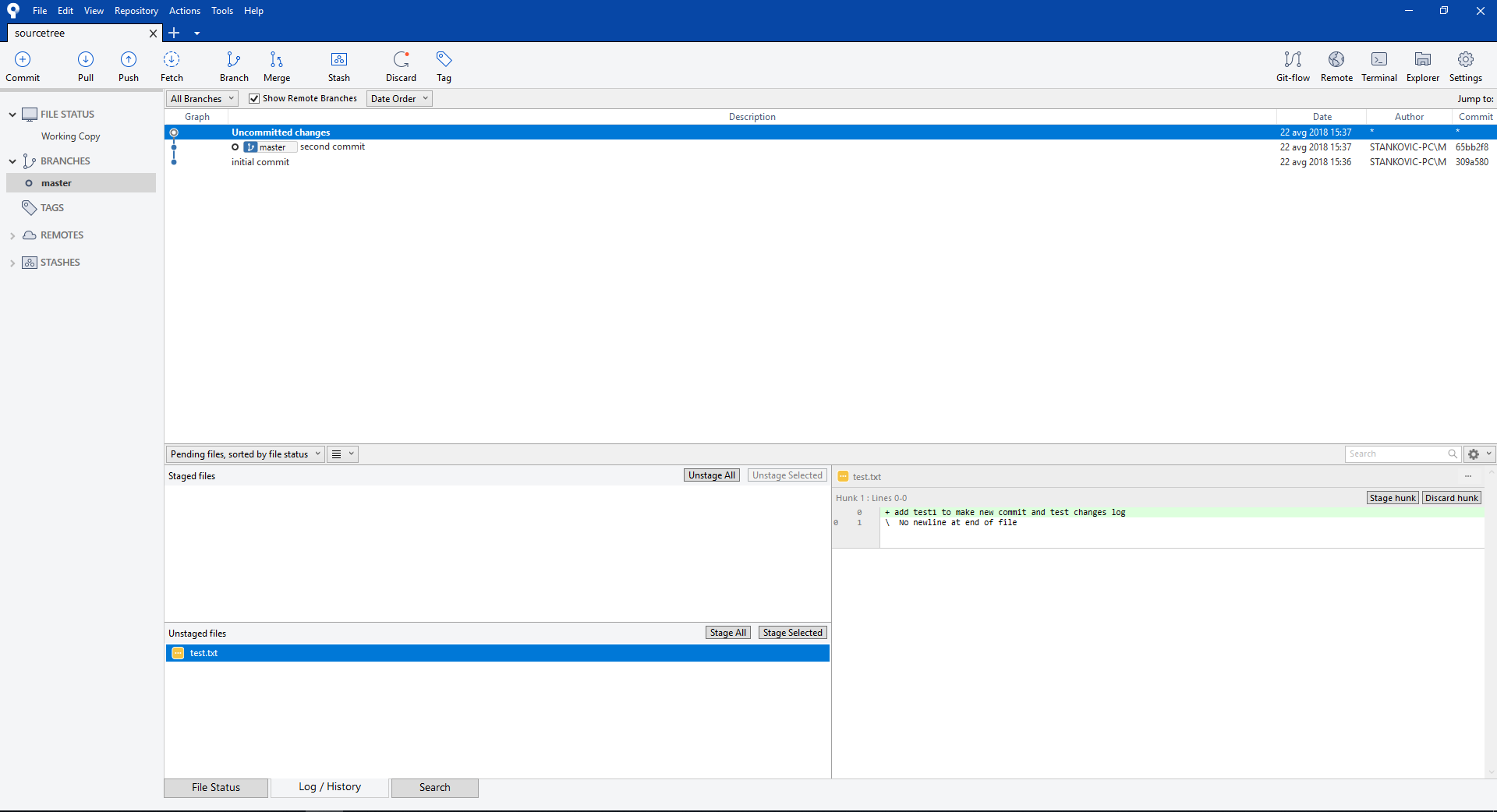
At the beginning, we create an empty repository. When we create some document in repository folder, it will appear in sourceTree (refer to the picture below):



* Bottom left corner contains all files that differ from the last commit. When you want to commit it to GitHub you need to stage files (all or part of it), add description to the commit in the textbox on the lower part of the screen. Then, press commit and that’s it.

# Commit after changes, stage files, check status

We need to commit the source code whenever some changes happened to it. In the following picture we can find some useful information:

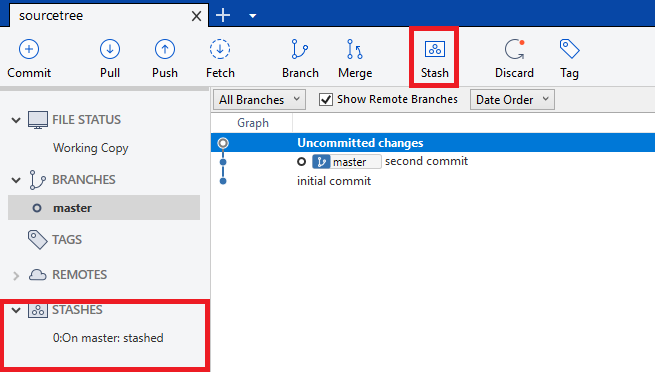
****

* We have unstaged files list, it contains all files, which are changed, since the last commit
* When we click on the one of unstaged files, at the right side window we can see all changes in that file since the last commit
* We can stage and commit files, like in initial commit

# Stash command

When you have uncommitted changes, which are not good enough for an official commit, and you want to change branch and do something else, stash command is suitable for you.

With stash command we can clean the branch to be equal to the last committed state. Stashed changes are saved on stack, and you can revert them when you want.



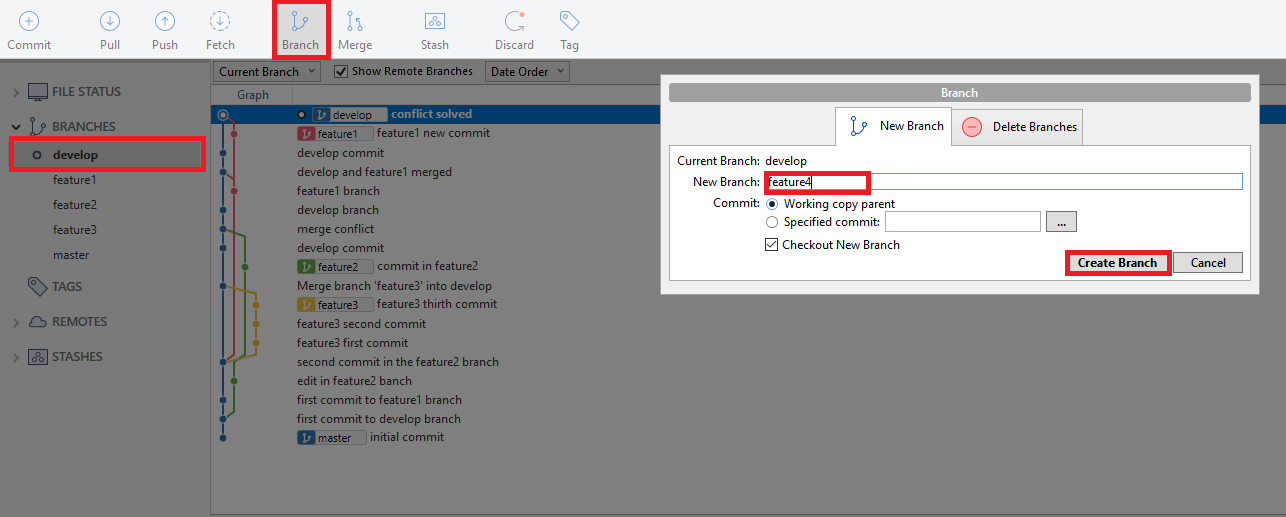
* Stash code with stash button
* Find all stashes in the stashes menu

# Put code in the cloud, push/pull changes

All commits are changes to the local repository. If we need to put that on the GitHub repository on the cloud, we need to push these branches. When branch is on GitHub, every following commit need to be pushed. If branch is in the cloud, it has ‘origin’ prefix.

# Branching

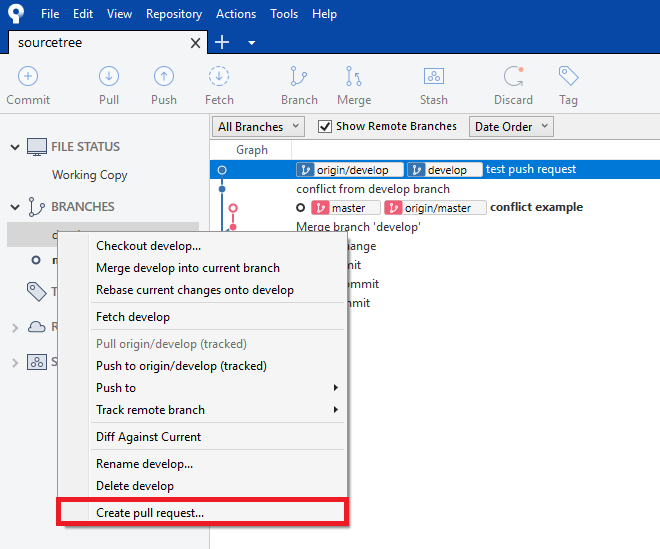
To create a new branch (suitable for develop, feature, hotfix… branches), follow the steps indicated on the image below:



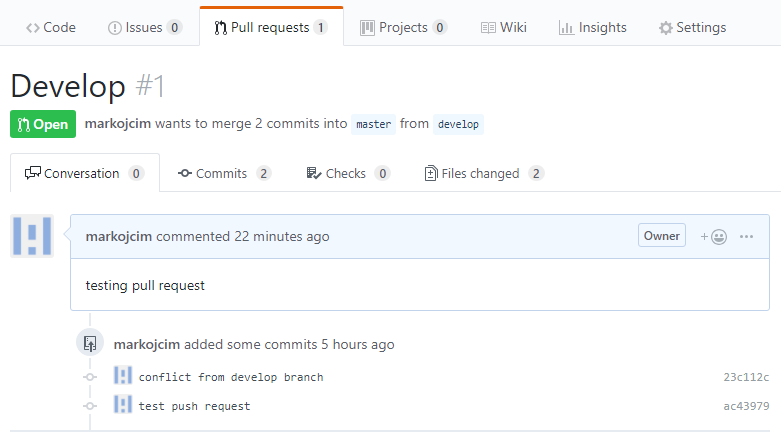
* Navigate to the root branch (develop in this showcase)
* Click on the branch button, to open the branch window
* In the branch window, add the branch name and click on the “create branch” button

# Pull request

To create pull request, follow next steps:



Right click on merged branch and chose create pull request. It will be redirect to the following GitHub page:

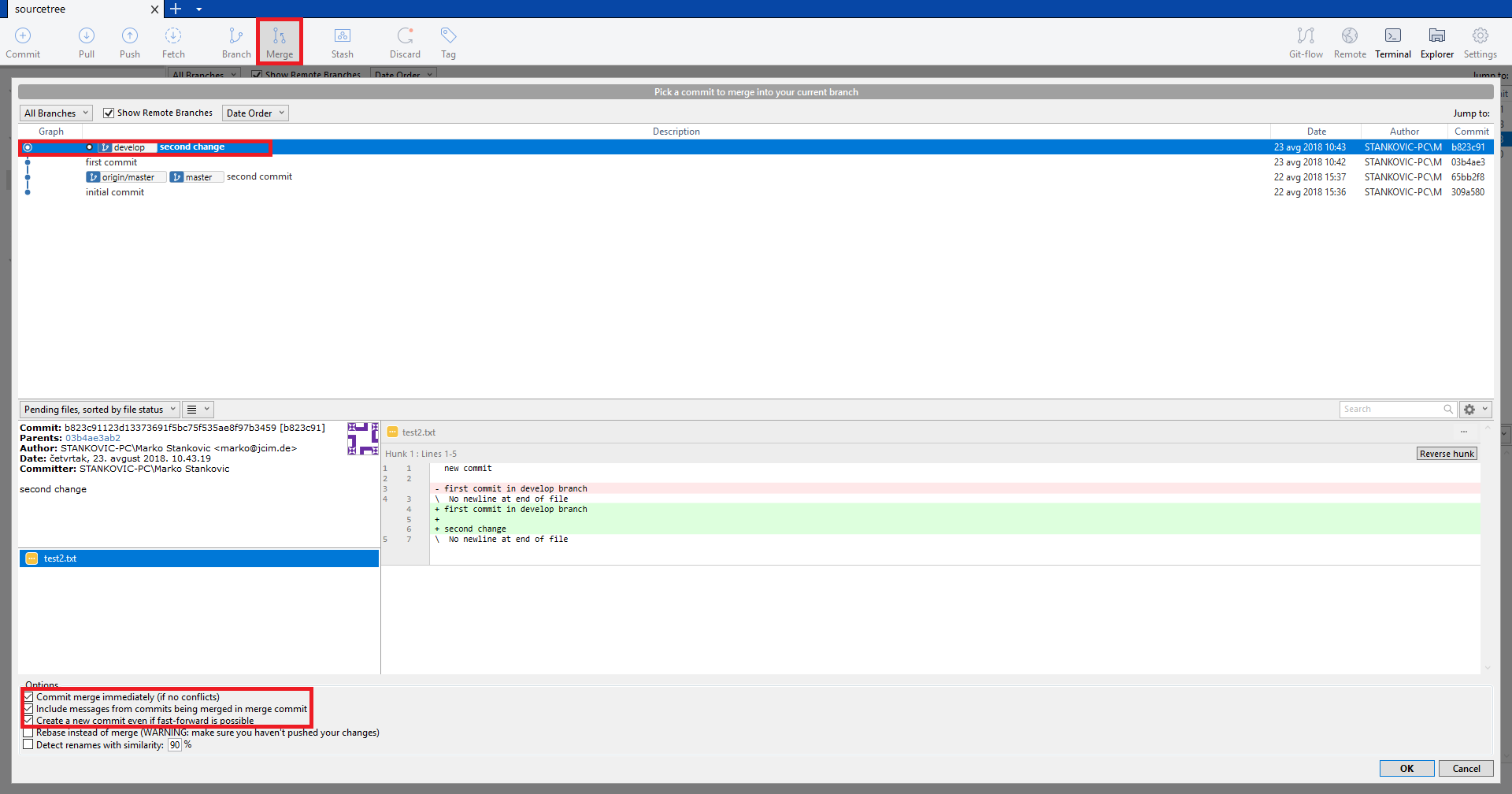


With pull request creating, you can specify name and description of pull request and add specific reviewers.

\*\*\*It needs to be adjust with pull request at reviewer side, when we have real case\*\*\*

# Merging

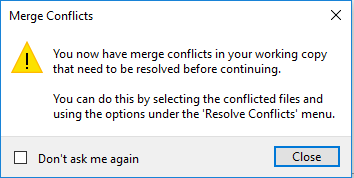
To merge 2 branches, follow the instructions indicated on the picture below:



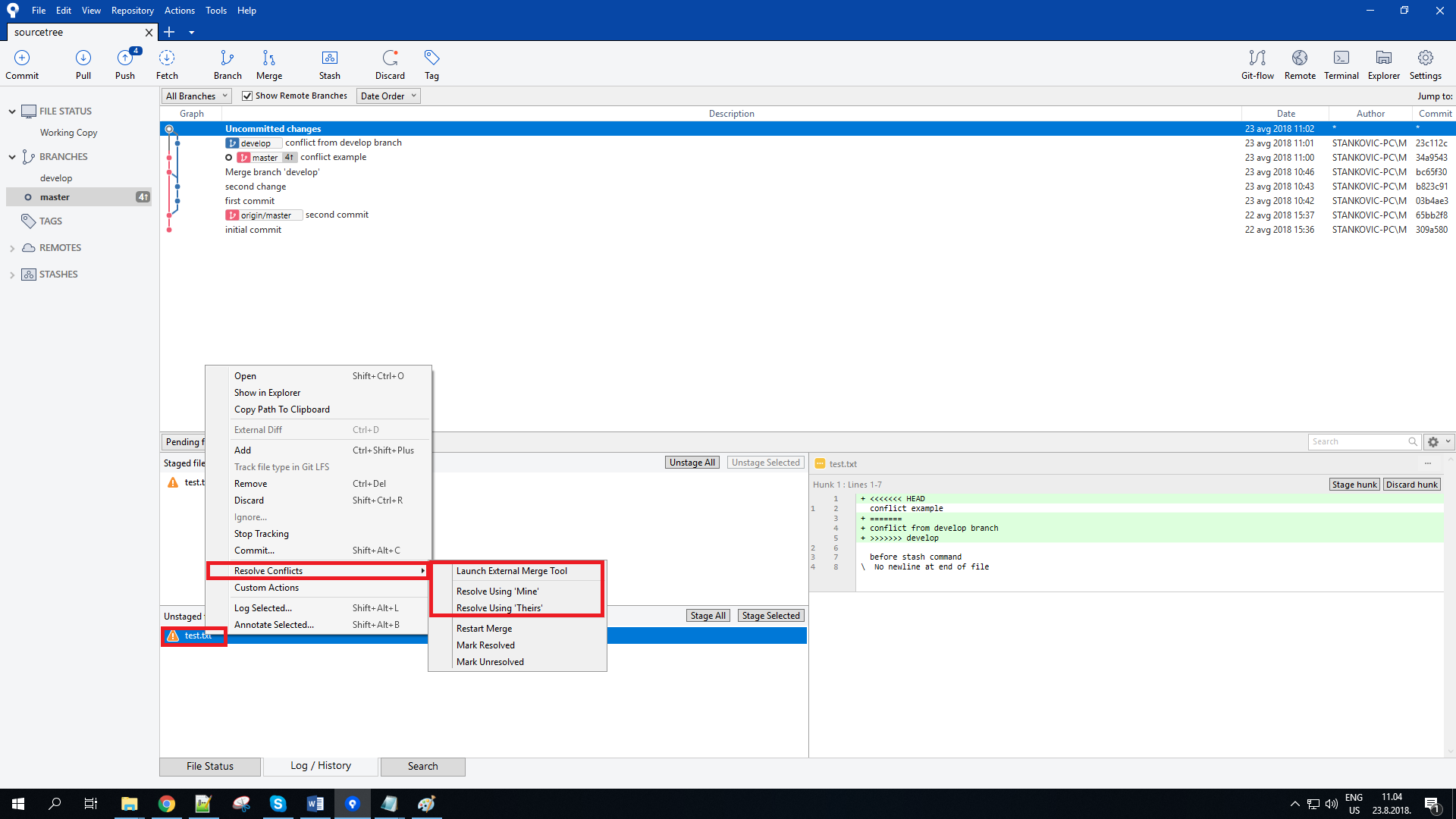
* Navigate to the branch to which we want to merge (master in our case)
* Click on the merge button
* Choose the branch that you want to merge with the current branch (develop to master)
* Tick “Include messages from commits being merged in merge commit” and “Create a new commit even if fast-forward is possible”. With these commands, we won’t lose any information relevant to our GIT repository.

# Resolve conflicts

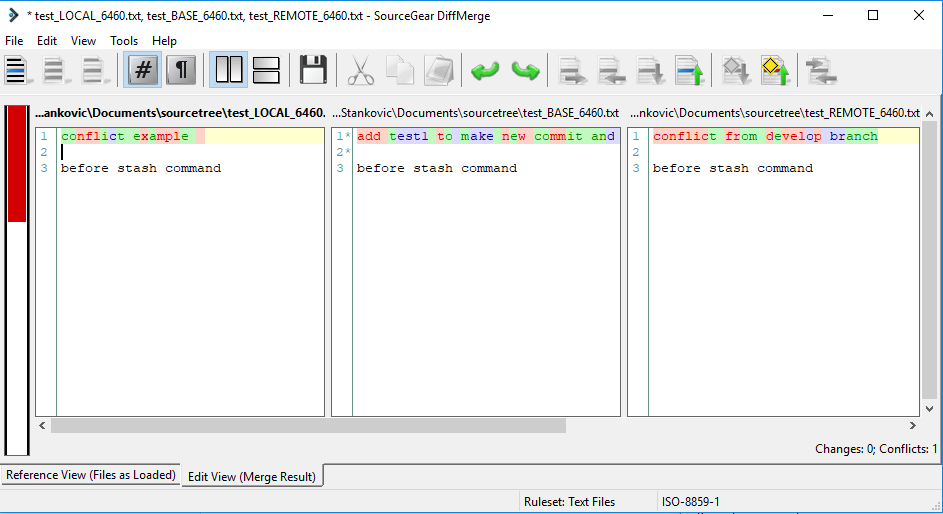
Merge conflicts can happen during the merge process. SourceTree will show the following message:



When that happens, it is important to handle the case properly.



After notification that conflicts have happened, sourceTree shows the window indicated above:

* At unstaged files, we have exclamation mark next to file that contains conflicts
* When you right-click on that file -> resolve conflicts, SourceTree offers 3 possibilities to resolve it
* We can resolve using mine or resolve using theirs, where we completely ignore changes from one side, and take those from another.
* We also can launch external merge tool, and diffMerge tool will start
* With diffMerge, we can resolve conflicts in a custom manner. It shows a version from the root branch, a version from the merged branch and a new unified version. We can resolve the conflict by creating a new unified version, following project requirements.