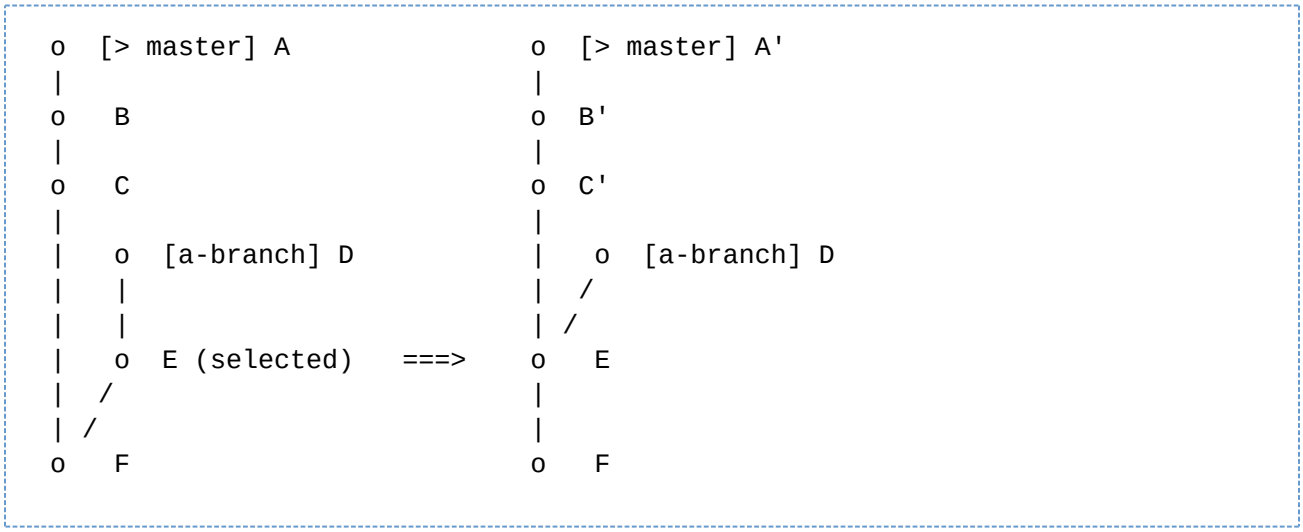
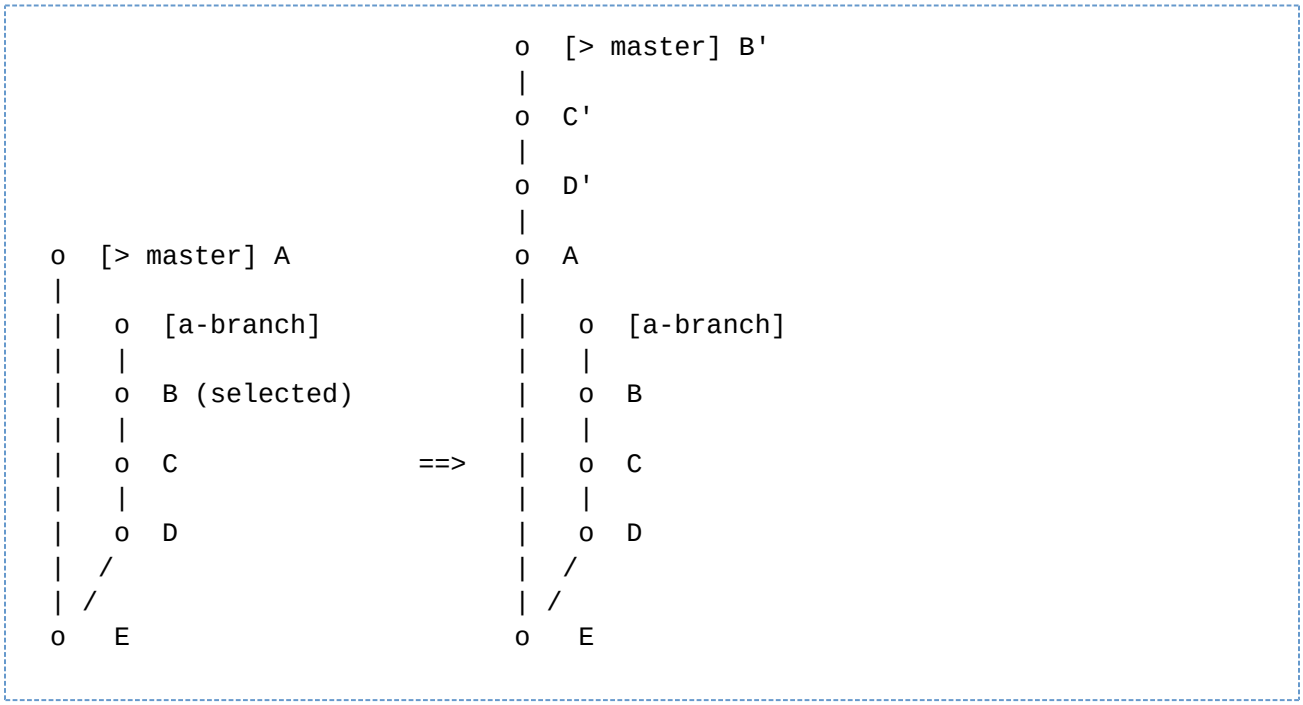
Rebase

The Rebase command allows you to apply commits from one branch to another. Rebase can be viewed as more powerful version of Cherry-Pick, which is optimized to apply multiple commits from one branch to another. In SmartGit, a distinction is made between **Rebase HEAD to** and **Rebase to HEAD**:

Rebase HEAD to rebases ("moves") the commits below the HEAD to the selected commit. The HEAD will be moved to the new fork.



Rebase to HEAD duplicates commits from a separate branch to the HEAD (similar to what Cherry-Pick does). The HEAD moves forward on its fork.



To **Rebase Onto** you may use the **Log** window. Consider following example where the `quickfix2` branch should not start at the `quickfix1` branch, but rather on the `master` branch:

```
q2b (quickfix2)
 |
q2a
 |
q1b (quickfix1)
 |
q1a
 |
 x (master)
 |
...
```

To achieve this, just drag the `q2a` commit onto the `x (master)` commit and you will get the desired result:

```
q2b (quickfix2)
 |
q2a
 |
 |  q1b (quickfix1)
 |   |
 |  q1b
 | /
 x (master)
 |
...
```

In SmartGit, there are several places from which you can initiate a rebase:

- **Menu and toolbar** On the main window, select **Branch|Rebase HEAD to** or **Branch|Rebase to HEAD** to open the **Rebase** dialog, where you can select the branch to rebase the HEAD onto, or the branch to rebase onto the HEAD, respectively. Depending on your toolbar settings, you can also open this dialog via the buttons **Rebase HEAD to** and **Rebase to HEAD** on the toolbar.
- **Branches view** In the **Branches** view, you can right-click on a branch and select **Rebase HEAD to** to rebase your current HEAD onto the selected branch.
- **Log Graph** On the Log graph of the **Log** window, you can perform a rebase by right-clicking on a commit and selecting **Rebase HEAD to** or **Rebase to HEAD** from the context-menu.
- **Log Graph** In the Log graph of the **Log** window, you can drag and drop commits or refs and then select to rebase in the occurring dialog after the drop.

Just like a merge, a rebase may fail due to merge conflicts. If that happens, SmartGit will leave the working tree in *rebasing* state, allowing you to either manually resolve the conflicts or to **Abort** the rebase. See Resolving Conflicts for further information.

Interactive Rebase

You can start an interactive Rebase in the Journal view by right-clicking the first commit that should be changed. For further information, please see Journal View.

Resolving Conflicts

Core Git rebase conflicts are different to other kinds of merge conflicts, because *left* and *right* files are swapped: when rebasing branch A to B, Git will first checkout B, then applies all commits from A. If a conflict occurs, HEAD still points to B and hence the *left* file would be the file as it's present in B.

From a user's perspective, the *left* file should always be his/her own file ("ours"), i.e. the file as it's present in A. For this reason, in case of rebase conflicts, SmartGit will swap *left* and *right* files. This gives a more consistent user experience, however may result in following different behavior (compared to normal merge conflicts):

- When staging left lines (**Ours**) in the **Conflict Solver**, these lines will finally show up as staged, because your rebase branch `B` is actually "theirs"
- When invoking **Resolve** and selecting **Ours**, you will see staged file content, because your rebase branch `B` is actually "theirs"