**Minimum GIT Setup in RAD:**

Ensure you have followed the Developer Setup instructions on the Wiki. There are five (5) documents that you need to follow.

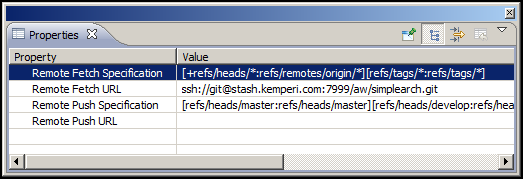
**Assumptions:**

You have a RAD Workspace with:

* Where the Git repo has been configured with Stash(Fetch, Merge & Pull).
* The RAD projects are building cleanly and correctly.

**Verify your Remote Fetch Specification (Optional):**

* In the Git Repository view., expand Remotes
* Right-mouse on “origin”, Show In > Properties
* Ensure the Remote Fetch Specification matches below:



**Before accepting remote changes:**

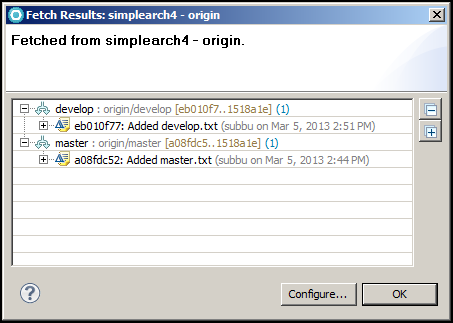
* Make sure your working directory is as clean as possible.
* Try to commit all changes. Uncommitted changes cannot be automatically merged by Git.

VERY IMPORTANT!!!

If you do a Fetch & Merge or Pull and the EGit plugin just returns without a dialog and you get nothing there is a conflicting file in your working directory that must be committed before attempting the command again. This is a quirk with the EGit plugin.

**Fetching the latest code from Origin:**

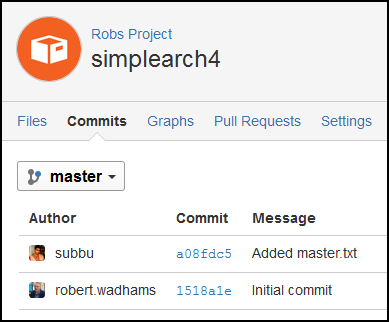
* In the Git Repository view
* Right-mouse on “origin”, Fetch

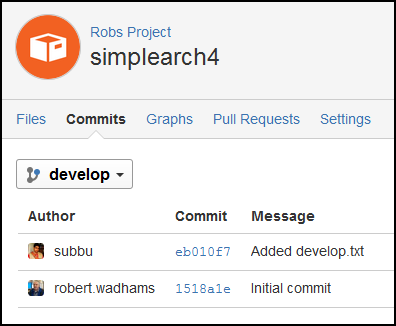


This dialog is showing:

* commit “eb010f77” on the develop branch is being fetched.
* commit “a08fdc52” on the master branch is being fetched.

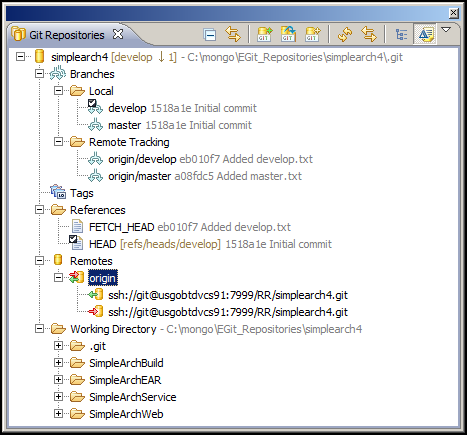
The Stash commit views below verifies the commit SHAs





Notice three (3) things:

* Remote Tracking (origin/develop and origin/master) reflect the latest commit SHA for each branch.
* The Local (develop and master) branches still have the prior commit SHA (1518a1e). This also means your workspace code has not changed.
* The repo (simplearch4) is now annotated with [develop **↓** 1]. This is telling you that your code base in one (1) commit behind the remote branch.



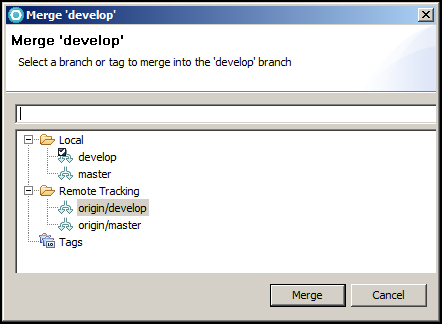
**Important!!!** Fetch does not update your code base in your local branches. You will be required to merge the remote branches into your local branches.

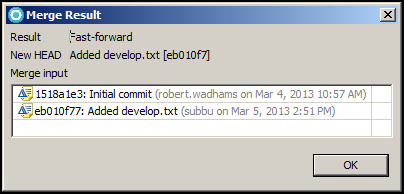
**Merge the fetched code into the local branches:**

Merging generally requires you to be in the branch that you wish to merge changes into. The example below starts with the “develop” branch.

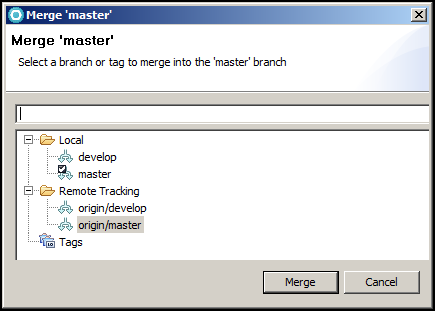
**Make sure you are on the correct branch before merging!**

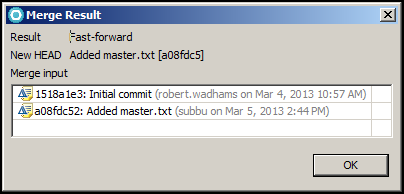
* In the Git Repository view
* Right-mouse on the repo, Merge… (Always use Merge…)
* Select the Remote Tracking branch where the new changes are (in this case, origin/develop).
* Merge.
* Verify the Merge Result dialog.





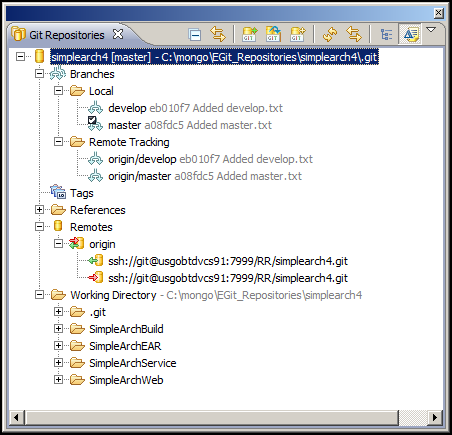
* In the Git Repository view
* Right-mouse on the repo, Switch To > master
* In the Git Repository view
* Right-mouse on the repo, Merge… (Always use Merge…)
* Select the Remote Tracking branch where the new changes are (in this case, origin/master).
* Merge.
* Verify the Merge Result dialog.





The Git Repository view below shows that the two local branches are at the same commit as the Remote Tracking branches.

You are now completely caught up and ready to continue work. Make sure you change to the correct branch before changing and committing code.



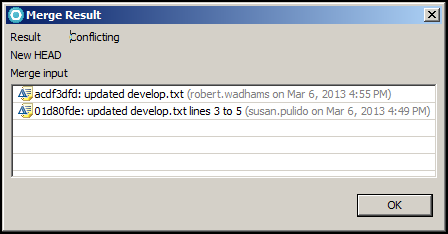
**What happens to local changes when merging?**

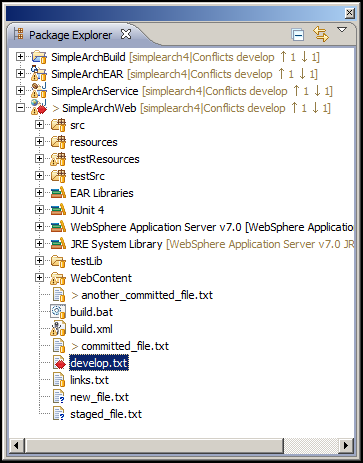
* New files (untracked files) remain in your working directory.
* Staged files are reverted back to “unstaged”.
* Commits are merged as expected.

**What happens with merge conflicts?**

Merge conflicts occur on those rare occasions where two developers change the same file / same lines. In this case Git flags the conflict and annotates the file showing the conflicting lines. **Note:** Two developers can change different parts of the same file and Git will happily merge both changes together.

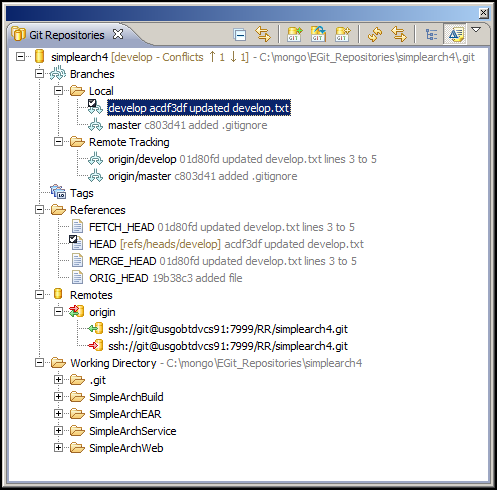
When a merge conflict occurs, you’ll see the following dialog boxes and views. Resolving the conflict is also explained.



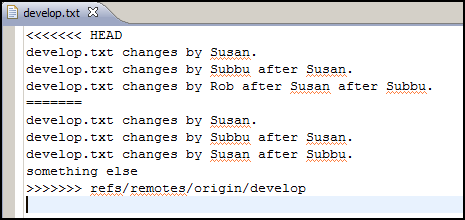


**Note:**

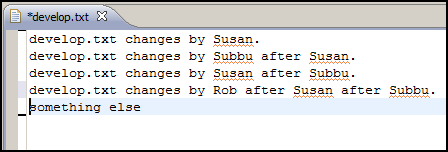
* The red double-diamond conflict marker.
* The label decoration after each project (simplearch4|Conflicts…)



Open the conflicting file in your editor

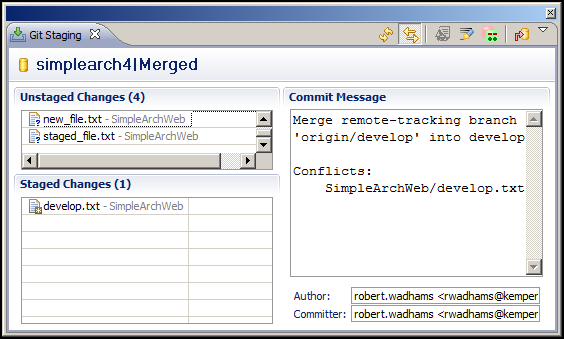


Remove the conflict markers and make the file look the way you want it.



Save the file.

In the Git Staging view, drag the resolved file into “Staged Changes”. An automatic commit message will be provided. Add more commit details if you wish.



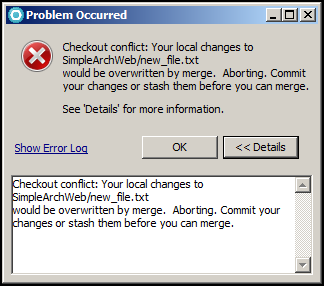
Commit.

**What happens when a new file (untracked) slashes with remote changes containing the exact same new file?**

This situation would occur if both these conditions happened:

* You created a new file in your working directory and didn’t commit.
* The remote fetch contained a commit that also contained the new file.

A merge under these circumstances would result in a rejected merge:



Solution:

Delete or rename your local file. Then attempt the merge again.