Why Gerrit does not allow in-file merges and why rebase has to be done multiple times?

By ABAIT Team

Reasons for rebasing multiple times

- Parallel verifications done for all changes approved at once in ABAIT CI.
- So when two/more patches make changes to same file the first one to download remains in the tree and others are removed from the batch.
- Technically its like a race to the mainline and the first change in the batch wins the race.
- Other changes that modify the same file need to be rebased since Gerrit does not know to resolve the in-file merge.
- Lets see how this works.

A typical scenario for rebase

- Developer “D1” changes File “1” on Tip “T” and uploads a change “C1” in Gerrit first. https://review-android.quicinc.com/#change,8177. (Assume Change 8177 is in gerrit before status is merged)
- Developer “D2” changes the same File “1” but on different line on the same tip Tip “T” and uploads a change “C2” to Gerrit after D1 had uploaded. https://review-android.quicinc.com/#change,8178.
- Assumption : C1 is approved before C2. (8177 approved before 8178)
- End Result in Gerrit: C1 is verified and submitted but C2 fails to be merged due to a merge conflict.
- In the example that follows we assume File 1 has feature definitions and changes C1 and C2 are enabling and disabling two features, FA and FB.
- Now let’s study why D2’s change C2 failed to merge to mainline.

Patches vs Tip
when first uploaded to Gerrit

Tip T : File 1	D1 : File 1 Change C1	D2 : File 1 Change C2
• FA : on • FB : on	• FA : on • FB : off	• FA : off • FB : on

https://review-android.quicinc.com/#patch,sidebyside,8177,1,File1
https://review-android.quicinc.com/#patch,sidebyside,8178,1,File1

FA => Feature A
FB => Feature B

View of tip while C1(8177) in Gerrit review in D1's tree

Tip T : File 1	D1 : File 1	D1's Expected Tip T1 : File 1
• FA : on • FB : on	• FA : on • FB : off	• FA : on • FB : off

The diff to be applied
- FB : on
+ FB : off

The Tip T1 is what D1 sees while uploading the change C1. Note that the change C1 is not merged to mainline yet, but T1 is what D1 wants to be as the new Tip T.
https://review-android.quicinc.com/#patch,sidebyside,8177,1,File1 (Assume before submit.)

View of tip while C2(8178) in Gerrit review in D2's tree

Tip T : File 1	D2 : File 1	D2's Expected Tip T2 : File 1
• FA : on • FB : on	• FA : off • FB : on	• FA : off • FB : on

The diff to be applied
- FA : on
+ FA : off

The Tip T2 is what D2 sees while uploading the change C1. Note that the change C2 is not merged to mainline yet, but T2 is what D2 wants to be as the new Tip T.
https://review-android.quicinc.com/#patch,sidebyside,8178,1,File1

View of tip after C1(8177) is merged

Tip T : File 1	D1 : File 1	Tip T1, New Tip T : File 1
• FA : on • FB : on	• FA : on • FB : off	• FA : on • FB : off

The diff applied to tip
- FB : on
+ FB : off

The New T and Tip T1 is what the mainline is since D1's change got verified and is now merged to mainline.

Current status of C1 in Gerrit : https://review-android.quicinc.com/#change,8177

New Tip : http://git-android.quicinc.com/?p=tools/test/project0.git;a=blob;f=File1;h=d043fa6a112ea198f1e6a43bd74dfc0d6a00956a;hb=HEAD

View of tip against C2(8178) after C1(8177) is merged

New Tip T : File 1	D2 : File 1	D2's Expected Tip : File 1	Tip if an in-file merge was allowed.
• FA : on • FB : off	• FA : off • FB : on	• FA : off • FB : on	• FA : off • FB : off

D2's Anticipated diff to be applied

- FA : on

+ FA : off

The diff that happened against D2's tree in the end

- FA : on

+ FA : off

- FB : on

+ FB : off

Clearly we can see that D2's Expected Tip is not the same as the result when an in-file merge was allowed. D2 tested his change C2 with "FB : on" and not with "FB : off".

You can see C2 https://review-android.quicinc.com/#change,8178 could not be submitted after verification in Gerrit from the Comments section because Gerrit did not allow the merge to happen.

Result of allowing in-file merge in Gerrit

- From the previous slide D2's view of tip in the end is not the same as that when he uploaded the change.
- Both D1 and D2 got their patch applied to mainline but neither of them tested that the combined patches would work.
- Assume a case where D1 disables Feature B by replacing "FB : on" with "FB : off". He feels this combination with Feature A in line 1 would work fine.
- But D2's change disables Feature A by replacing "FA : on" with "FA : off". He feels this combination with Feature B in line 2 would work fine.
- After in-file merge both Feature A and Feature B is disabled and the functionality is broken while both D1 and D2 think only one of them have to be enabled. Now we need D3 to triage the failure.
- Who takes the blame? The tools? ☺

Problems faced by D1, D2, D3 now.

- Firstly one or many of developers have to compromise in letting the change of the other be merged first.
- D2, D3 rebasing their changes C2, C3 on top C1 if C1 got merged first if all three had changes on the same File 1.
- D2 rebases C2 again on top of C3 change if C3 got merged to mainline second.
- Man hours wasted in rebasing, review, approval, verification and submit.

Advantages on not allowing in-file merge in Gerrit.

- From Slide 7 we can clearly see that if we allow an in-file merge to happen *__there is a chance__* for the functionality to be broken.
- Triaging a broken functionality takes more time than doing a trivial rebase.

What we're doing to help

- Starting with the next Gerrit version (will be deployed on September 20th) you'll no longer need to email ABAIT if you upload a new patch set after a previous failed verification. Your new patch set will automatically be picked up once it is approved.
- In the following Gerrit version (exact date TBD, but should be mid-October) trivial rebases will preserve approval bits (and possibly verification bits as well). This will automate the current "send ABAIT your patch-ids" process.

Thank You!

- We hope that this PPT has given a substantial explanation of the need for multiple rebases on the same change and why the tools behave this way.