### Useful git documentation

# **Add Below to your .****gitconfig file , to use meld as difftool and mergetool**

# Add the following to your .gitconfig file.

[diff]

tool = meld

[difftool]

prompt = false

[difftool "meld"]

cmd = meld "$LOCAL" "$REMOTE"

# Add the following to your .gitconfig file.

[merge]

tool = meld

[mergetool "meld"]

# Choose one of these 2 lines (not both!) explained below.

cmd = meld "$LOCAL" "$MERGED" "$REMOTE" --output "$MERGED"

#cmd = meld "$LOCAL" "$BASE" "$REMOTE" --output "$MERGED"

**Git command :**

git config core.fileMode false

Ignore file permission changes in git

**Git command :**

git remote add rezlive https://dipendu\_rezlive@bitbucket.org/rezlive/rezliveagile.git

- Use to add a remote repository

**Git command :**

git remote

- Show all remotes

**Git command :**

git fetch rezlive

Fetch remote data : The remote’s there, but you don’t yet have any information about it. You need to fetch that.

**Git command :**

git config --global mergetool.keepBackup false

After performing a merge, the original file with conflict markers can be saved as a file with a .orig extension. If this variable is set to false then this file is not preserved.

Defaults to true (i.e. keep the backup files).

**Git command :**

**git stash**

Save your changes to stash

To record the current state of the working directory and the index, but want to go back to a clean working directory.

**git stash list**

List all stashes

**git stash show**

Show changes of last stash

**git stash apply**

Restore last stash on top of a different commit

**git stash pop**

Remove and Restore last stash on top of a different commit

**git stash pop stash@{1}**

Remove and Restore stash based on index on top of a different commit

**git difftool**

Shows file changes in gui tool instead of terminal if , tool is setup in .gitconfig file

### **List remote branches**

**git branch**

shows your local branches.

**git branch -a**

To see a list of them all, including the remote branches:

**git reset**

This will unstage all files you might have staged with git add:

**git checkout .**

This will revert all local uncommitted changes (should be executed in repo root):

**git checkout [some\_dir|file.txt]**

You can also revert uncommitted changes only to particular file or directory:

**git reset --hard HEAD**

Yet another way to revert all uncommitted changes (longer to type, but works from any subdirectory):

**git clean -fdx**

This will remove all local untracked files, so only git tracked files remain:

WARNING: -x will also remove all ignored files, including ones specified by .gitignore! You may want to use -n for preview of files to be deleted.

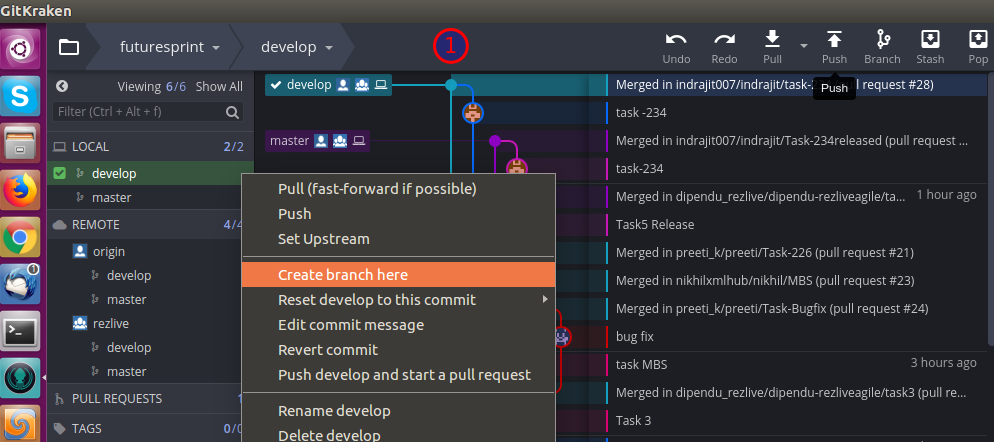
**git branch -D release/test**

To remove a local branch from your machine

**git push origin --delete release/test**

To remove a remote branch from the server:

1) Create new branch from develop branch.

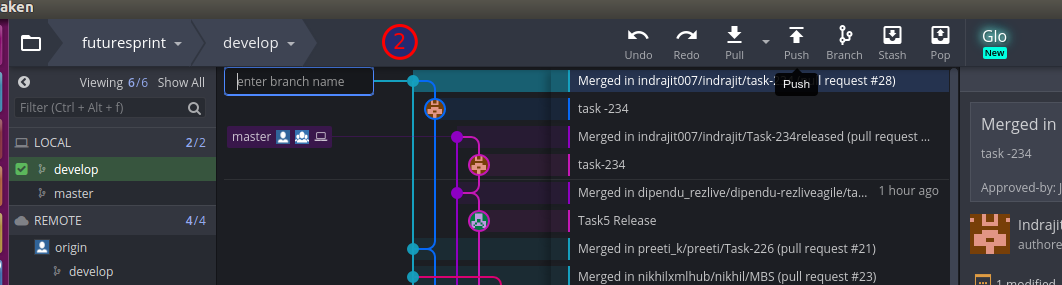


**Git command :** git checkout -b dev/TASK-276 develop

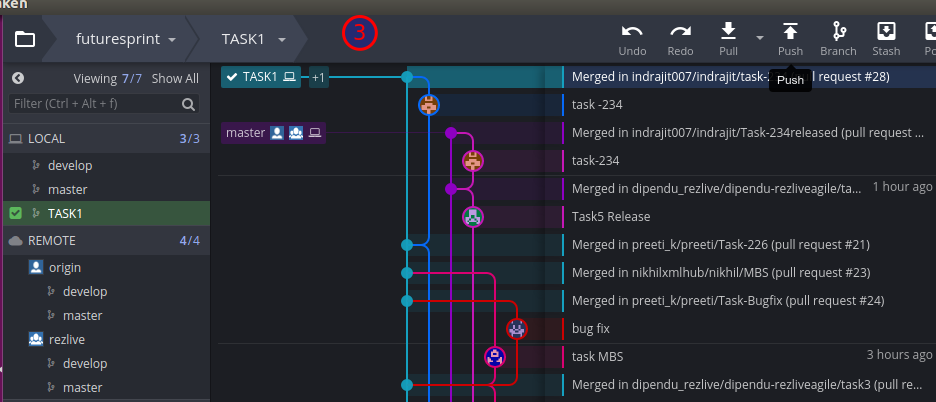
- Creates dev/TASK-276 branch from develop and checkout

- The -b flag tells it to create a new branch at the same time

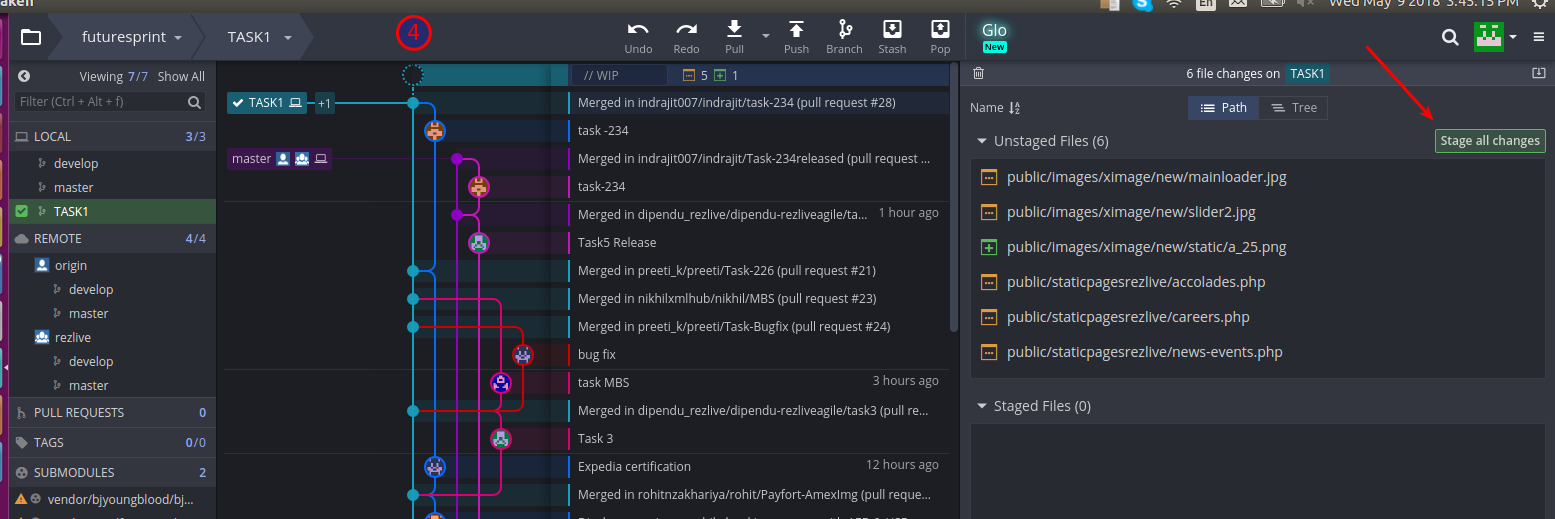
2) Enter your new Jira Task No.



3) Select your Task branch (**TASK1**)



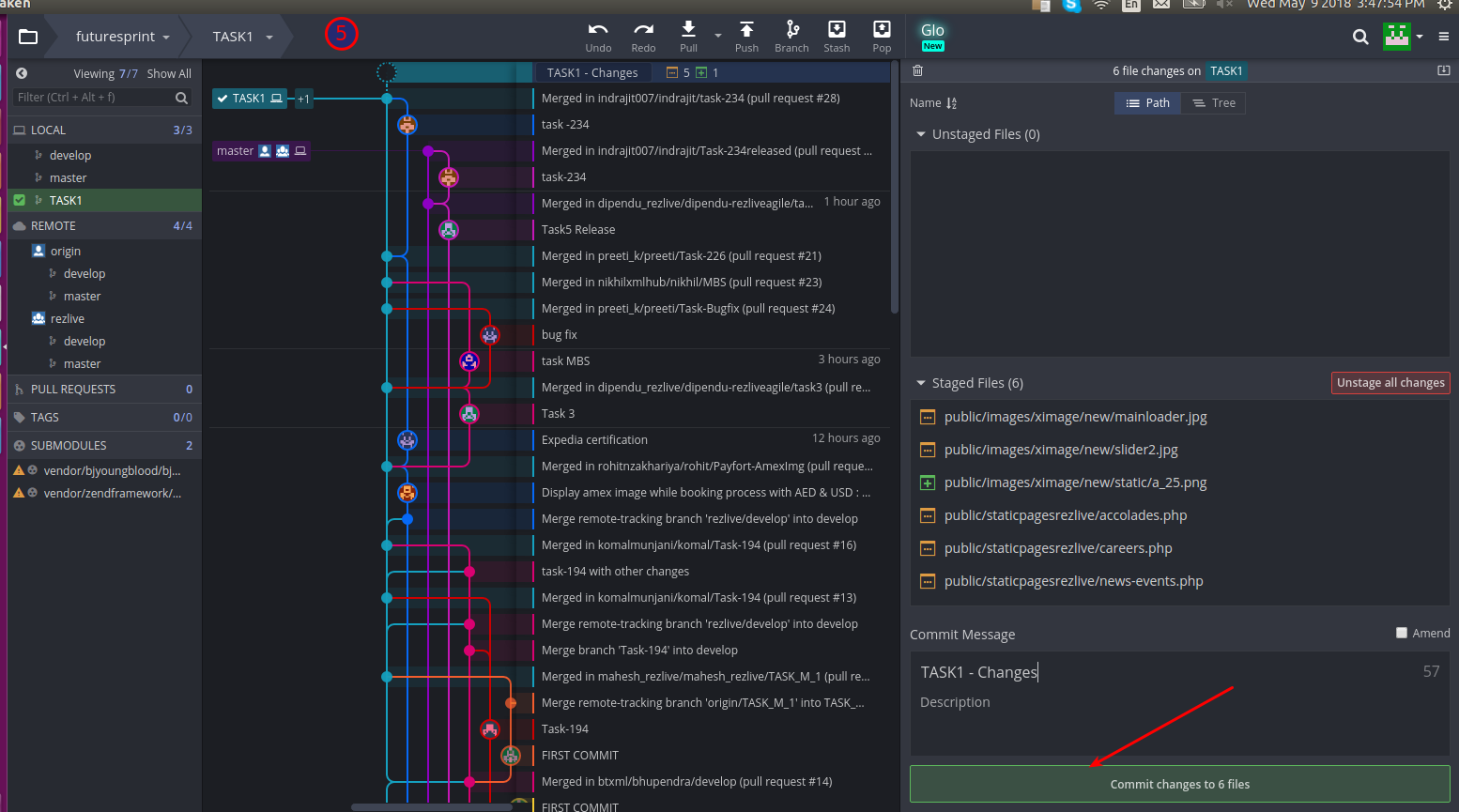
4) Click stage all changes.



**Git command :** git add -A

- Adds new files for commit if any

5) Enter your comment and commit changes.



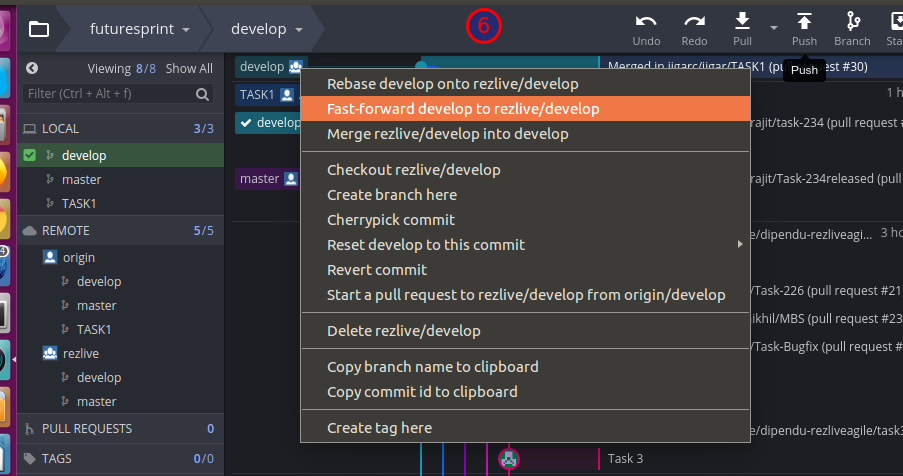
**Git command :** git commit -m "TASK-276"

- Commit in local repo

**Git command :** git commit --amend -m "TASK-276"

- Amends new changes in previous commit

6) Shift to branch develop and right click on rezlive develop , then click on fast- forward develop to rezlive/develop.

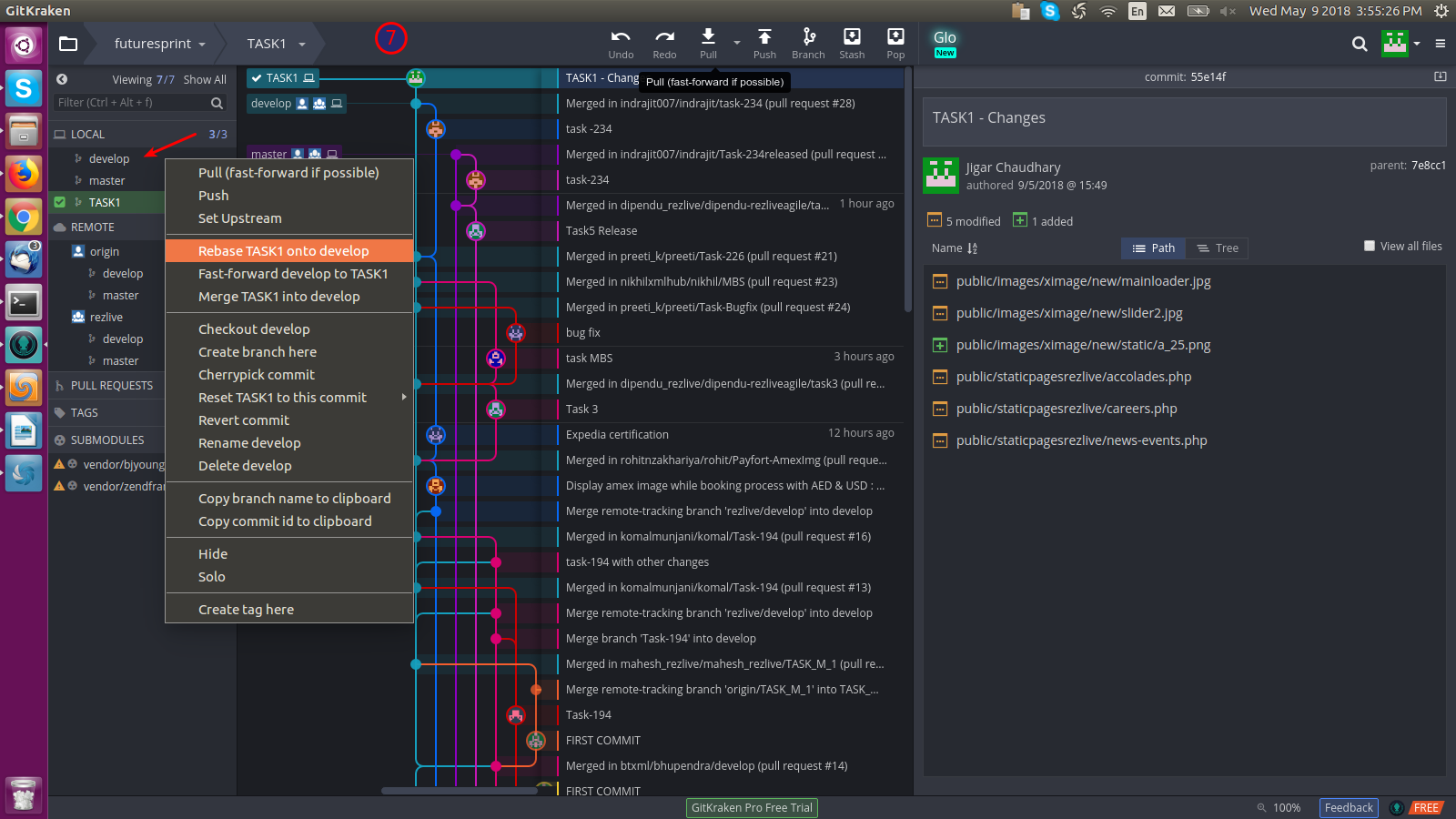


**Git command :**

git checkout develop

git pull --ff-only rezlive develop

7) shift to TASK1 and right click on develop then you can find **Rebase – TASK1 onto Develop**



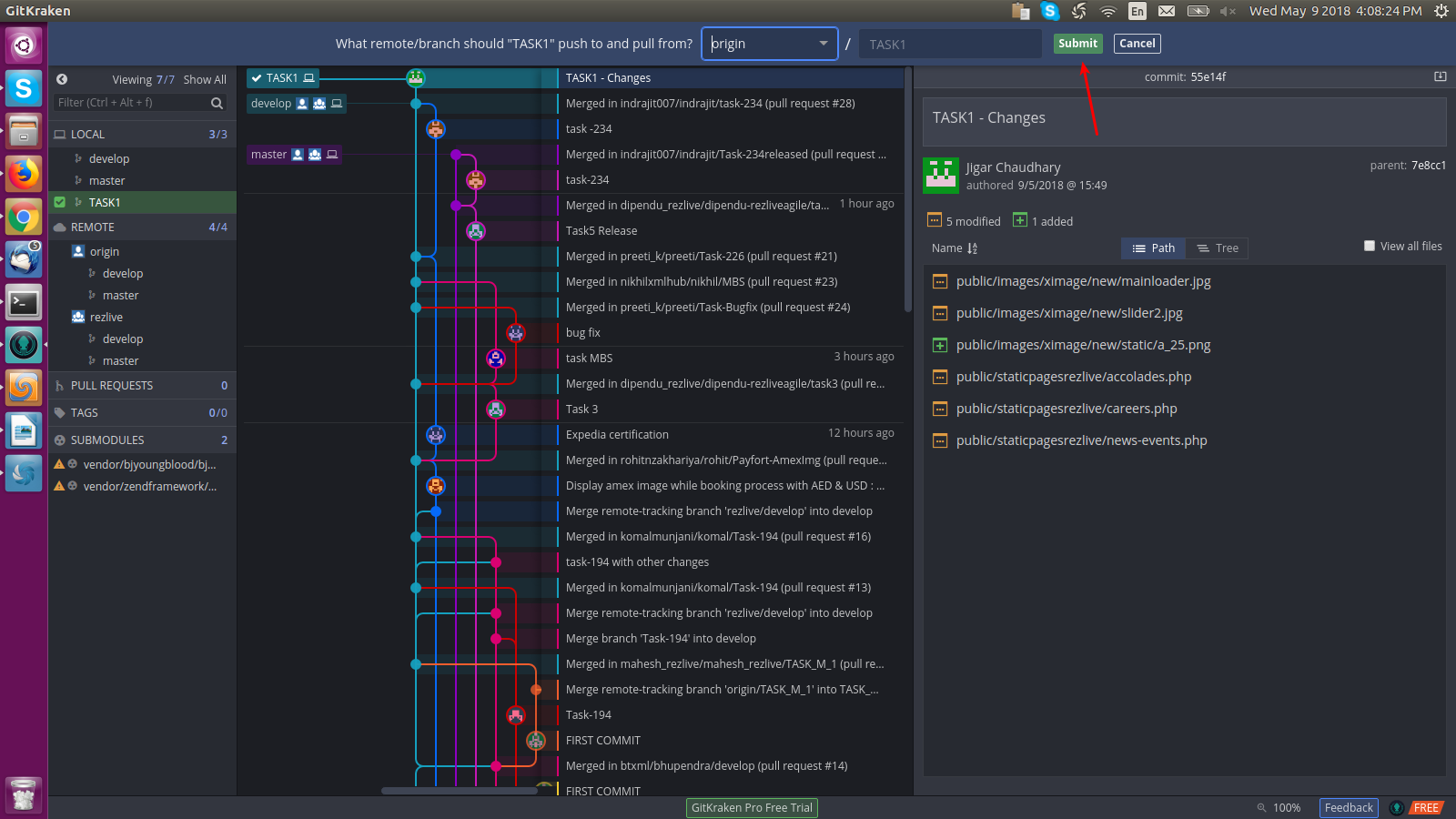
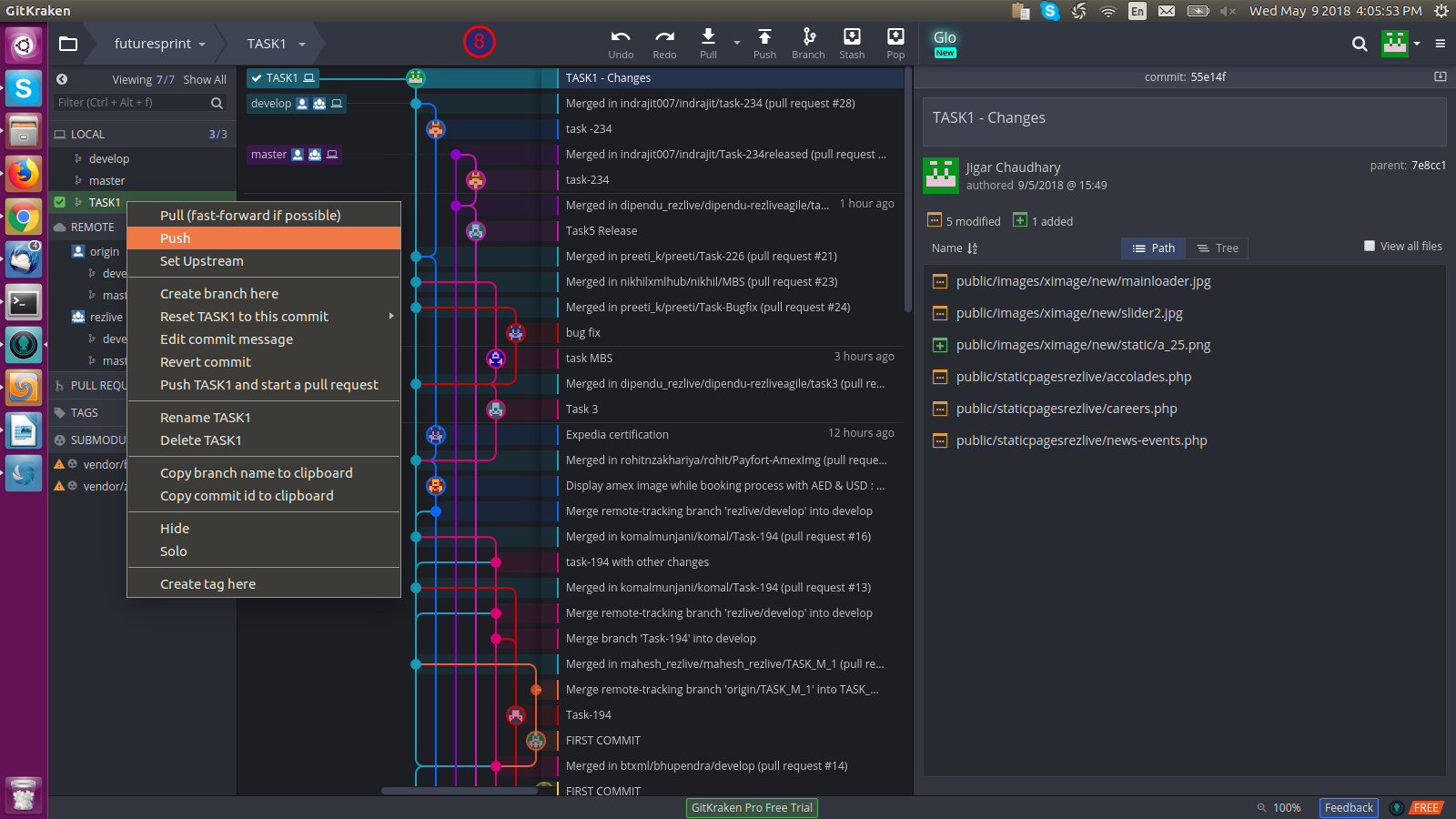
**Note: Resolve all the conflicts and commit all changes.**

**Git command :**

git checkout develop/TASK-276

git rebase develop

8) Right click on TASK1 and click on push and then submit

****

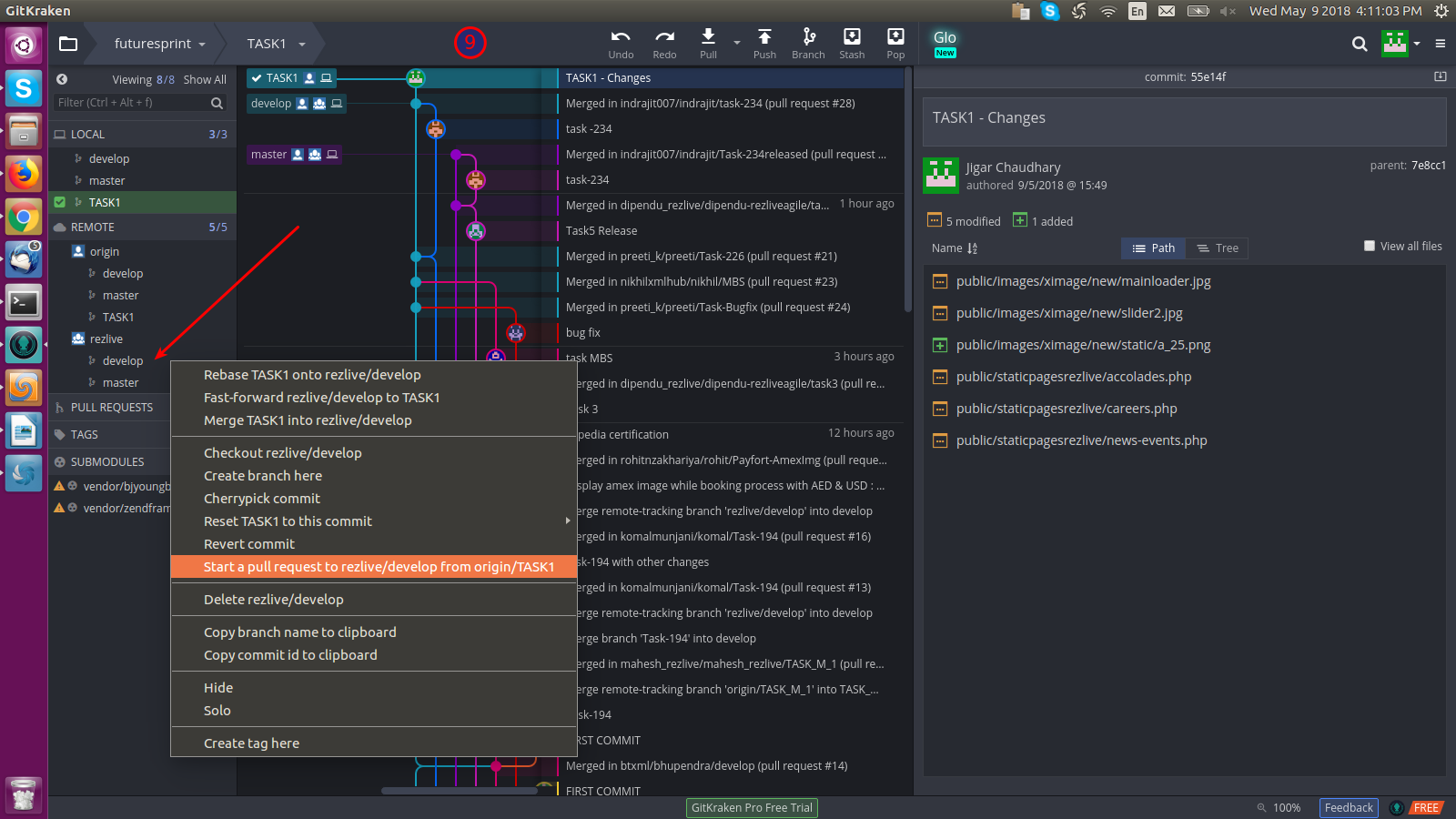
**Git command :**

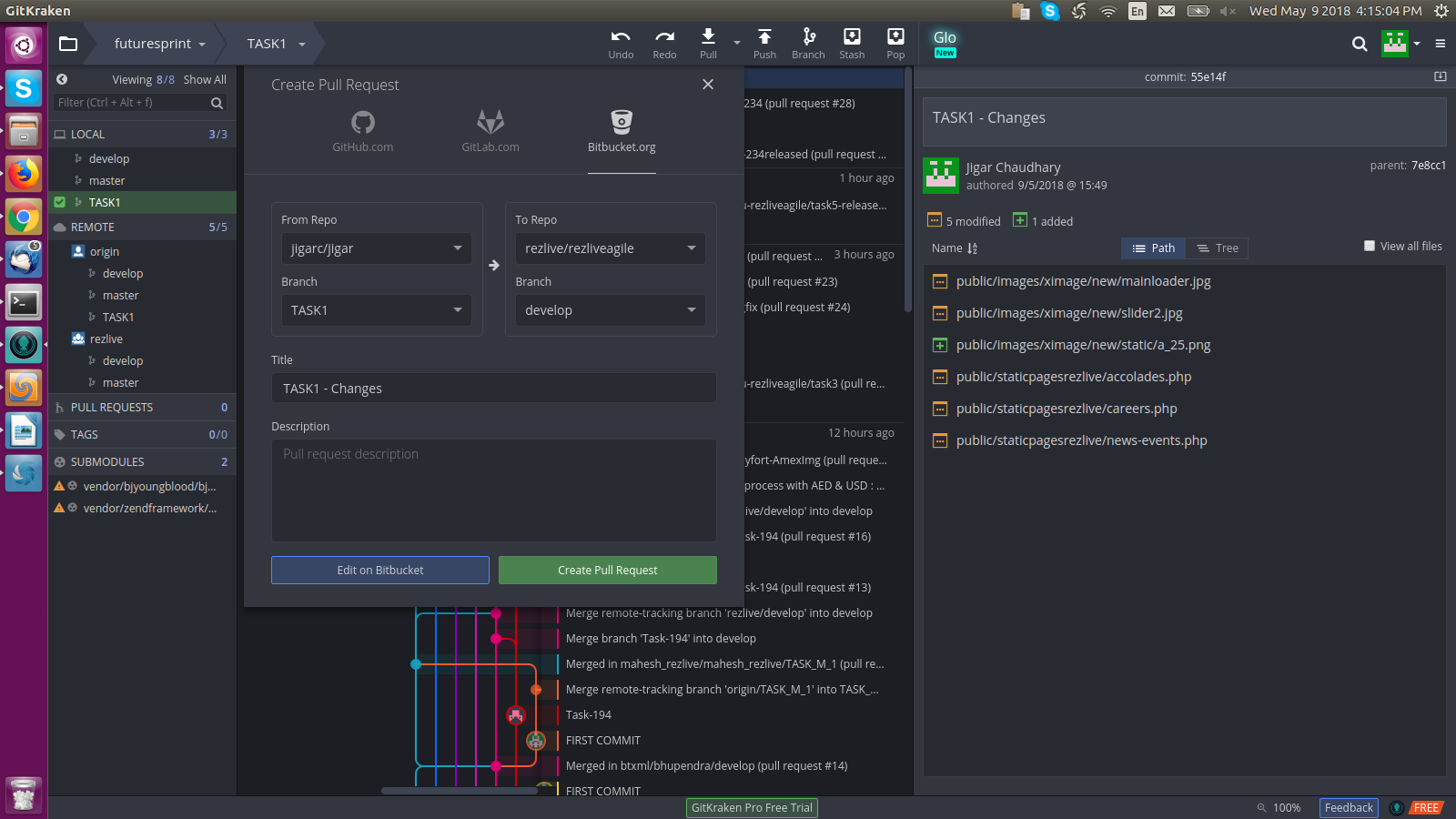
git push origin develop/TASK-276

- Above command will create a link to create a pull request on web interface.

9) Right click on Rezlive-develop and then click on start a pull request to rezlive/develop from origin/task1

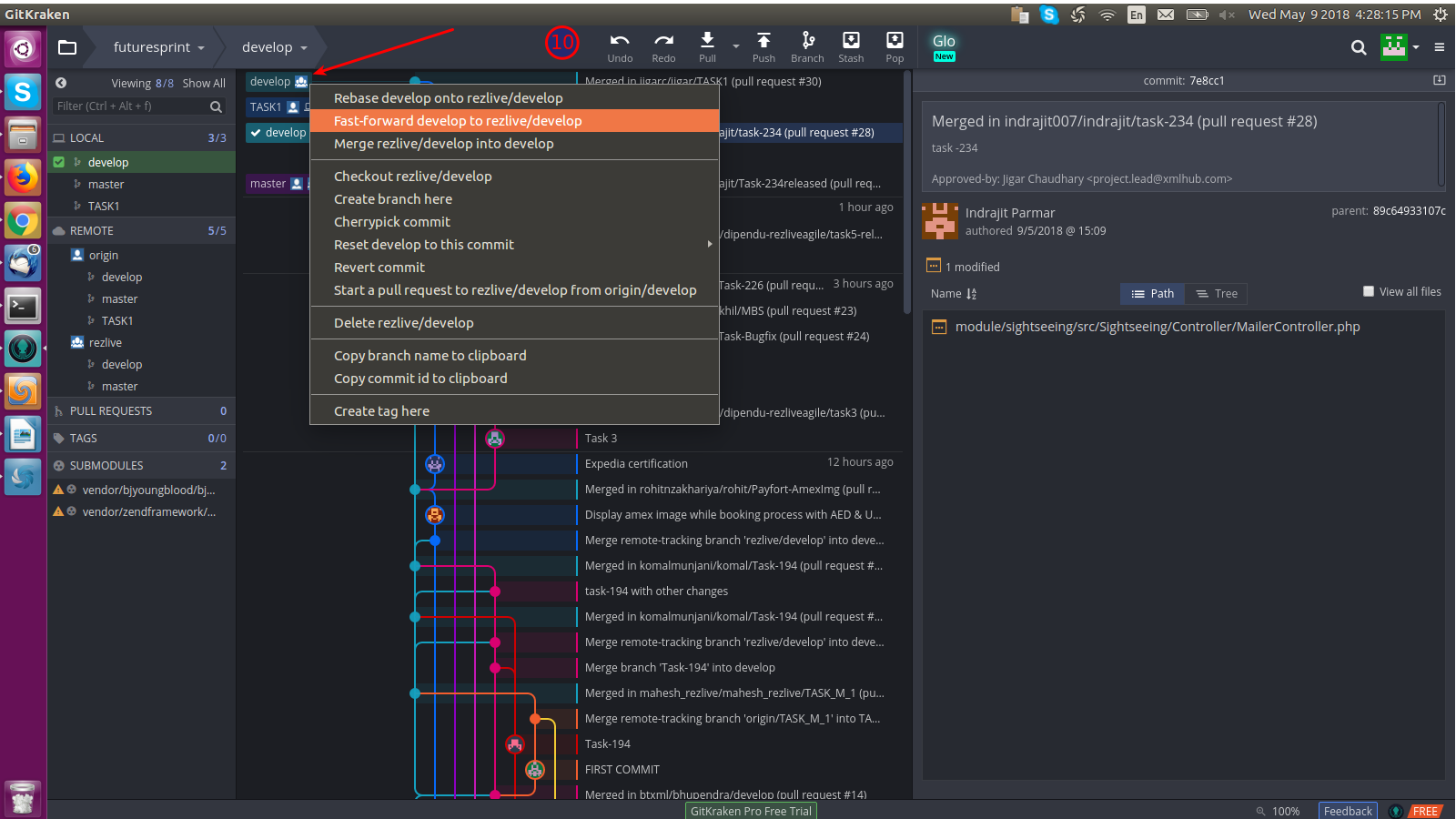
- Then click on create pull request



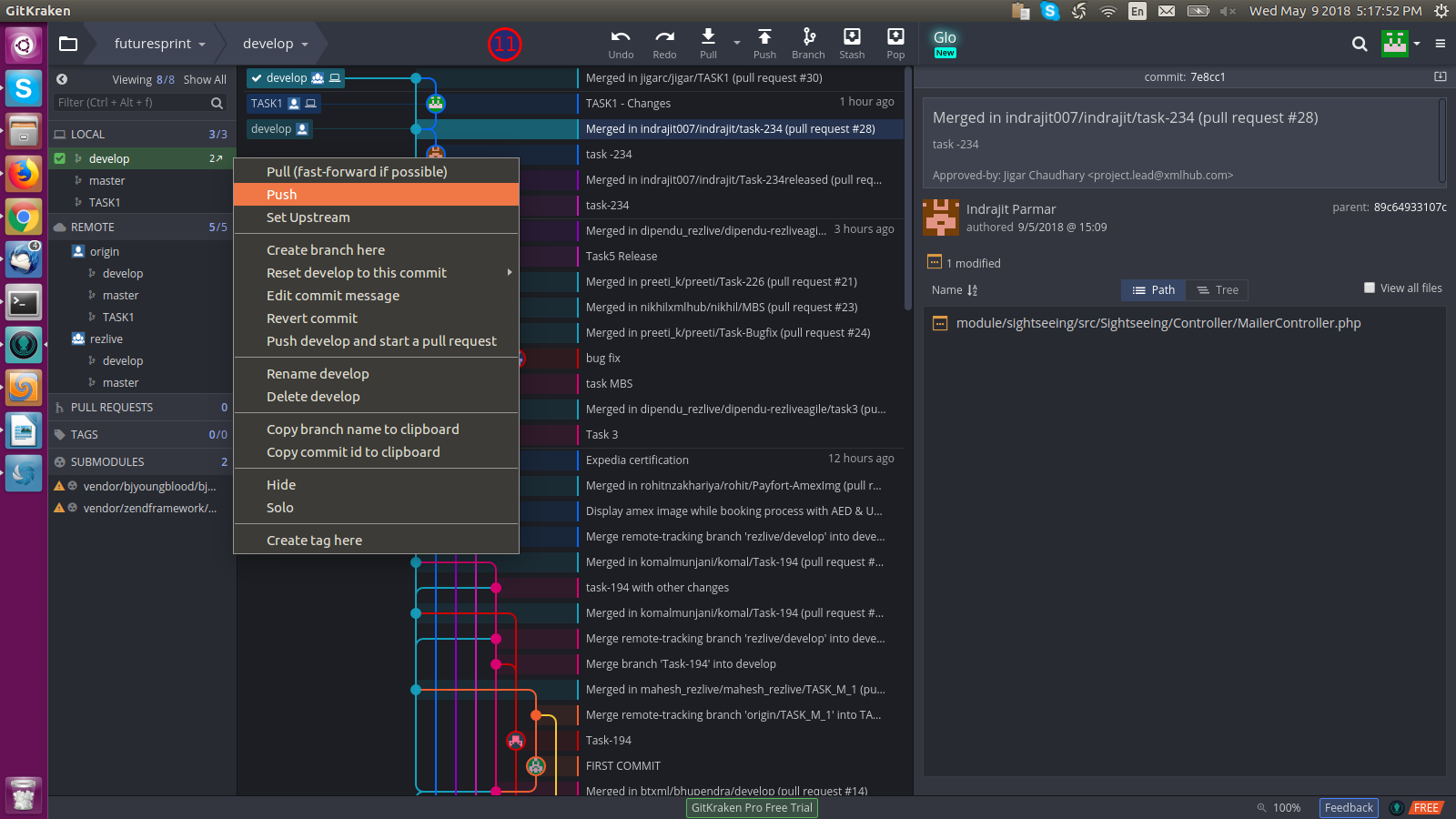


**Note** : After Pull request is approved and merged to rezlive develop, follow next steps.

10) Switch to local **develop** branch and right click on **rezlive-develop as shown in fig.** and then click on **fast-forward develop to rezlive/develop**

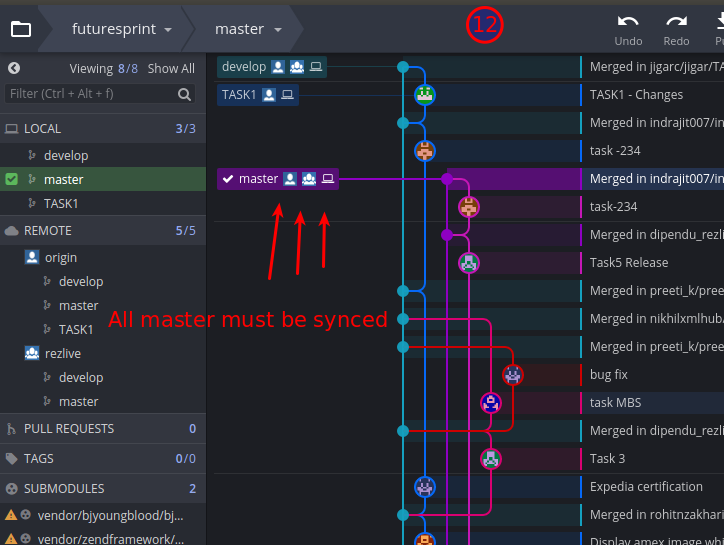


11) Push develop branch to origin : Right click on develop and click on push



**\* After QA approval, steps to move TASK1 to production**

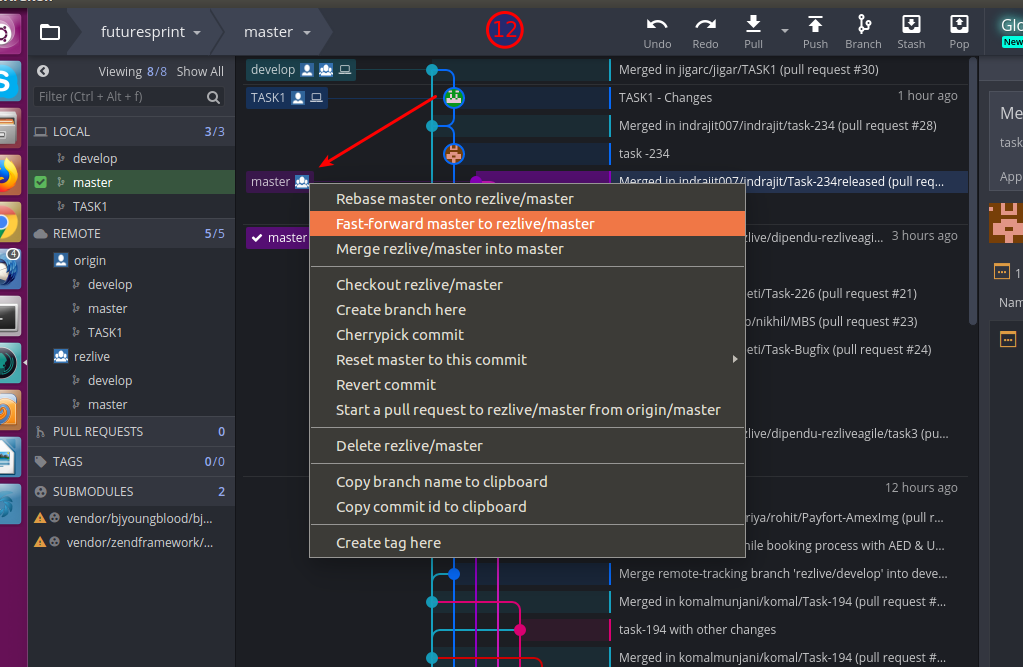
12) Make sure your local master, origin master is synced with rezlive master.



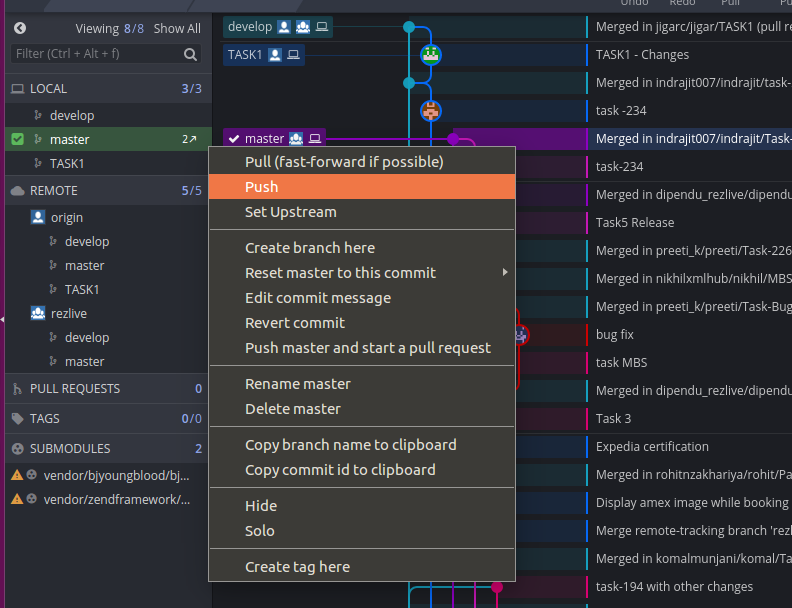
**OR**

If not follow below steps to sync local master and origin master.

- Switch to master branch , and then Right click on rezlive master and click on **fast-forward master to rezlive/master**

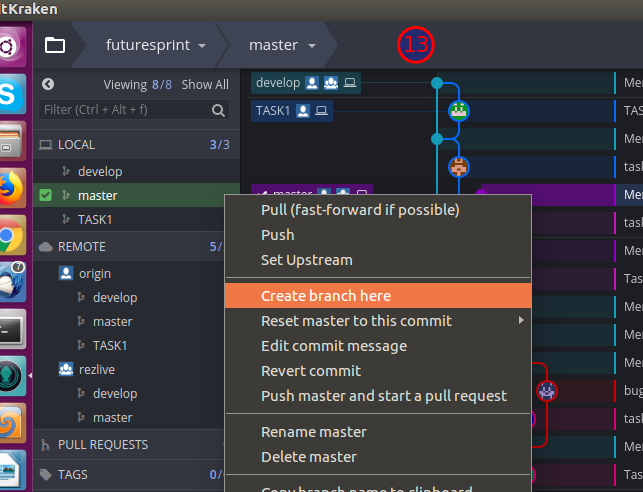


**-** then right click on local master and then click push

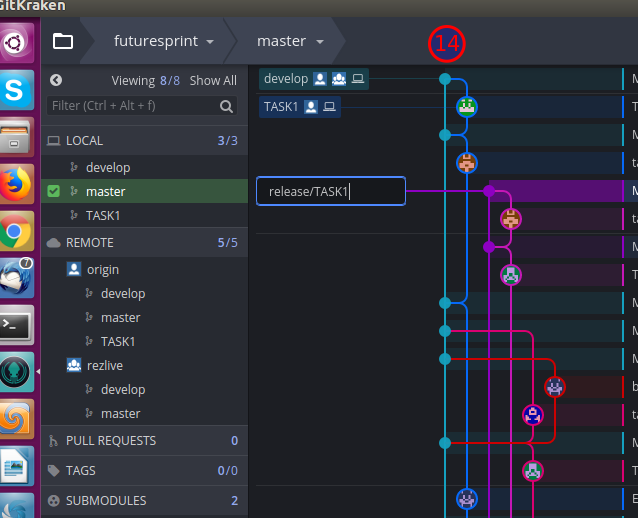


Now master will be synced

13) Switch to master if not done, right click on local master and click on create branch here.

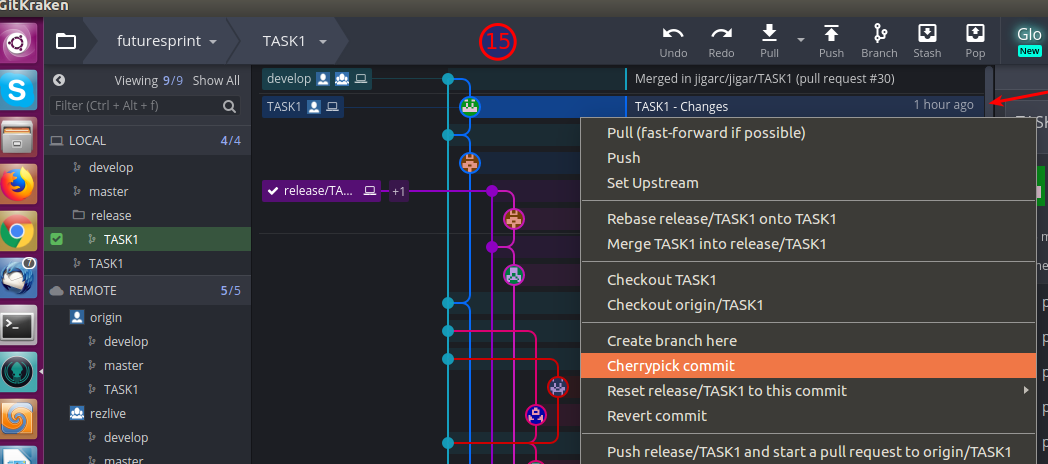


14) Name branch as “release/TASK1”.

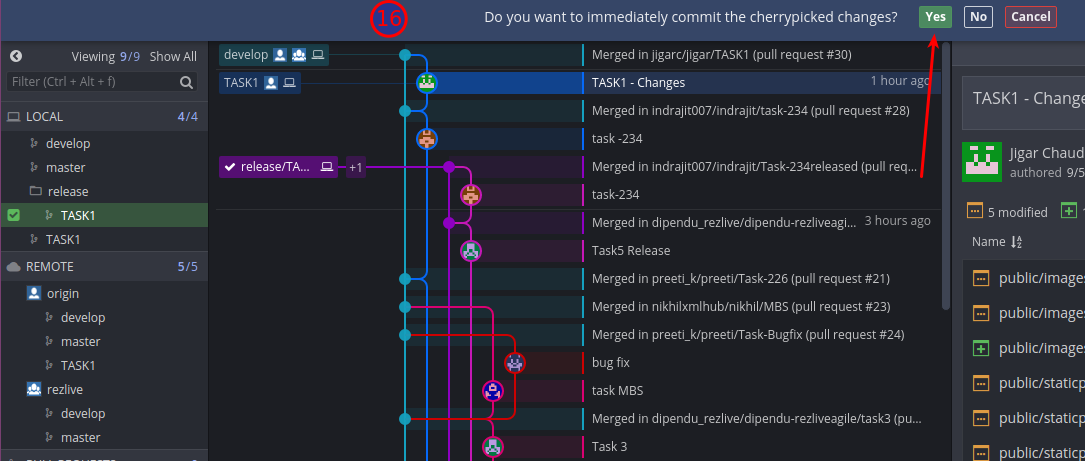


15) Checkout feature/TASK1 branch and then merge only your task specific files and changes using Cherry pick commit.

- Right click on your task commits and then click on cherrypick commit



16) Click on Yes



**Git command :**

git checkout dev/TASK1

Checkout your task branch.

**Git command :**

git log

Shows Commit History with commit id – Copy your commit ids you want to cherry pick into your release/task branch.

Enter “wq” to exit.

git log --author="dipendu"

To view commits from dipendu

git log --stat --author="Dipendu"

To view commits with files modified from dipendu

**Git command :**

git checkout dev/TASK1

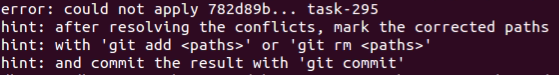
Checkout your task branch in which you want to cherry pick those commits.

**Git command :**

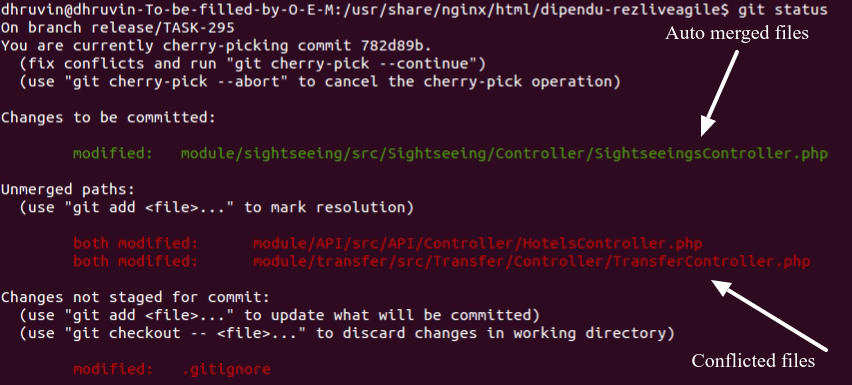
git cherry-pick 4ca1bdcc8c52e98f0fd93f1a76f0b778233d1076

Cherry picks your selected commit id in your current branch.

- If you receive below error, conflicts need to be resolved.



- **git status** command shows below two conflicted unmerged files



**Resolve file conflicts :**

**Git Command:**

**git mergetool**

Resolve conflict. As described in below .gitconfig,

cmd = meld "$LOCAL" "$MERGED" "$REMOTE" --output "$MERGED"

Left File is your : $LOCAL is the file in the current branch (e.g. master).

Center File us your : $MERGED is the partially merged file with the merge conflict information in it.

Right File is your : $REMOTE is the file in the branch being merged (e.g. branch\_name).

**IMPORTANT :**

**Check in merged file that you keep only your changes and if there are others task changes, kindly remove it.**

**Tip:** If Left and Center File shows major conflict and changes, skip auto merge for that file by copying the left file code to center and save it. Then manually add changes to those file.

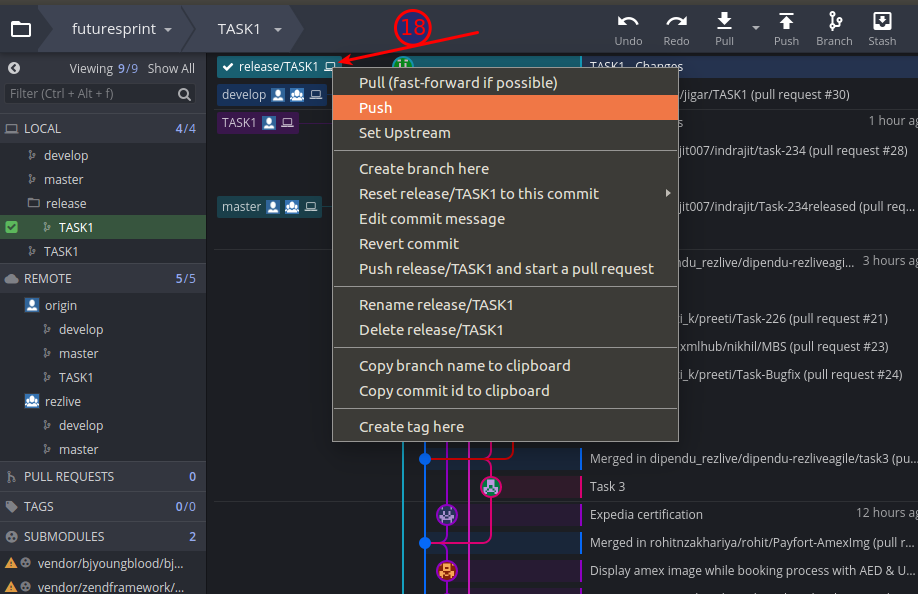
[For Reference](https://stackoverflow.com/questions/34119866/setting-up-and-using-meld-as-your-git-difftool-and-mergetool?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa)

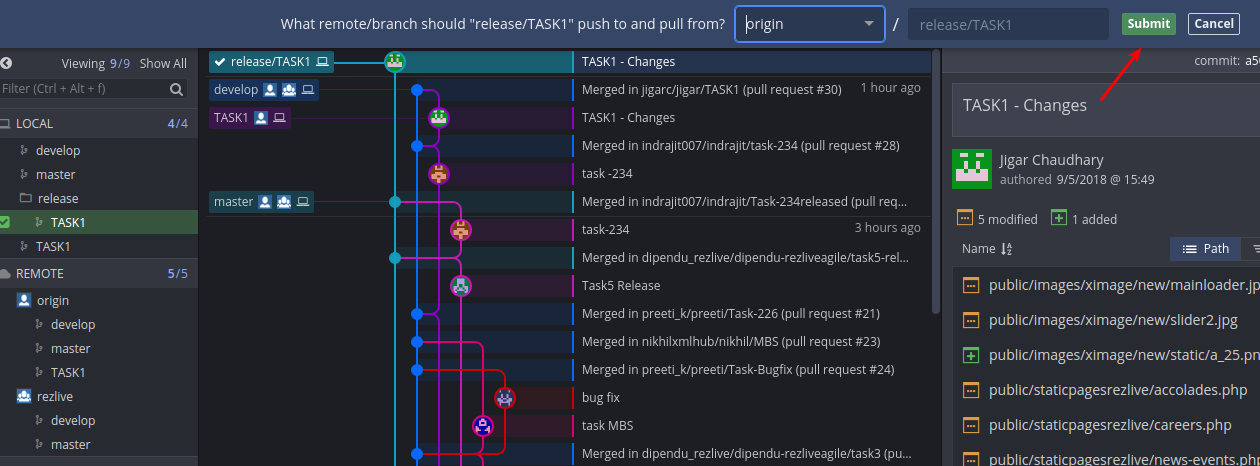
**To abort cherrypick commit**

git cherry-pick --abort

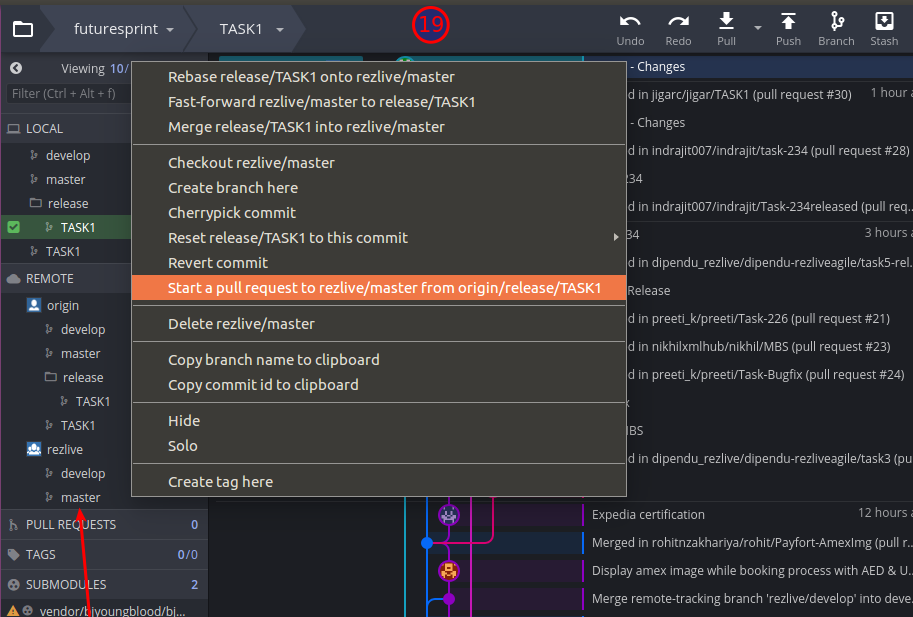
17) Repeat steps 15 and 16 if we have more than one commits

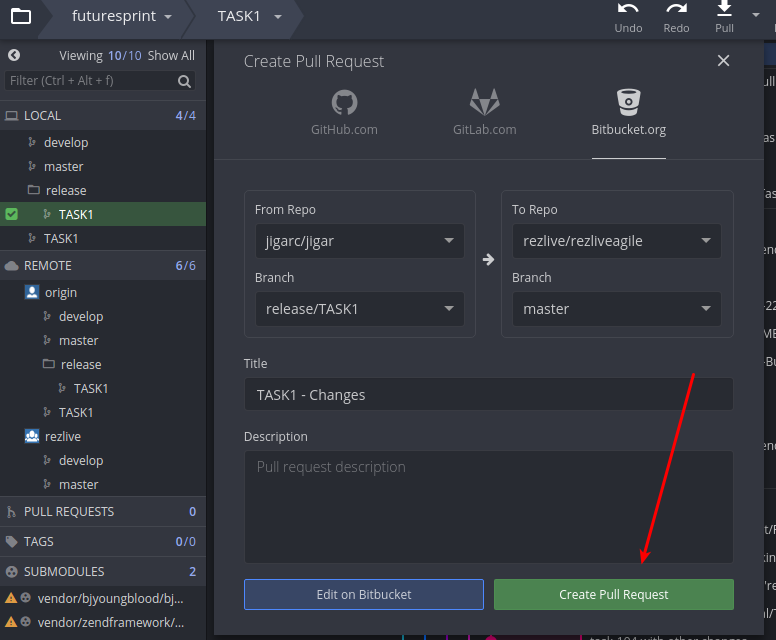
18) Push feature/TASK1 to origin.





19) Create a pull request to rezlive/master from origin/feature/TASK1.





Finally moved to Production.