**General notes:**

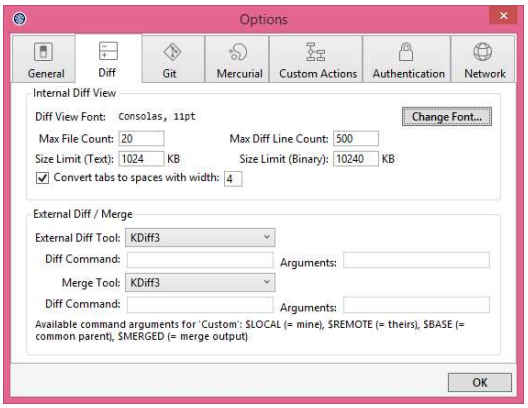
- The main purpose of these tasks is good knowledge of git-flow, which is necessary for passing task interview. It is assumed that git basics are already known;
- If during these tasks implementation mistakes are found, or you have suggestions for improvement, let the mentor know;

**Documentation notes:**

- http://nvie.com/posts/a-successful-git-branching-model/;
- https://www.atlassian.com/git/tutorials/merging-vs-rebasing/workflow-walkthrough
- https://git-scm.com/book/en/v2 (also in russian https://git-scm.com/book/ru/v1)
- https://www.codeschool.com/courses/try-git

**Implementation notes:**

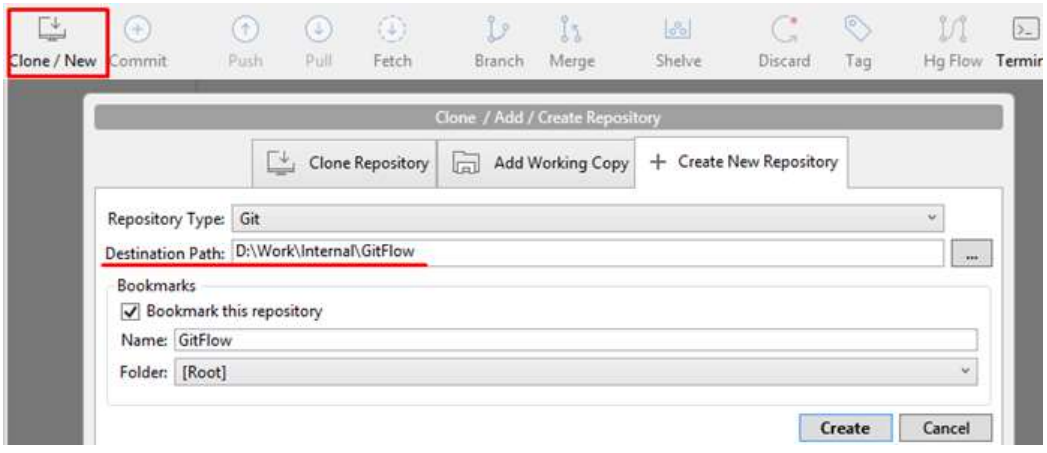
- You may wish to download and install SourceTree. It's a good UI for git. It's not required though. You may do everything via command line.
- If you have installed SourceTree, you may also like kdiff3 or diffmerge. You can set it as default merge tool in SourceTree.




- All commits must have meaningful comments. First words: "Task#[1] "
- Please do not delete branches. Push them all to remote repository
- Please merge all your changes with "--no-ff" flag.
- Please configure globally your email and name in git. Use FirstName LastName style and you@bsu.by email. (*git config --global user.name " FirstName LastName"; git config --global user.email you@bsu.by*)

**Task 1: Create Git Repository**

- Create local git repository (*git init*).

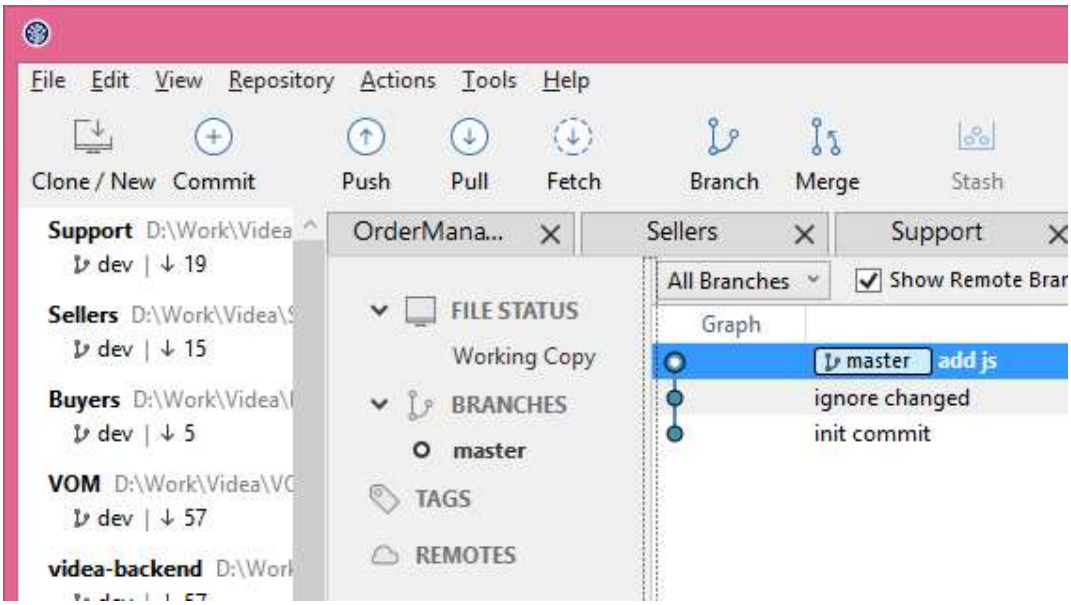


- Create remote repository empty project.
- Create **master** branch (*git checkout -b master*).
- Create index.html file.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8"/>
    <title>Git task</title>
</head>
<body>
    <h1>Hello World!</h1>
</body>
</html>
```

Index.html

- Commit your files to **master** with message (*git add --all*); (*git commit -m "initial commit"*). This will be the only direct commit to **master** branch, all the other work should be done in feature branches.
- Open SourceTree and look at your local repository (press F5 to refresh SourceTree info):



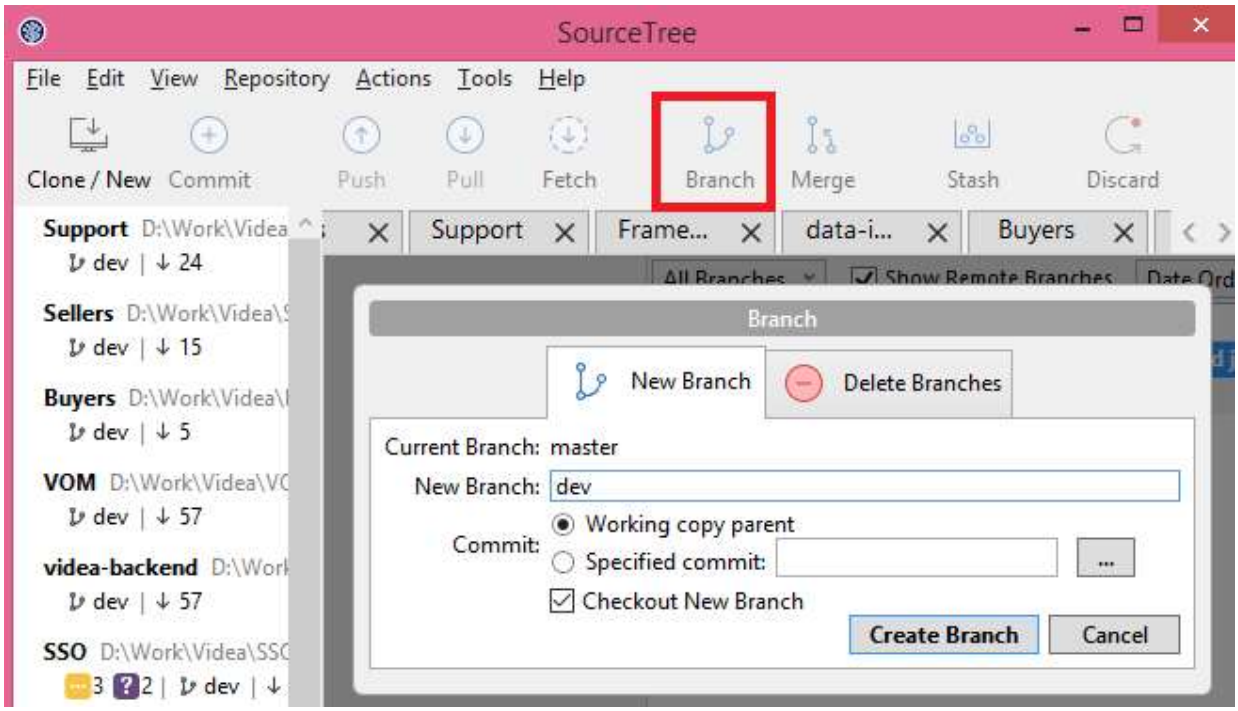

- Put your repository to remote server. Use github. GO to Repository -> Add Remote… Add Remote path.

Alternatively, you can do it using git console: (*git remote add origin http://guthublink*) – origin – name of remote repo. You can use any other name if you wish.

- Push your changes to the remote repository (*git push origin*)

So now, imagine other people join the project and you need to better coordinate your work. Direct commits to **master** are no longer reasonable in this situation. Git Flow to the rescue!

- Create **dev** and **release** branches in you repository. Both must have **master** branch as root. (*git branch dev*) (*git branch release*) Or, if you're using SourceTree, you can use a dialog to create new branches.




- Push new branches to the remote repository (*git push origin dev*) (*git push origin release*)

Task 2: Merge

- Checkout **dev**. Create branch **feature/hello-Alex** *(git checkout -b feature/hello-Alex)*
- Change index.html to

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8"/>
    <title>Git task</title>
</head>
<body>
    <h1>Hello Alex!</h1>
</body>
</html>
```

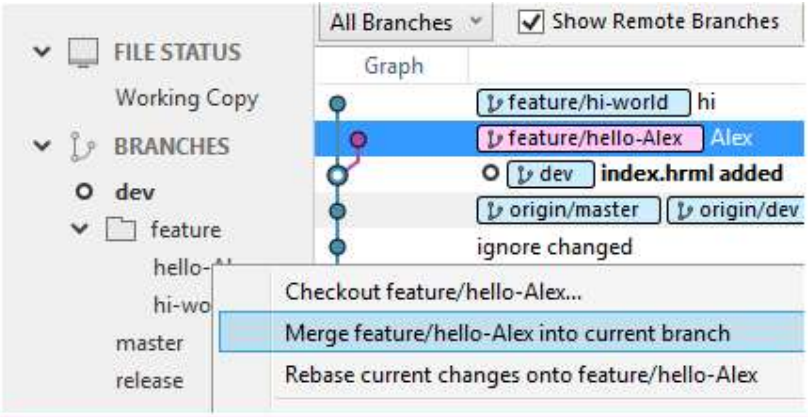
Index.html

- Commit your changes and switch back to **dev**
- Create new branch **feature/hi-world** and change index.html to

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8"/>
    <title>Git task</title>
</head>
<body>
    <h1>Hi World!</h1>
</body>
</html>
```

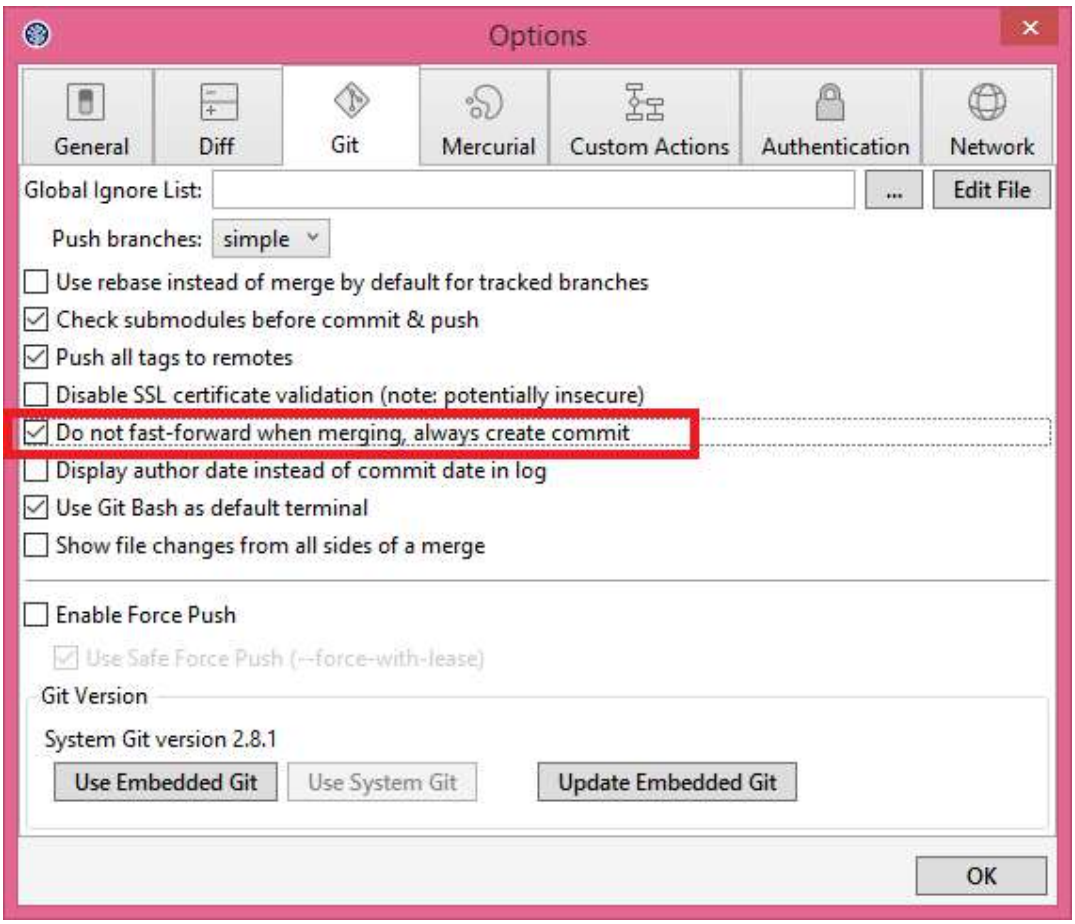
Index.html

- Merge **feature/hello- Alex** into **dev** with no fast forward flag. (*git checkout dev ; git merge --no-ff feature/hello-Alex*)



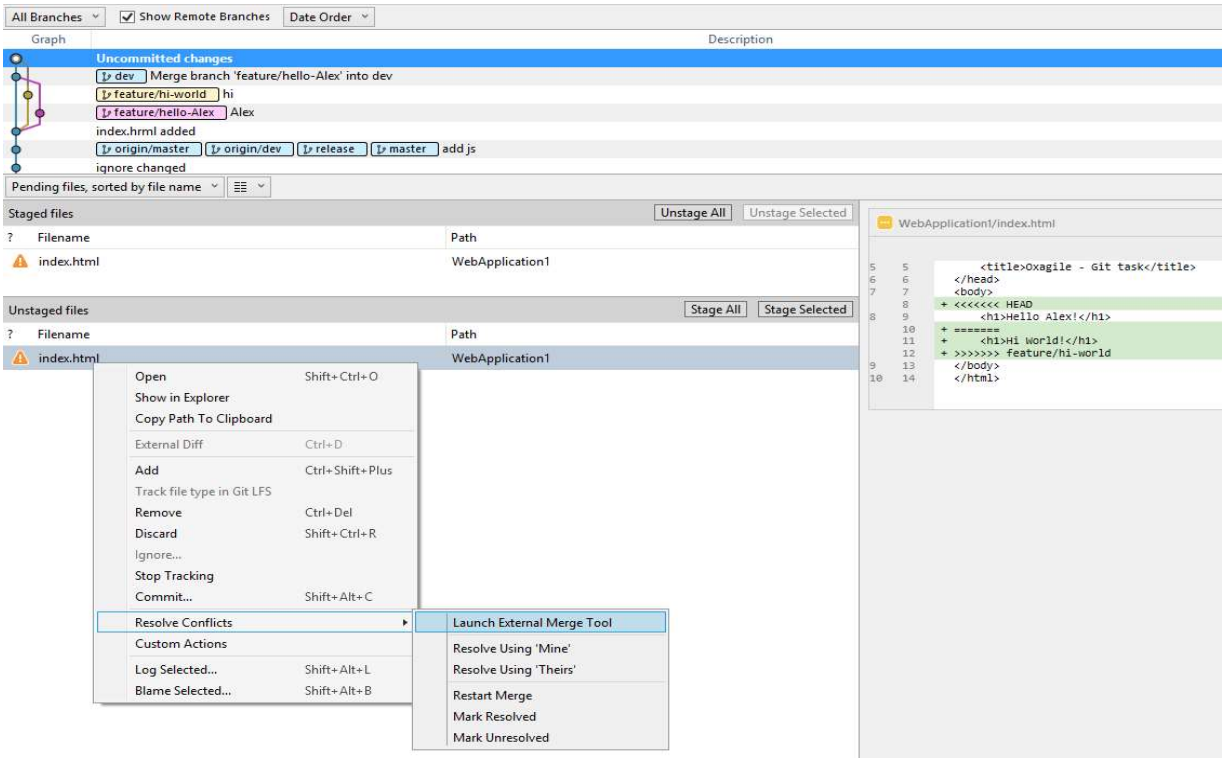


- You can set this flag by default in SourceTree: go to Tools -> Options -> Git




- Merge **feature/hi-world** into **dev** with no fast forward flag. (*git merge --no-ff feature/hi-world*)
- Use merge tool (kdiff3 or other) to handle merge conflict (*git mergetool --tool=kdiff3*) Probably, you will need to configure your mergetool. Run `git mergetool --tool-help` for the list of valid <tool> settings. If SourceTree already installed youg can run *git mergetool --tool=sourcetree.*

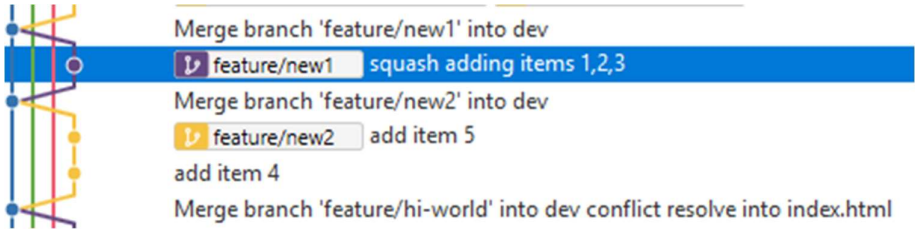
- In general, whenever you have merge conflicts you'll need a good understanding of what was the purpose of changes in each branch, so that you can reconcile these conflicting purposes.




- Need to know pros and cons of no-fast-forward merging.
- Push changes.

**Task 3: Rebase**

- Create new branch **feature/new1**. Make few commits (at least three) with changes to index.html. You can add <ul> with items.
- Squash different commits into single one (squacosh). [*git rebase –i HEAD~n* (where n – number of your commits)]
- Do not merge **feature/new1** branch into **dev** yet. Just create new branch (**feature/new2**) from **dev**, make few commits with changes to same <ul> tag as in **feature/new1** and merge with **feature/new2** branch into **dev** with no fast forward flag.
- Go back to **feature/new1** and try '*git rebase dev*' command. Use merge tool to resolve conflicts.
- Remember to run '*git rebase –continue*'
- Merge **feature/new1** into **dev** with no fast forward flag.
- Need to know pros and cons of rebasing and squashing.



**Task 4: Pull Request and rebase**

- Create new branch **feature/pull-request-rebase**. Change index.html title to “FPMI - Git task”

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8"/>
    <title>FPMI - Git task</title>
</head>
<body>
    <h1>Hi Alex!</h1>
</body>
</html>
```

Index.html

- Push your changes. Create pull request on github. Do not merge (accept) it yet.

In general the purpose of pull request (merge request) is to let someone review your changes and either propose improvements or accept you work. You can leave some comments in your PR just for fun.

- Assume you are waiting for your Pull Request approval and you need previous work (“FPMI - Git task” title). Checkout **feature/pull-request-rebase** and create new branch **feature/pull-request-rebase2**. Change index.html to “FPMI - Git task2” and commit your changes.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8"/>
    <title> FPMI - Git task2</title>
</head>
<body>
    <h1>Hi Alex!</h1>
</body>
</html>
```

Index.html

- Go to github. Open your pull request and approve it. This will merge changes from **feature/pull-request-rebase** into **dev**. Open your change log tree in Source Tree or in any other tool.
  - Pull merged code from remote server (*git checkout dev); (git pull*)
  - Checkout **feature/pull-request-rebase2** and rebase it onto **dev**. Investigate how the commits tree has changed. Merge **feature/pull-request-rebase2** with no fast-forward flag into **dev**. Push your changes.

**Task 5: Pull Request and merge**

- Create new branch **feature/pull-request-merge**. Change index.html to:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8"/>
    <title> FPMI - Git task2</title>
</head>
<body>
    <h1>Hi Alex!</h1>
    <h2>Merge it!</h2>
</body>
</html>
```

Index.html

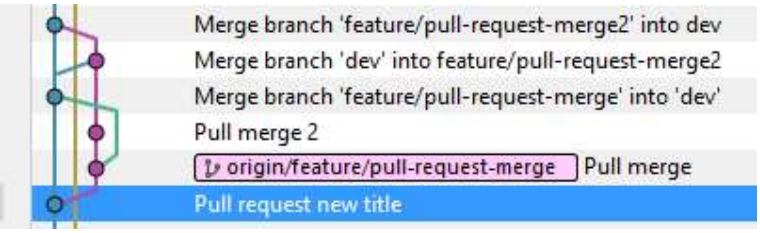
- Push your changes. Create pull request on github.

- Assume you are waiting for your Pull Request approval and you need previous work (“merge it!” h2). Checkout **feature/pull-request-merge** and create new branch **feature/pull-request-merge2**. Change index.html to “merge it2!”

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8"/>
    <title> FPMI - Git task2</title>
</head>
<body>
    <h1>Hi Alex!</h1>
    <h2>Merge it2!</h2>
</body>
</html>
```

Index.html

- Go to github. Open your pull request and approve it. This will merge changes from **feature/pull-request-merge** into **dev**. Open your change log tree in Source Tree or in any other tool.
- Checkout **feature/pull-request-merge2** and merge with no fast forward flag **dev** into this branch. Investigate how the tree has changed. Merge **feature/pull-request-merge2** with no fast-forward flag into **dev**. Push your changes.




**Task 6: Hot fix**

- Currently your **dev** branch has more commits, than **master**. You need to change index.html in **master** and **dev**, so that body looks like in next listing. The title in **master** should not be changed. How can you accomplish this?

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8"/>
    <title>Git task</title>
</head>
<body>
    <h1>Hi Alex!</h1>
    <div>You are the best!</div>
</body>
</html>
```

Index.html

**Solution**

**Task 7: Cherry pick**

- You like how title in index.html looks after Merge branch ‘**feature/pull-request-rebase2'** with no fast forward into **dev**.

```
<head>
    <meta charset="utf-8"/>
    <title>FPMI - Git task2</title>
</head>
```

Index.html

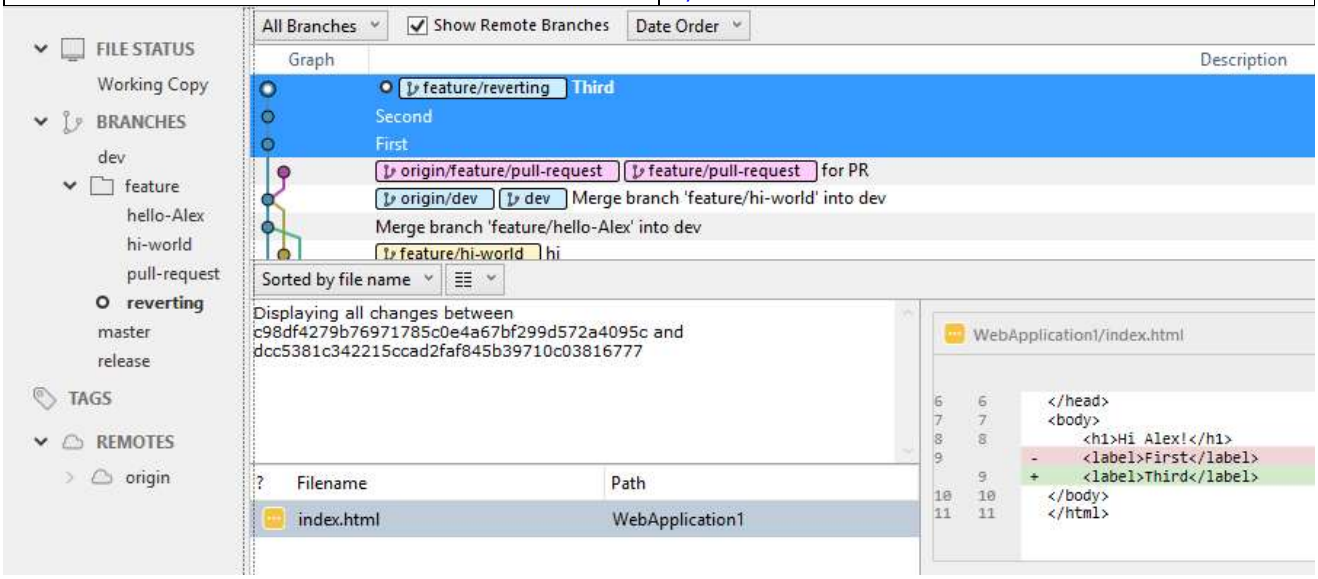
- How you can add only this change to **master** branch?

**Solution**

## Task 8: Revert changes

- Read this https://www.atlassian.com/git/tutorials/undoing-changes/git-checkout
- Checkout **dev** and create new branch **feature/reverting** and make four commits.

| Commit message | Change in index.html |
|---|---|
| First | `<select>`<br>`</select>` |
| Second | `<select>`<br>`  <option value="1">One</option>`<br>`</select>` |
| Third | `<select>`<br>`  <option value="1">One</option>`<br>`  <option value="2">Two</option>`<br>`</select>` |
| Fourth | `<select>`<br>`  <option value="1">One</option>`<br>`  <option value="2">Two</option>`<br><br>`</select>` |




- Reset Fourth commit. (*git reset --hard HEAD~1*)
- Revert Second commit. *(git revert [HASH], where [HASH] – your second commit's hash).*
- Result:

```
<select>
  <option value="2">Two</option>
</select>
```

- Merge your changes into **dev**. Push your work.

## Good commands to know:

| | |
|---|---|
| Git Checkout -b [branch] | |
| Git status | |
| Git branch | |
| Git fetch | |
| Git pull | |
| Git commit --amend | |
| Git merge --no-ff | |

| | |
|---|---|
| Git rebase | |
| Git cherry-pick | |
| Git reset --hard | |
| Git revert | |
| Git rebase --i HEAD~2 (squash, fixup) | |
| Git tag | |
| Git remote | |
| Git mergetool -t kdiff3 | |