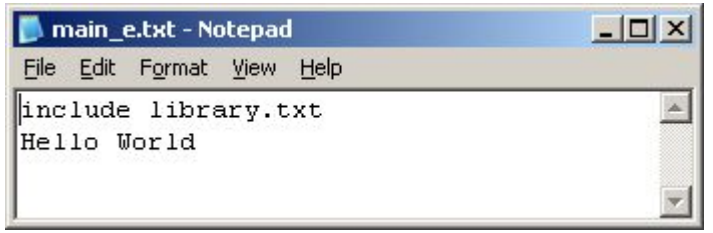
The Working Cycle

Exercise I: adding files

First say to Git who are you, this info will appear on each commit you do
$ git config --global user.name "harry"
$ git config --global user.email harry@gitlab.com

Please Create new (git init) „bare" repository to be your origin(remote) repository
$ git init --bare --shared ~/gitlab.<your-username>
Then clone it to harry repo: cd .git and overview repository objects
$ git clone ~/gitlab.<your-username> ~/harry

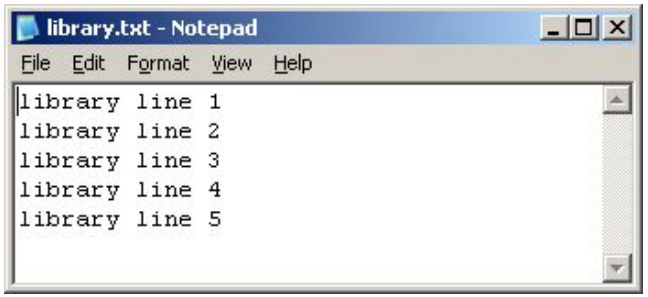
Go to harry folder and create two files in your working directory (fill them with these text lines):
$ cd ~/harry
`main_e.txt`
`library.txt`



Add your changes to the index
`$` git add *

Check your status before commit
$ git status

Commit your changes as „initial project"
$ git commit -m "initial project"

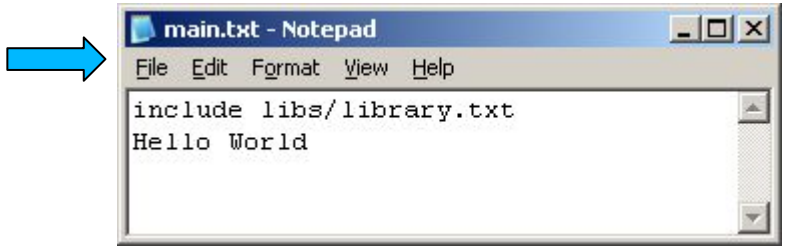


The Working Cycle
Exercise II: restructuring files

Rename main_e.txt into main.txt
$ git mv main_e.txt main.txt

Create a folder named libs
$ mkdir libs

Move library.txt into libs (modify main.txt accordingly!)
And add modified main.txt to Index
$ git mv library.txt libs
$ vi main.txt
$ git add *



Check your status before commit
$ git status

```
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       renamed:    library.txt -> libs/library.txt
#       new file:   main.txt
#       deleted:    main_e.txt
#
```

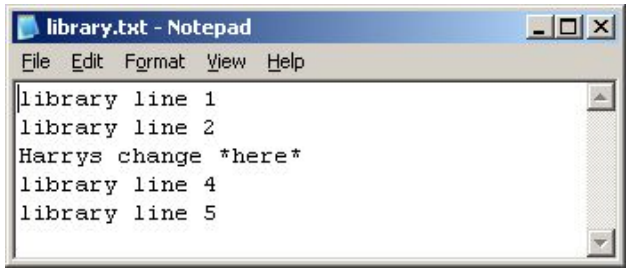
Commit your changes as „restructuring project"
$ git commit -m "restructuring project"

```
Counting objects: 7, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (5/5), 443 bytes, done.
Total 5 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (5/5), done.
To /nfs/iil/disks/iec_cm/git_repo/gitlab.git
   69a8b48..e2a900c  master -> master
```

Push all your work to origin (central) repo
$ git push origin master:master

The Working Cycle
Exercise III: teamwork

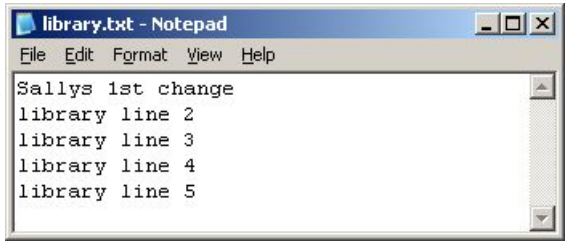
Change the third line in Harry‘s repo and check the status
$ git status



Commit Harrys changes as „Harrys 1st changes in our library“
$ git commit -a -m "Harrys 1st changes in our library“

Clone the gitlab repo again (cd .. and into a folder named „sally“)
$ git clone ~/gitlab.<your-username> ~/sally

Push all your work in Harry‘s repo to origin (central) repo
$ git push origin master:master



Change the first line in Sallys repo commit it and try to push
$ git commit -a -m “Sally’s 1st changes in our library“
$ git push origin master:master

Pull updates to Sallys repo:
$ git pull Overview what you got:
$ gitk Comapre what changes you got in „Commit Viewer“:

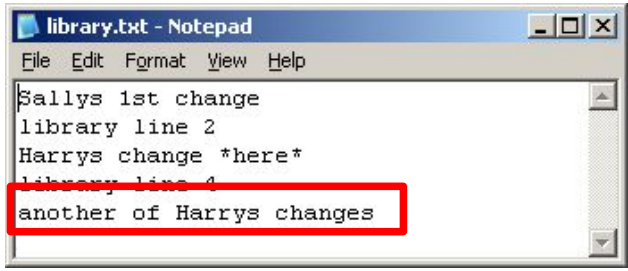
$ git diff HEAD^ HEAD – compare HEAD commit with previous commit

Push Sallys changes: $ git push

The Working Cycle
Exercise IV: conflicts

Pull Harrys repo to get sallys last changes: ~/harry] $ git pull

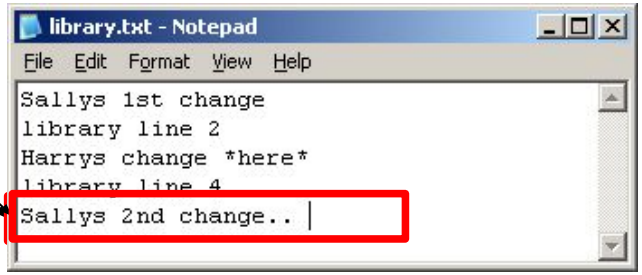
Change the fifth (last) line in Harrys repo, see your change:
$git diff HEAD



Commit Harrys changes as „Harrys 2nd changes in our library"
~/harry]$ git commit -a -m "Harrys 2nd changes in our library"

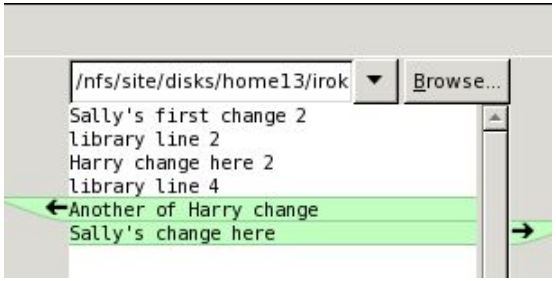
Push Harry's changes:~/harry]$ git status ~/harry]$ git push

Change the fifth (last) in Sallys repo commit and try to push
~/sally]$ git commit -a -m "Sally 2nd change in library"
~/sally]$ git push Why do you think it failed?



Pull Sallys repo: ~/sally]$ git pull

Edit and resolve the conflict: ~/sally]$ git gui -> Right Click library.txt
-> Run Merge Tool



Commit and push Sally's changes:
~/sally]$ git commit -a -m "Merge of Sally"

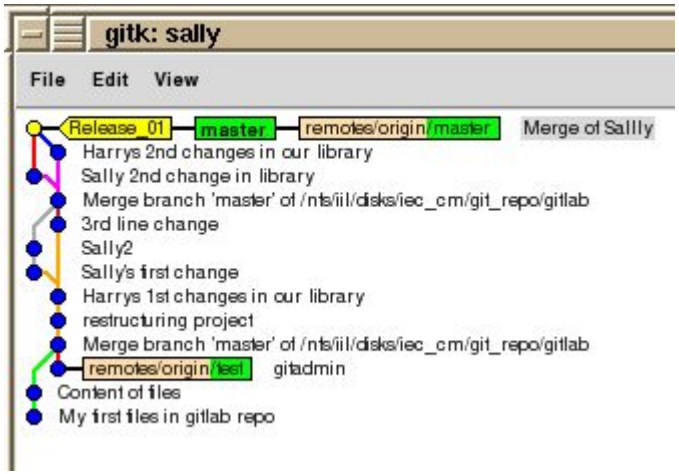
~/sally]$ git status ~/sally]$ git push

Exercise V Tagging

Create a tag named „**`Release_01_your-username`**“
with commit message "tag Release_01"
~/sally]$ git tag Release_01_<your-username> -m "tag Release_01_<your-username>“

Run gitk to see the Tag in yellow color
~/sally]$ gitk

Push the tag to origin
~/sally]$ git push --tags



Go to Harry repo, run „Commit Viewer“ gitk to see, that you still do not see the tag in Harry repo
~/harry]$ gitk

Pull the tag from the harry repo
~/harry]$ git pull

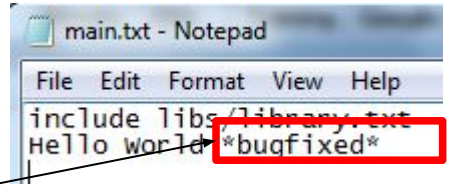
Run „Commit Viewer“ gitk to see the tag in yellow color
~/harry]$ gitk

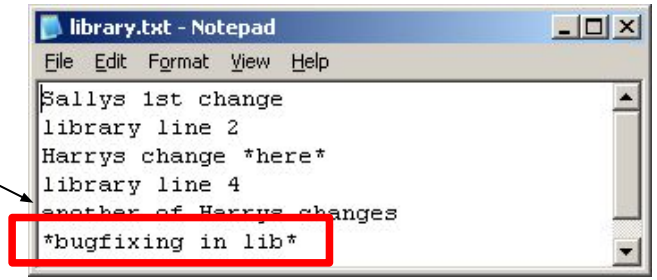
Exercise VI: Branching

Create a branch named „**bugfix_release_01_your-username**“
~/harry]$ git branch bugfix_release_01_<your-username>
All in Harry repo only!

Check out this new branch:
~/harry]$ git checkout bugfix_release_01_<your-username>

Change Harry repo main.txt and libs/library.txt:
Then commit with message „bugfixing“ and push the change
~/harry]$ git commit -a -m "bugfixing“
~/harry]$ git push origin bugfix_release_01_<your-username>





Check out master branch: ~/harry]$ git checkout master

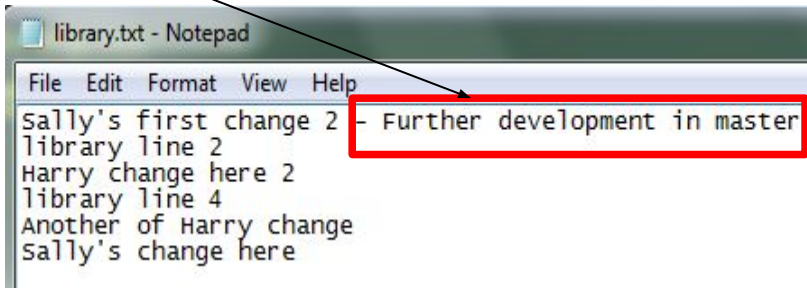
Add a line in Harrys repo library.txt master branch, diff and commit it as „making progress in master“, then check status and push the change

~/harry]$ git diff HEAD or ~/harry]$ git gui

~/harry]$ git commit -a -m "making progress in master“

~/harry]$ git status

~/harry]$ git push



Exercise VII: Merging

Check if you are on master branch in Harry repo

~/harry]$ git status

On branch master

nothing to commit (working directory clean)

Merge bugfix_release_01_you-username into master branch

~/harry]$ git merge bugfix_release_01_<your-username>

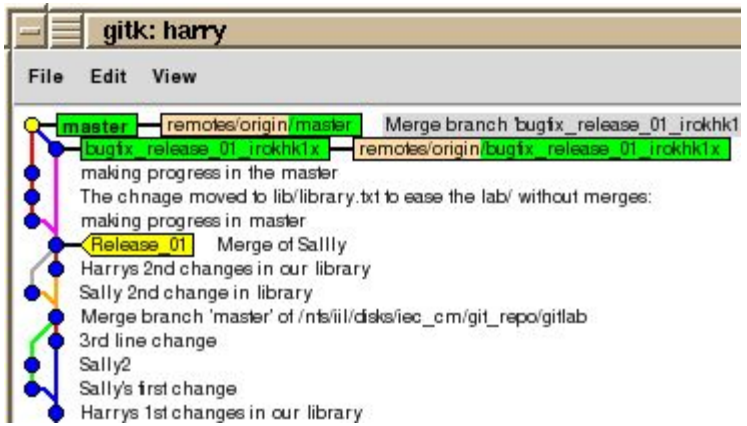
```
commit daef71070a044679e491f8adafb68a34159aa7c1
Merge: 3d11e75 0ac136b
Author: irokhk1x <ilyax.rok
Date:   Mon Jan 6 18:40:39 2014 +0200

    Merge branch 'bugfix_release_01_irokhk1x'

MM      irokhk1x/libs/library.txt
```

Check if the merge add the bugfix into the master branch

~/harry]$ git log -c --name-status

~/harry]$ git diff HEAD^ HEAD

~/harry]$ gitk



Push the merge changes to original repo,
see file annotation

~/harry]$ git push

~/harry]$ git blame ./libs/library.txt

```
3d11e757 irokhk1x/libs/library.txt (irokhk1x 2014-01-06 18:39:16 +0200 1) Sally's first change 2 - Further development in master
c7225035 irokhk1x/library.txt      (irokhk1x 2014-01-02 18:12:33 +0200 2) library line 2
f4c97122 irokhk1x/libs/library.txt (irokhk1x 2014-01-05 11:27:53 +0200 3) Harry change here 2
c7225035 irokhk1x/library.txt      (irokhk1x 2014-01-02 18:12:33 +0200 4) library line 4
f7595725 irokhk1x/libs/library.txt (irokhk1x 2014-01-05 14:32:03 +0200 5) Another of Harry change
75c03db6 irokhk1x/libs/library.txt (irokhk1x 2014-01-05 14:40:29 +0200 6) Sally's change here
0ac136bc irokhk1x/libs/library.txt (irokhk1x 2014-01-06 16:51:32 +0200 7) *bugfixing in lib*
```

Rebase
Exercise VIII

Go to Sally's repository and add line to the END of file library.txt

```
$ ~/sally/lib (master)
$ vi library.txt
```

Commit the change in Sally's repository

```
$ ~/sally/lib (master)
$ git commit -a -m "sally's commit"
[master 4a14da9] sally commit
 1 file changed, 2 insertions(+)
```

Push the change to origin

```
$ ~/sally/lib (master)
$ git push origin
```

Go to Harry's repository

```
$ ~/sally/lib (master)
$ cd ~/harry/lib
```

Add first line to the beginning of the file library.txt

```
$ ~/harry/lib (master)
$ vi library.txt
```

Rebase
Exercise VIII Continue

Commit the change in Harry's repository
$ ~/harry/lib (master)
$ git commit -a -m "harry's change"
[master 70ed6d3] harry's change
 1 file changed, 2 insertions(+)

Fetch the Sally's commit
$ ~/harry/lib (master)
$ git fetch
From c:/Users/Ilya/gitlab.ilyar
   06cfb7a..4a14da9 master   -> origin/master

See the log, where is the Sally's commit and where is Harry's commit, are they on the same line?
$ ~/harry/lib (master)
$ gitk

Now run rebase
$ ~/harry/lib (master)
$ git rebase
First, rewinding head to replay your work on top of it... Applying: harry's change

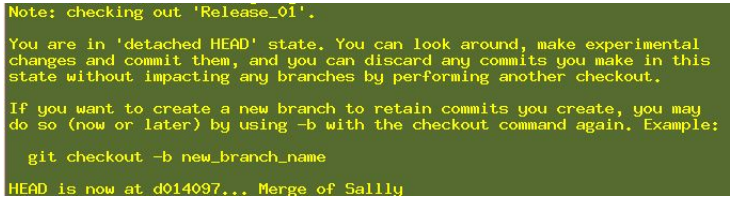
Now see the log, where is the Sally's commit and where is Harry's commit now? Why?
$ ~/harry/lib (master)
$ gitk

Check Out, Branch and Chery Pick old commit

Exercise IX

Check if you are on master branch in Harry's repo
~/harry]$ git status
On branch master
nothing to commit (working directory clean)

Run log to find commit we want to check out (with tags)
~/harry]$ git log --decorate



Check out specific revision, Tag Release_01 points to
~/harry]$ git checkout Release_01 Or ~/harry]$ git checkout d01409754a6d77cdb1e7f2c3a72394fee7ddeb0d

Check out the merge add the bugfix into the master branch
~/harry]$ git checkout -b Release_01_Fix Or ~/harry]$ git branch Release_01_Fix Release_01

Cherry Pick merge commit („making progress in the master") into current branch Release_01_Fix
~/harry]$ git log master --decorate --name-only and copy appropriate commit string
~/harry]$ git cherry-pick 3d11e757f7e6d6843a2f7965c2e901874d13c48f
~/harry]$ git diff HEAD^ HEAD – review check-pick merge results
~/harry]$ gitk --all

Push the merge changes to the original repo
~/harry]$ git push --all

Git Reverts of all kinds
Exercise X

Check if you are on master branch in Harry's repo
~/harry]$ git status
On branch master
nothing to commit (working directory clean)

Edit the file main.txt add a line „# Add temp line" save the file
~/harry]$ vi main.txt

Run git status and see how to discard last file changes
~/harry]$ git status

```
[irokhk1x@itstl005 irokhk1x]$ git status
# On branch master
# Changed but not updated:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   main.txt
#
no changes added to commit (use "git add" and/or "git commit -a")
```

Discard the change in the Working Copy in the file main.txt, remember the discard is unreversable
~/harry]$ git diff main.txt – Always see what you are going to discard first!!!
~/harry]$ git checkout -- main.txt (discard the change)

```
[irokhk1x@itstl005 irokhk1x]$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   main.txt
```

Do the bellow steps again (vi main.txt, git status, git diff main.txt)
~/harry]$ git add main.txt – To add main.txt to the Index
~/harry]$ git reset HEAD main.txt (Unstage the file)
~/harry]$ git checkout -- main.txt (discard the change)

Git Reverts of all kinds
Exercise X Continue

Do the upper steps again (vi main.txt, git status, git diff main.txt, git add main.txt)
~/harry]$ git commit -a -m "Adding temp line number 3 commit"

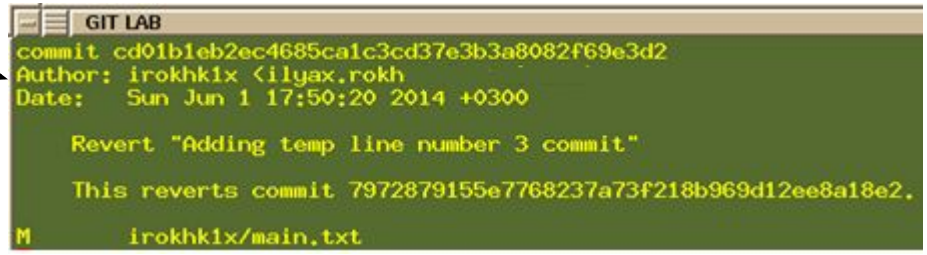
Do the upper steps again and add 1st line "# Header line" to main.txt
~/harry]$ git commit -a -m "Adding Header Line number 1 commit"

Run log to see the two latest commits
~/harry]$ git log --name-status

Run git revert to disable influence of the
„Adding temp line number 3 commit" to the file main.txt

~/harry]$ git revert 7972879155e7768237a73f218b969d12ee8a18e2
Save the comment and close the comment window

Run diff of the file main.txt to see what,
we get after revert
~/harry]$ git diff HEAD^ HEAD



Run log now to see the revert commit
~/harry]$ git log --name-status

Git Reverts of all kinds
Exercise X Continue

Push the the revert changes to the original repo
~/harry]$ git push

Now let us revert already pushed to origin commit, never mind who did it. Let us revert commit „Adding Header Line number 1 commit" ..6a6de
~/harry]$ git revert 2f6c172e05dc158f85e21b61c5ed1dfab586a6de

Run diff of the file main.txt to see what we get after revert
~/harry]$ git diff HEAD^ HEAD – you should see 1 line less

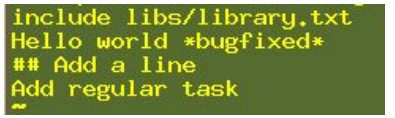
Push the origin commit revert to the original repo
~/harry]$ git push

Git Stash usage
Exercise XI

When you are in the middle of something, your boss comes in and demands that you fix something immediately. Stash you work, do emergency fix, commit it and pop you stash back

Do regular task in master branch, change file main.txt add line „Add regular Task“

```
~/harry]$ vi main.txt
~/harry]$ git stash
Saved working directory and index state WIP on master: ca3568f Revert "Adding second commit"
HEAD is now at ca3568f Revert "Adding second commit“
```



Do emergency fix in master branch, change file main.txt add first line „Add quick fix immediately “

```
~/harry]$ vi main.txt
```


```
~/harry]$ git commit -a -m "Fix in a hurry"

~/harry]$ git stash pop
Auto-merging ilyar/main.txt
# On branch master
Dropped refs/stash@{0} (bd59435b9a1e08a702f8416f642c4d207d72e020)
```

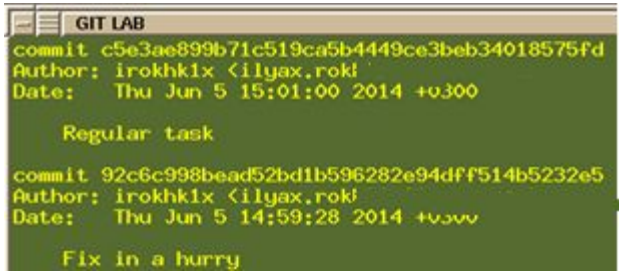
Now continue you work, diff changes, commit and push

```
~/harry]$ git diff

~/harry]$ git commit -a -m "Regular task"
~/harry]$ git push
```

Advanced working with branches
Exercise XII

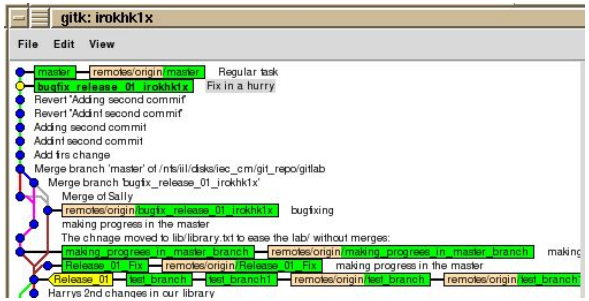
How to take changes that happened on the master in the meanwhile

Let us check out bugfix_release_01_<your-username> branch
~/harry]$ git checkout bugfix_release_01_<your-username>

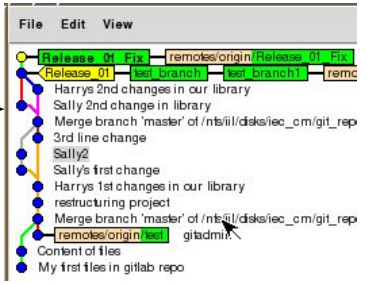


Run git log --all (Branches) and copy „Fix in a hurry" commit
~/harry]$ git log –all
Now let us merge our branch bugfix_release_01_<your-username> with commit HEAD^ on master branch 92c6c998bead52bd1b596282e94dff514b5232e5

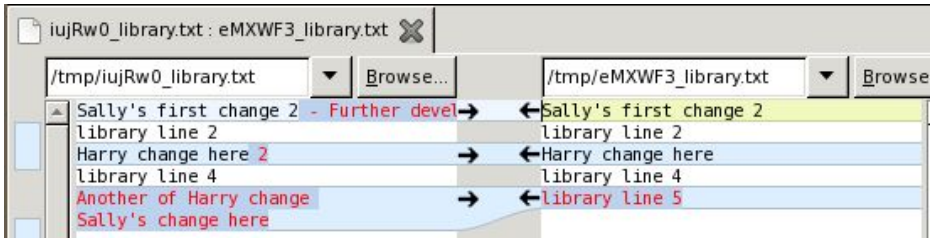
~/harry]$git merge 92c6c998bead52bd1b596282e94dff514b5232e5
Updating 0ac136b..92c6c99
Fast-forward
ilyar/libs/library.txt | 2 +-
ilyar/main.txt | 3 ++-
2 files changed, 3 insertions(+), 2 deletions(-)



Compare my branch to the master (as it was when I last updated from master)
~/harry]$ git checkout Release_01_Fix
Run Git log to see when last time was merged from master?
~/harry]$ gitk &



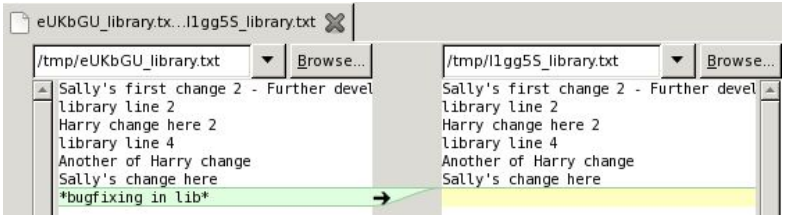
Advanced working with branches
Exercise XII Continue



~/harry]$ git diff HEAD bccdb778dc129c990a0c09147c914a34eff12dda

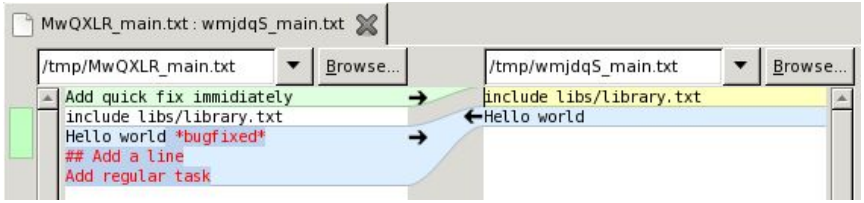
How to Compare two branches

Let us diff master branch with Release_01_Fix branch
~/harry]$ git diff master Release_01_Fix



~/harry]$ git diff --no-ext-diff master Release_01_Fix – without external tool

```
diff --git a/ilyar/libs/library.txt b/ilyar/libs/library.txt
index 59a920d..0ebca13 100644
--- a/ilyar/libs/library.txt
+++ b/ilyar/libs/library.txt
@@ -4,4 +4,3 @@ Harry change here 2
 library line 4
 Another of Harry change
 Sally's change here
-*bugfixing in lib*
diff --git a/ilyar/main.txt b/ilyar/main.txt
index 75bb070..1e22422 100644
--- a/ilyar/main.txt
+++ b/ilyar/main.txt
@@ -1,5 +1,2 @@
-Add quick fix immidiately
 include libs/library.txt
-Hello world *bugfixed*
-## Add a line
-Add regular task
+Hello world
```



Using the git log
Exercise XIII

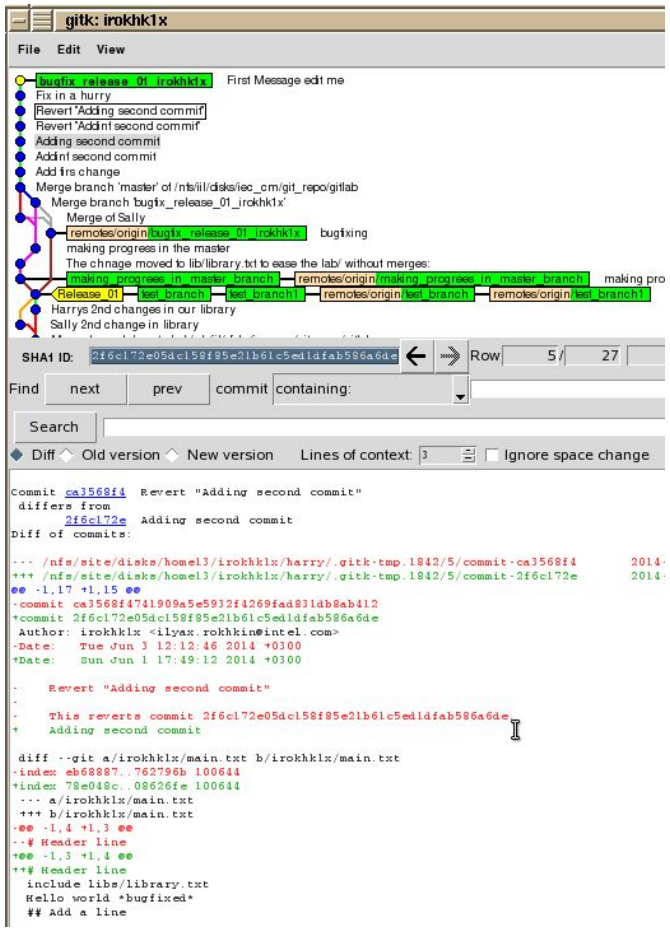
Git log in cmd and GUI
This is the log command in cmd to see merge info + files, that were changed + all branches
~/harry]$ git log -c --name-status --all

~/harry]$ gitk --all & to see gui log of all branches and

~/harry]$ gitk & to see log of only current branch

Write click on commit and select “Mark this commit”
Now go to another commit, write click on it and
select “Compare with marked commit”
Check out bugfix_release_01_<yourusername> branch

~/harry]$ git checkout bugfix_release_01_<yourusername>
Check that you have commits not yet pushed,
only such commits Can be changed!!!



Using the git log
Exercise XIII Continue

```
~/harry]$ git push origin --dry-run
   0ac136b..b597fd2  bugfix_release_01_ilyar -> bugfix_release_01_ilyar
```

Change commit message to “We can edit unpushed messages only”

```
~/harry]$ git commit --amend -m "We can edit unpushed messages only"
[bugfix_release_01_ilyar d4e3205] We can edit unpushed messages only
 1 files changed, 1 insertions(+), 0 deletions(-)
```

```
~/harry]$ git log
commit d4e3205f7c80db7d4bda4ac588b5fcfd73f9ee31
Author: ilyar <astra07_2010@yahoo.com>
Date:   Sun Jun 8 15:30:33 2014 +0300
    We can edit unpushed messages only
```

Now push latest changes to the origin repo

```
~/harry]$ git push
```

Format/Apply Patch
Exercise XIV

Go to Harry's repository and add line to the END of file library.txt
$ ~/harry/lib (master)
$ vi library.txt

Commit the change in Harry's repository
$ ~/harry/lib (master)
$ git commit -a -m "harry's commit for patch"

Format patch of the last commit
$ ~/harry/lib (master)
$ git format-patch -1 7b82010d86bcc9f020c413e216a1caf3d0651147
0001-harry-s-patch-change.patch

See the patch file on file system
$ ~/harry/lib (master)
$ ls
0001-harry-s-patch-change.patch library.txt

Go to Sally's repository
$ ~/harry/lib (master)
$ cd ~/sally/lib

Format/Apply Patch
Exercise XIV Continue

See what you got in the patch
$ ~/sally/lib (master)
$ git apply --stat ~/harry/lib/0001-harry-s-patch-change.patch
 library.txt | 2 ++
 1 file changed, 2 insertions(+)

Check if the patch is applicable, “no output no errors”
$ ~/sally/lib (master)
$ git apply --check ~/harry/lib/0001-harry-s-patch-change.patch

Apply the patch with signature of the patch
$ ~/sally/lib (master)
$ git am --signoff < ~/harry/lib/0001-harry-s-patch-change.patch
Applying: harry's patch change

$ ~/sally/lib (master)
$ git log -1
commit 38a8ce3ce4cd260303ba930ab1a3c2b49e3dd217
Author: Ilya <astra07_2010@yahoo.com>
Date: Fri May 29 18:33:10 2015 +0300

    harry's patch change

    Signed-off-by: Ilya <astra07_2010@yahoo.com

Squash several commits to 1 commit
Exercise XV

In Sally's repository do 2 changes and 2 commits

We want to squash 2 commits 9fb9d06 and 5822495
$ ~/sally/lib (master)
$ git log --pretty=oneline -3
9fb9d060ab4ffe34f033d7b61163ce6697894a7b Add 2nd commit for squash
5822495ba795f4b0a18e89d52e77473dcd700c65 Add alt commit line
38a8ce3ce4cd260303ba930ab1a3c2b49e3dd217 harry's patch change

Run git rebase interactively on commit **before** 2 commits you want to squash
$ ~/sally (master)
$ git rebase -i 38a8ce3ce4cd260303ba930ab1a3c2b49e3dd217
The first commit **pick** (leave it)

```
pick 5822495 This is parent commit
f 9fb9d06 This is child commit
# Rebase 38a8ce3..9fb9d06 onto 38a8ce3
```

[detached HEAD e33ede3] Add alt commit line
 2 files changed, 3 insertions(+)
Successfully rebased and updated refs/heads/master.

Run now git log --pretty=oneline -2 You should see only 1 commit instead of 2 that was before

Push directly to another repo not origin, From sally to harry directly. Exercise XVI

Pull then push in Sally's repository and pull again to be fully synced with Harry

```
$ ~/sally/lib (master)
$ git log --pretty=oneline -3
93dffff1a6160c747d18c5fd36d0cf82062042a2 harry's change
e33ede3a4a6428f523d8bec1041098464cad58b7 Add alt commit line
930ab1a3c2b49e3dd217 harry's patch change
```

Pull then push in Harry's repository and Pull again to be fully synced with Sally

```
$ ~/harry/lib (master)
$ git log --pretty=oneline -3
93dffff1a6160c747d18c5fd36d0cf82062042a2 harry's change
e33ede3a4a6428f523d8bec1041098464cad58b7 Add alt commit line
930ab1a3c2b49e3dd217 harry's patch change
```

Change diff commit sally's file

```
$ ~/sally/lib (master)
$ vi library.txt

$ git diff HEAD

$ ~/sally (master)
$ git commit -a -m "Push for harry"
```

Push directly to another repo not origin, From sally to harry directly. Exercise XVI continue

$ ~/sally/lib (master)
$ git push ~/harry master:master

Worked? Why not?

Go to Harry's repository
$ ~/sally/lib (master)
$ cd ~/harry/lib

Check another branch:
~/harry]$ git checkout bugfix_release_01_<your-username>

Back to Sally's repository $ cd ~/sally

$ ~/sally (master)
$ git push ~/harry master:master
To c:/Users/Ilya/harry 93dffff..fde4fd3 master -> master

$ ~/harry (bugfix_release_01_<your-username>)
$ git checkout master

$ ~/harry (master)
$ git log --pretty=oneline -1
fde4fd370a20dd1491e5f5edd8498f1cc833912d Push for harry