

# The Working Cycle

## Exercise I: adding files

First say to Git who are you, this info will appear on each commit you do. Example:

```
$ git config --global user.name <YourUSERNAME>
```

```
$ git config --global user.email <YourEMAIL>
```

Please Create new (git init) „bare“ repository to be your origin(remote) repository

```
$ git init --bare --shared ~/gitlab
```

Then clone it to harry repo: cd .git and overview repository objects

```
$ git clone ~/gitlab ~/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

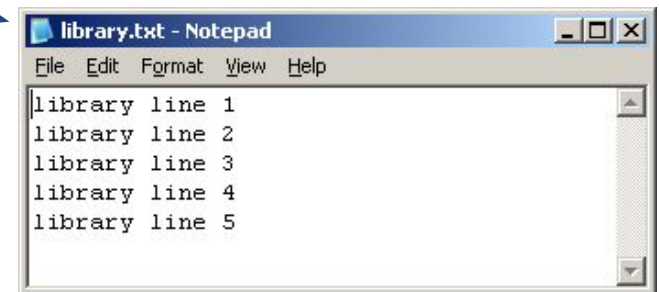
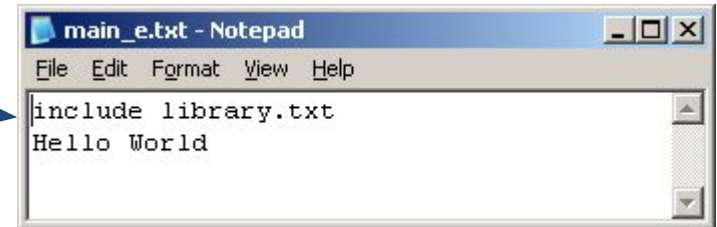
```
$ git add *
```

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 and diff before commit

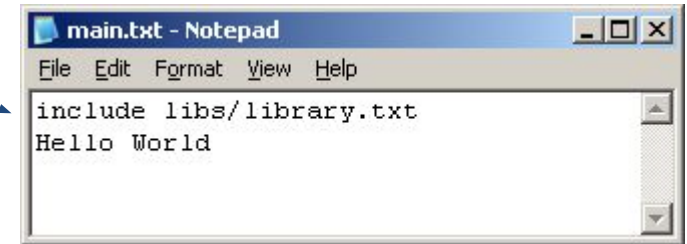
```
$ git status $ git diff --staged main.txt
```

Commit your changes as "Restructuring project"

```
$ git commit -m "Restructuring project"
```

Push all your work to origin (remote) repository

```
$ git push
```



```
# 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
#
```

```
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
```

# The Working Cycle

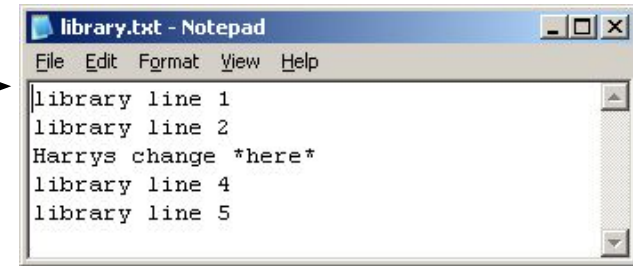
## Exercise III: teamwork

Change the third line in Harry's repo and check the status

```
$ git status $ git diff HEAD
```

Commit Harry's changes as "Harry's 1st changes in our library"

```
$ git commit -a -m "Harry's 1st changes in our library"
```



Clone the gitlab repo again (cd .. and into a folder named „sally“)

```
$ git clone ~/gitlab ~/sally
```

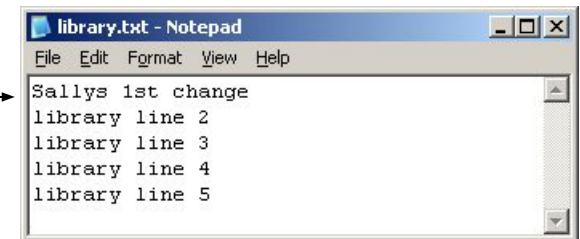
Push all your work in Harry's repo to origin (central) repo

```
$ git push
```

Change the first line in Sally's repo commit it and try to push

```
$ git diff $ git commit -a -m "Sally's 1st changes in our library"
```

```
$ git push
```



Pull updates to Sally's repo:

```
$ git pull --rebase
```

Overview what you got:

```
$ gitk
```

Compare what changes you got in „Commit Viewer“:

```
$ git diff HEAD^ HEAD
```

– compare HEAD commit with previous commit, do you see both Sally and Harry hange?

Push Sally's changes: 

```
$ git push
```

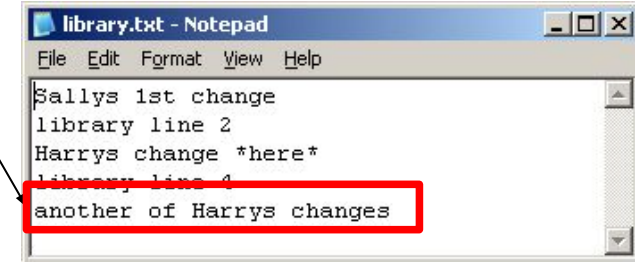
# The Working Cycle

## Exercise IV: conflicts (with rebase)

Pull Harry's repo to get Sally's last changes: ~/harry] \$ **git pull**

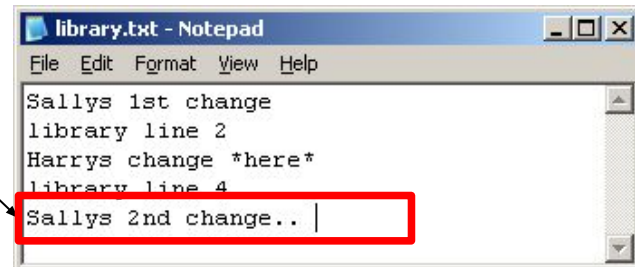
Change the fifth (last) line in Harry's repo, see your change:  
\$**git diff HEAD**

Commit Harry's changes as „Harry's 2nd changes in our library“  
~/harry]\$ **git commit -a -m "Harry's 2nd changes in our library"**



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

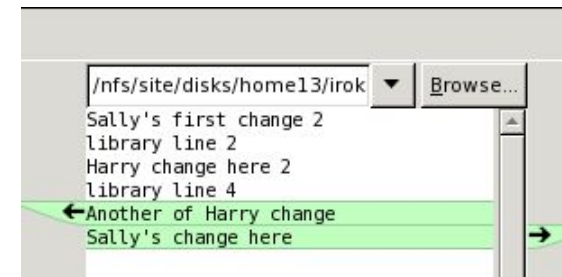
Change the fifth (last) in Sally's repo commit and try to push  
~/sally]\$ **git commit -a -m "Sally 2nd change in library"**  
~/sally]\$ **git push**      Why do you think it failed?



Pull Sally's repo: ~/sally]\$ **git pull --rebase**

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

Save the file library.txt after the merge conflicts were solved



# The Working Cycle

## Exercise IV: conflicts (with rebase) (Continue)

Run "git add library.txt" to mark resolution and add the file to staging area

```
~/sally]$ git add library.txt
```

Run git status to see what is next?

```
~/sally]$ git status
```

rebase in progress; onto fe343dc. You are currently rebasing branch 'master' on 'fe343dc'.  
(all conflicts fixed: run "git rebase --continue")

Run git rebase --continue to continue and finish the rebase

```
~/sally]$ git rebase --continue
```

Applying: Sally 2nd change in library

Recorded resolution for 'library.txt'

Run git status to see what is next?

```
~/sally]$ git status
```

Your branch is ahead of 'origin/master' by 1 commit.

(use "git push" to publish your local commits)

Run git log with diff to see results of rebase and conflict resolution

```
~/sally]$ git log -c -2
```

Now we can push Sally's changes after rebase

```
~/sally]$ git push
```

# Exercise V Tagging

Create a tag named „Release\_01“  
with commit message "tag Release\_01"

```
~/sally]$ git tag Release_01 -m "Official Release_01 GA"
```

Run git log to see the Tag in yellow color

```
~/sally]$ git log --graph --oneline --all
```

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
```

Fetch with the tag from the harry repository

```
~/harry]$ git fetch --tags
```

Run git log to see the tag in yellow color

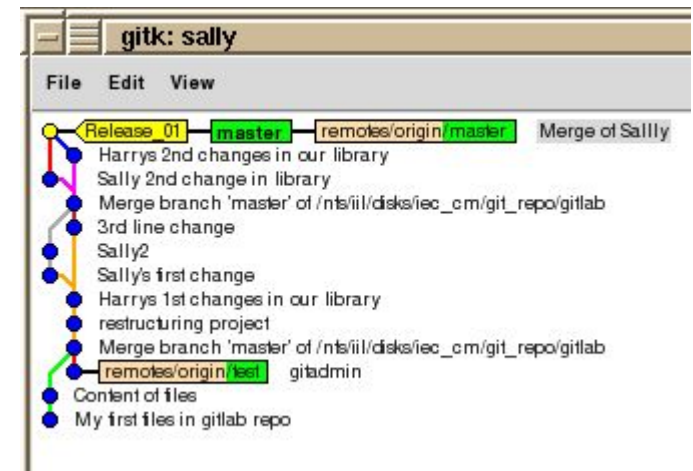
```
~/harry]$ git log --graph --oneline --all
```

Rebase to align local master branch with origin/master

```
~/harry]$ git rebase
```

First, rewinding head to replay your work on top of it...

Fast-forwarded master to refs/remotes/origin/master.



# Exercise VI: Branching

Create a branch named „bugfix/release\_01\_fix1“

~/harry]\$ **git branch** ` All in Harry repo only!

Check out this new branch:

~/harry]\$ **git checkout bugfix/release\_01\_fix1**

Change Harry's repo main.txt and libs/library.txt:

Then commit with message „bugfixing“ and push

~/harry]\$ **git commit -a -m "bugfixing"**

~/harry]\$ **git push --set-upstream origin bugfix/release\_01\_fix1**

Check if remote tracking branch was set, for bugfix/release\_01\_fix1

~/harry]\$ **git branch -vv**

\* bugfix/release\_01\_fix1 afae908 [origin/bugfix/release\_01\_fix1]

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

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

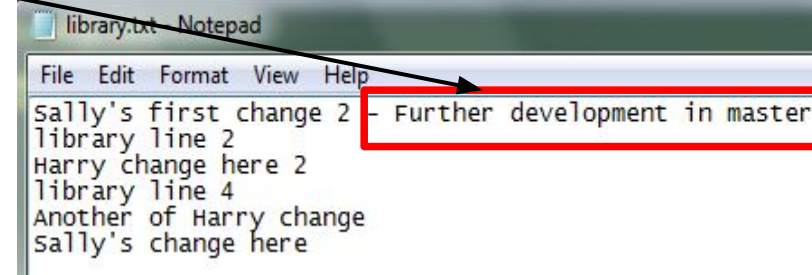
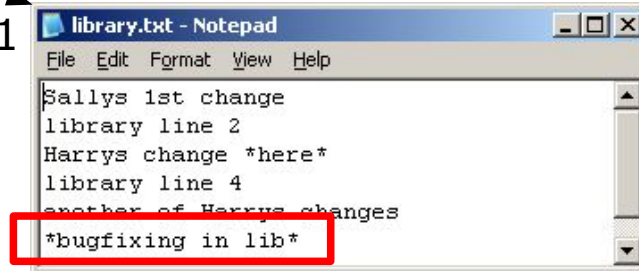
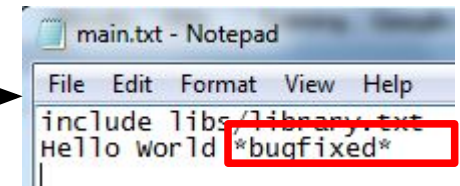
~/harry]\$ **git diff HEAD** or ~/harry]\$ **git gui**

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

See new branch in log graph

~/harry]\$ **git log --graph --oneline --branches**

~/harry]\$ **git status** and ~/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\_fix1 into master branch

```
~/harry]$ git merge bugfix/release_01_fix1
```

Check if the merge add the bugfix into the master branch

```
~/harry]$ git log -c
```

```
~/harry]$ git diff HEAD^ HEAD
```

```
~/harry]$ gitk
```

Push the merge changes to original repo,  
see file annotation

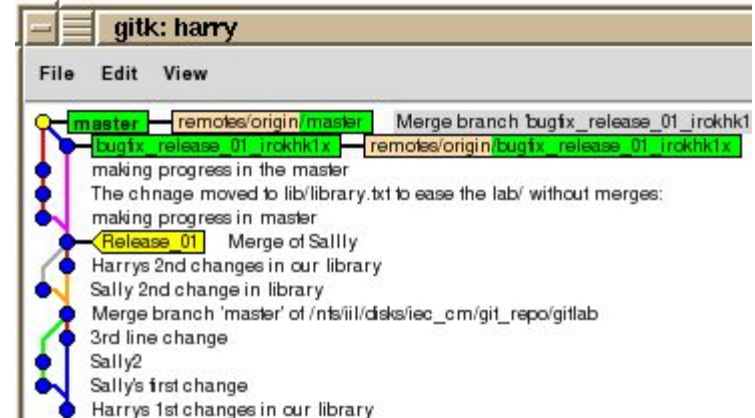
```
~/harry]$ git push
```

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

```
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
```



```
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*
```



# Check Out, Branch and Cherry Pick old commit

## Exercise VIII

Check if you are on master branch in Harry's repo

```
~/harry]$ git status
```

```
# On branch master
```

```
nothing to commit (working directory clean)
```

```
commit d01409754a6d77cdb1e7f2c3a72394fee7ddeb0d (tag: Release_01)
Merge: 75c03db ^59572
Author: irokhk'~ /' ----
Date:   Sun Jan 5 15:20:13 2014 +0200

Merge of Sally
```

Run log to find commit we want to check out (with tags)

```
~/harry]$ git log --decorate
```

```
Note: checking out 'Release_01'.
```

```
You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.
```

```
If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:
```

```
git checkout -b new_branch_name
```

```
HEAD is now at d014097... Merge of Sally
```

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 bugfix/release_01_fix2 Or ~/harry]$ git branch bugfix/release_01_fix2 Release_01
```

Cherry Pick commit („making progress in the master”) into current branch bugfix/release\_01\_fix2

```
~/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 cherry-pick changes to the original repo

```
~/harry]$ git push --set-upstream origin
```

# Git Reverts of all kinds

## Exercise IX

Check if you are on master branch in Harry's repo

```
~/harry]$ git status
```

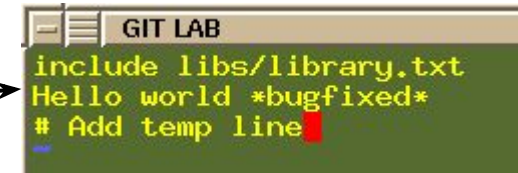
```
# On branch master
```

```
nothing to commit (working directory clean)
```

### How to undo modify

Edit the file main.txt add a line „# Add temp line“ save the file

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



```
GIT LAB
include libs/library.txt
Hello world *bugfixed*
# Add temp line
```

Run git status and see how to discard last file changes

```
~/harry]$ git status
```

Discard the change in the Working Copy in the file main.txt, `git diff HEAD main.txt` – Always see what you are going to discard first!!!

```
~/harry]$ git checkout -- main.txt (discard the change)
```

Run git status and see status of the file now

```
~/harry]$ git status
```

### How to undo staged changes

Do the below steps again (`vi main.txt`, `git status`, `git diff HEAD main.txt`)

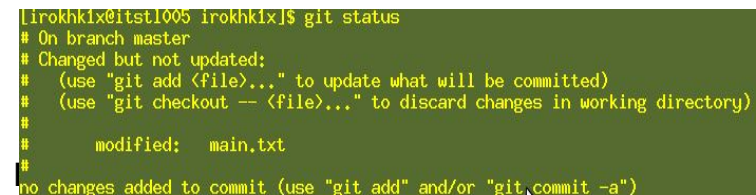
```
~/harry]$ git add main.txt (To add main.txt to the Index)
```

```
~/harry]$ git diff --staged (Compare HEAD and staged, see what you are going to discard first!!!)
```

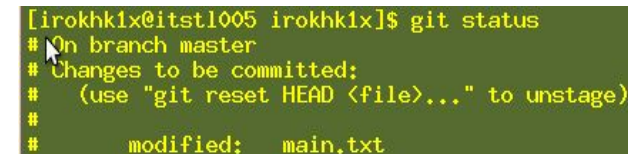
```
~/harry]$ git reset HEAD main.txt (Unstage the file)
```

```
~/harry]$ git diff
```

```
~/harry]$ git checkout -- main.txt (discard the change)
```



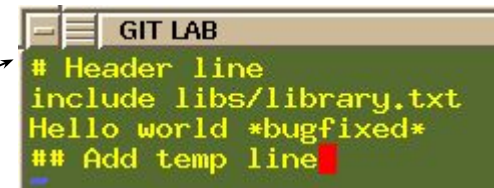
```
[irokhk1x@itst1005 irokhk1x]$ git status
# On branch master
#
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   main.txt
#
no changes added to commit (use "git add" and/or "git commit -a")
```



```
[irokhk1x@itst1005 irokhk1x]$ git status
# On branch master
#
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   main.txt
```

# Git Reverts of all kinds

## Exercise IX Continue



```
# Header line
include libs/library.txt
Hello world *bugfixed*
## Add temp line
```

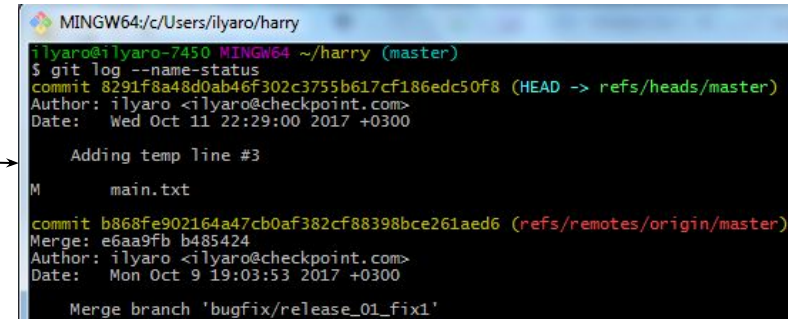
### How to revert latest local commit

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"`

Run log to see the two latest commits

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



```

ilyaro@ilyaro-7450 MINGW64 ~/harry (master)
$ git log --name-status
commit 8291f8a48d0ab46f302c3755b617cf186edc50f8 (HEAD -> refs/heads/master)
Author: ilyaro <ilyaro@checkpoint.com>
Date:   Wed Oct 11 22:29:00 2017 +0300

    Adding temp line #3

M       main.txt

commit b868fe902164a47cb0af382cf88398bce261aed6 (refs/remotes/origin/master)
Merge:  e6aa9fb b485424
Author: ilyaro <ilyaro@checkpoint.com>
Date:   Mon Oct 9 19:03:53 2017 +0300

    Merge branch 'bugfix/release_01_fix1'
```

Run git reset to reset current branch to previous commit

`~/harry]$ git reset --hard HEAD^`

(Be carefull `--hard` resets Index, Working Tree also)

Run log now to see the previous good commit back to be first

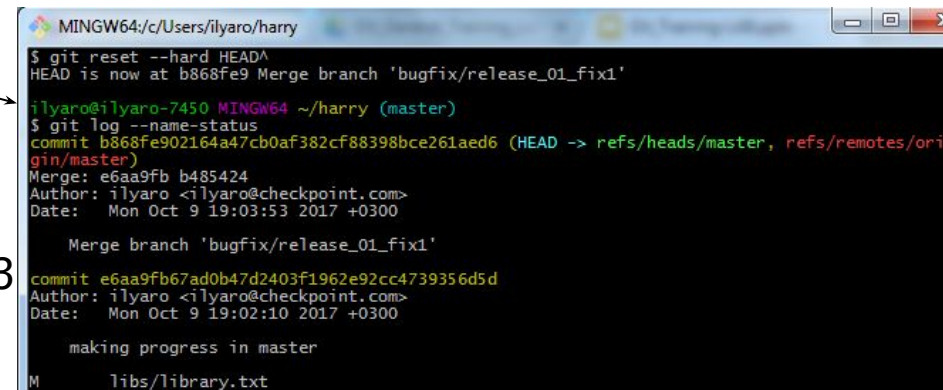
`~/harry]$ git log --name-status`

Where is the temp commit? It exists, just not Referenced, run reflog to see it:

`~/harry]$ git reflog`

`$ git reflog`

`8291f8a HEAD@{1}: commit: Adding temp line #3`



```

MINGW64/c/Users/ilyaro/harry
$ git reset --hard HEAD^
HEAD is now at b868fe9 Merge branch 'bugfix/release_01_fix1'

ilyaro@ilyaro-7450 MINGW64 ~/harry (master)
$ git log --name-status
commit b868fe902164a47cb0af382cf88398bce261aed6 (HEAD -> refs/heads/master, refs/remotes/origin/master)
Merge:  e6aa9fb b485424
Author: ilyaro <ilyaro@checkpoint.com>
Date:   Mon Oct 9 19:03:53 2017 +0300

    Merge branch 'bugfix/release_01_fix1'

commit e6aa9fb67ad0b47d2403f1962e92cc4739356d5d
Author: ilyaro <ilyaro@checkpoint.com>
Date:   Mon Oct 9 19:02:10 2017 +0300

    making progress in master

M       libs/library.txt
```

# Git Reverts of all kinds

## Exercise IX Continue

### How to Revert Remote Commit

Let us reset master branch back to "Adding temp line number 3 commit"

```
~/harry]$ git reset --hard 8291f8a
```

Push the commit to the original repo

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

See the log and copy commit sha1

```
~/harry]$ git log -2
```

Now let us revert already pushed to origin commit, never mind who did it.

Let us revert commit "Adding temp line number 3 commit"

```
~/harry]$ git revert 8291f8a48d0ab46f302c3755b617cf186edc50f8
```

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
```

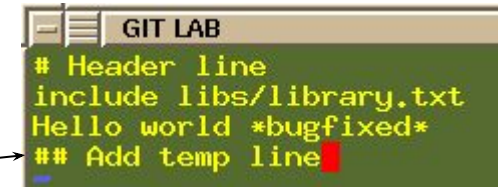
```
~/harry]$ git log -2
```

Run git status and see revert commit

```
~/harry]$ git status
```

Push the origin commit revert to the original repo

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



```
GIT LAB
# Header line
include libs/library.txt
Hello world *bugfixed*
## Add temp line
```

# Git Stash usage

## Exercise X

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: 581fe85 Revert "Adding temp line #3"

HEAD is now at 581fe85 Revert "Adding temp line #3"

```
include libs/library.txt
Hello world *bugfixed*
## Add a line
Add regular task
~
```

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 a bug in a quick"
```

```
GIT LAB
Add quick fix immediately
include libs/library.txt
Hello world *bugfixed*
## Add a line
```

```
~/harry]$ git stash pop
```

Auto-merging ilyar/main.txt

# On branch master

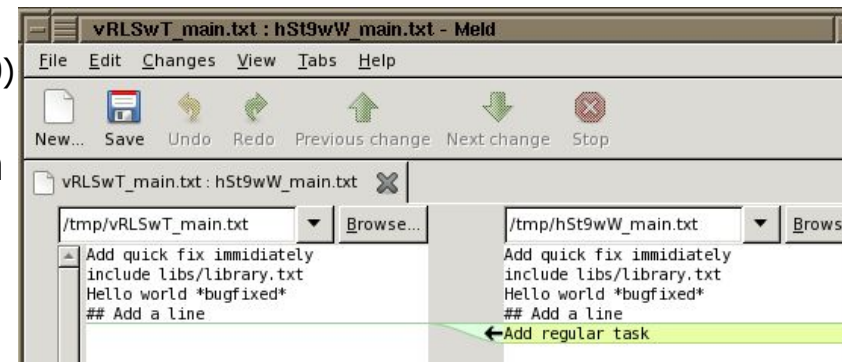
Dropped refs/stash@{0} (bd59435b9a1e08a702f8416f642c4d207d72e020)

Now continue you work, diff changes, commit and push

```
~/harry]$ git diff
```

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

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



# Format/Apply Patch

## Exercise XI

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

~/harry/libs (master)

```
$ vi library.txt
```

Commit the change in Harry's repository

~/harry/libs (master)

```
$ git commit -am "harry's commit for patch"
```

Format patch of the last commit

~/harry/libs (master)

```
$ git format-patch -1 7b82010d86bcc9f020c413e216a1caf3d0651147
```

0001-harry-s-patch-change.patch

See the patch file on file system

~/harry/libs (master)

```
$ ls  
0001-harry-s-patch-change.patch library.txt
```

Go to Sally's repository and pull latest commits

~/harry/libs (master)

```
$ cd ~/sally/libs
```

~/sally/libs (master)

```
$ git pull
```



# Format/Apply Patch

## Exercise XI Continue

See what you got in the patch

```
$ ~/sally/libs (master)
$ git apply --stat ~/harry/libs/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/libs (master)
$ git apply --check ~/harry/libs/0001-harry-s-patch-change.patch
```

See content of the patch file, looks familiar?

```
$ ~/sally/libs (master)
$ cat ~/harry/libs/0001-harry-s-patch-change.patch
```

Apply the patch with signature of the patch

```
$ ~/sally/libs (master)
$ git am --signoff < ~/harry/libs/0001-harry-s-patch-change.patch
Applying: harry's patch change
```

```
$ ~/sally/libs (master)
$ git log -1 -c
commit 38a8ce3ce4cd260303ba930ab1a3c2b49e3dd217
    harry's patch change
Signed-off-by: Ilya <astra07_2010@yahoo.com>
```

# Squash several commits to 1 commit

## Exercise XII

In Sally's repository do 2 changes and 2 commits

We want to squash 2 commits 9fb9d06 and 5822495

```
$ ~/sally/libs (master)
```

```
$ git log --pretty=oneline -3
```

```
9fb9d060ab4ffe34f033d7b61163ce6697894a7b Add 2nd commit for squash
```

```
5822495ba795f4b0a18e89d52e77473dcd700c65 Add alt commit line
```

```
38a8ce3ce4cd260303ba930ab1a3c2b49e3dd217 harry's commit for patch
```

Run git rebase interactively on commit **before** 2 commits you want to squash

```
$ ~/sally (master)
```

```
$ git rebase -i HEAD^^ OR $ git rebase -i 38a8ce3ce4cd260303ba930ab1a3c2b49e3dd217
```

The first commit **pick** (leave it)

The second squash into first commit

```
pick 5822495 This is parent commit
s 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 -c -2` You should see only 1 commit instead of 2 that was before

What happened with the 2 commits being squashed? Disappeared? Let us find them

```
$ git reflog
```

# Pull directly from another repository not origin.

## Exercise XIII

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

~/sally/libs (master)

```
$ git log --pretty=oneline -1
```

```
fde4fd370a20dd1491e5f5edd8498f1cc833912d Push for harry
```

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

Let's see harry's log from within sally's repository

~/**sally**/libs (master)

```
$ git -C ~/harry log master --pretty=oneline -1
```

```
fde4fd370a20dd1491e5f5edd8498f1cc833912d Add alt commit line
```

Change sally's file and commit it

~/sally/libs (master)

```
$ vi library.txt  $ git diff  $ git commit -am "Commit to pull for harry"
```

Go to harry's repository: \$ `cd ~/harry` and pull the commit

~/harry (master)

```
$ git pull ~/sally master:master
```

Let us see the logs of the 2 repositories again

~/harry (master)

```
$ git log master --pretty=oneline -1
```

```
c98ab2d24894ee330a5c75f1c7fc77d83a7061db (HEAD -> master) Commit to pull for harry
```

```
$ git -C ~/sally log master --pretty=oneline -1
```

```
c98ab2d24894ee330a5c75f1c7fc77d83a7061db (HEAD -> master) Commit to pull for harry
```

# Backup slides

# Rebase (Already present in TeamWork)

## Exercise VIII

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

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

Commit the change in Sally's repository

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

Push the change to origin

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

Go to Harry's repository

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

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

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

# Rebase (Already present in TeamWork)

## Exercise VIII Continue

Commit the change in Harry's repository

```
$ ~/harry/libs (master)
```

```
$ git commit -a -m "harry's change"
```

```
[master 70ed6d3] harry's change
```

```
1 file changed, 2 insertions(+)
```

Fetch the Sally's commit

```
$ ~/harry/libs (master)
```

```
$ git fetch
```

```
From c:/Users/Ilya/gitlab
```

```
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/libs (master)
```

```
$ git log --graph --oneline --branches
```

Now run rebase

```
$ ~/harry/libs (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/libs (master)
```

```
$ gitk
```



# 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/libs (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

Let's see harry's log from within sally's repository

```
$ ~/sally/libs (master)
```

```
$ git -C ~/harry log master --pretty=oneline -3
```

```
93dffff1a6160c747d18c5fd36d0cf82062042a2 harry's change
```

```
e33ede3a4a6428f523d8bec1041098464cad58b7 Add alt commit line
```

```
930ab1a3c2b49e3dd217 harry's patch change
```

Change sally's file

```
$ ~/sally/libs (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/libs (master)  
$ git push ~/harry master:master
```

Worked? Why not? Go to Harry's repository

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

Check another branch:

```
~/harry]$ git checkout bugfix/release_01_fix1
```

Back to Sally's repository `$ cd ~/sally`

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

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

# Advanced working with branches

## Exercise XI

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

Let us check out bugfix/release\_01\_fix1 branch  
~/harry]\$ **git checkout bugfix/release\_01\_fix1**

Run git log --all (refs) and copy „Fix in a hurry“ commit sha1  
~/harry]\$ **git log --all**

Now let us merge our branch **bugfix/release\_01\_fix1** with commit HEAD^ on **master** branch  
92c6c998bead52bd1b596282e94dff514b5232e5

~/harry]\$ **git merge 92c6c998bead52bd1b596282e94dff514b5232e5**

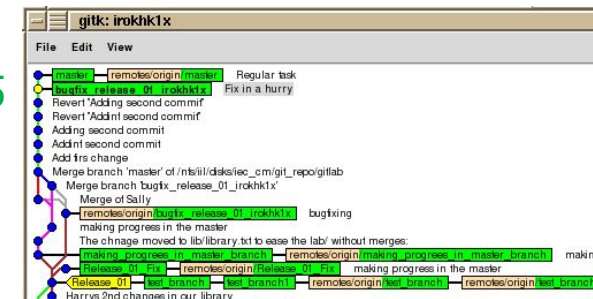
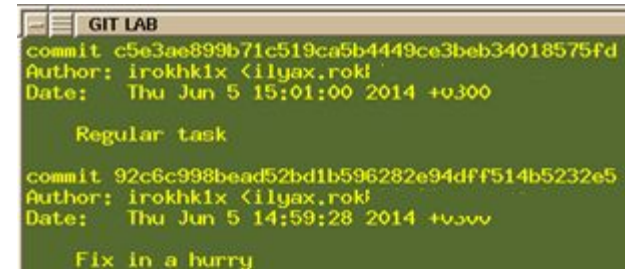
Updating 0ac136b..92c6c99

Fast-forward

ilyar/libs/library.txt | 2 ++

ilyar/main.txt | 3 +++

2 files changed, 3 insertions(+), 2 deletions(-)

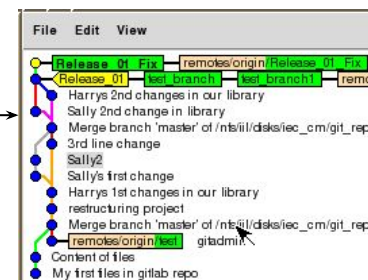


Compare my branch to the master (as it was when I last updated from master)

~/harry]\$ **git checkout bugfix/release\_01\_fix1**

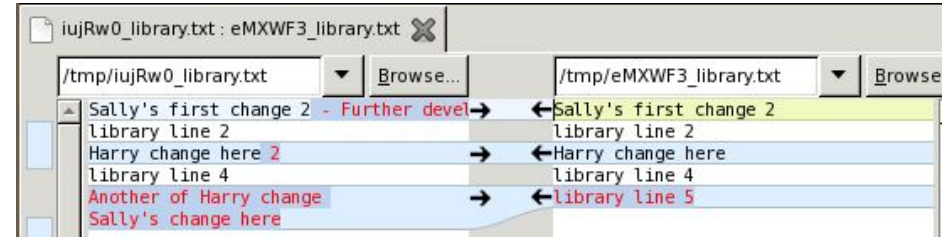
Run Git log to see when last time was merged from master?

~/harry]\$ **gitk &**



# Advanced working with branches

## Exercise XI Continue

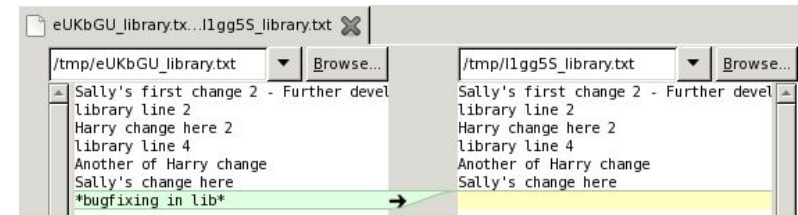


```
~/harry]$ git diff HEAD bccdb778dc129c990a0c09147c914a34eff12dda
```

How to Compare two branches

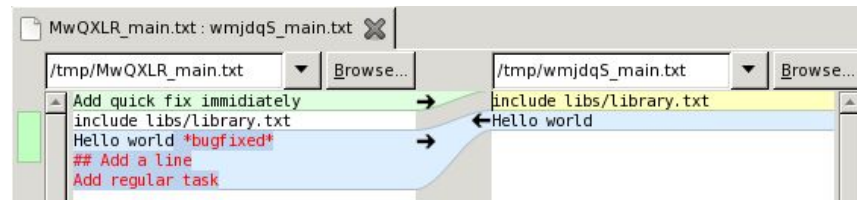
Let us diff master and bugfix\_release\_01\_fix1 branches

```
~/harry]$ git difftool master bugfix/release_01_fix1
```



```
~/harry]$ git diff master bugfix/release_01_fix1 --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 immediately
include libs/library.txt
-Hello world *bugfixed*
-## Add a line
-Add regular task
+Hello world
```



# Using the git log

## Exercise XII

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 --branches
```

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

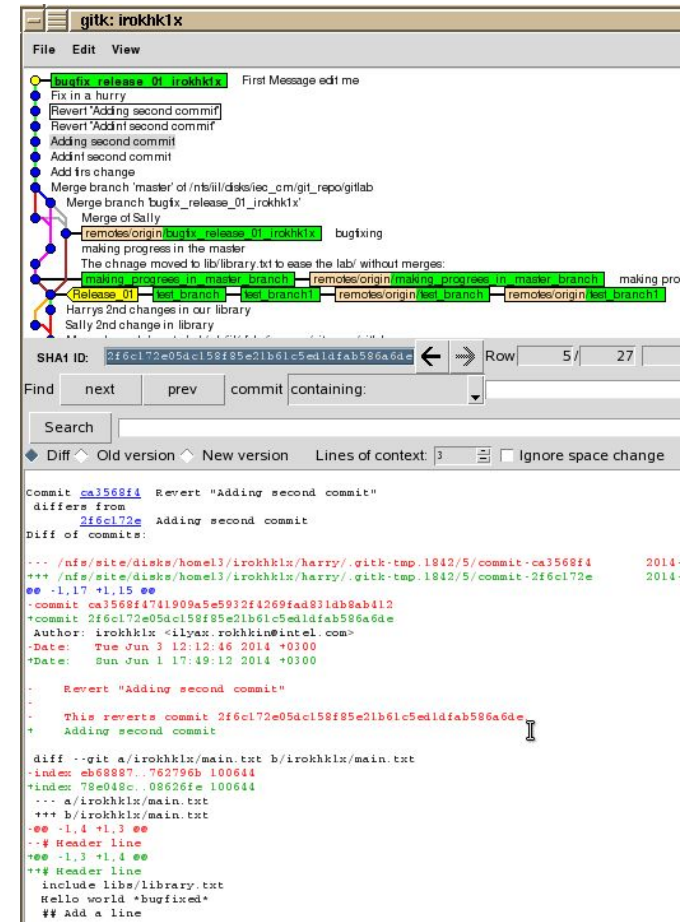
```
~/harry]$ gitk & to see log of only current branch
```

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\_fix1 branch

```
~/harry]$ git checkout bugfix/release_01_fix1
```

Check that you have commits not yet pushed, only such commits Can be changed!!!



# Using the git log

## Exercise XII Continue

```
~/harry]$ git push origin --dry-run  
0ac136b..b597fd2 bugfix/release_01_fix1 -> bugfix/release_01_fix1
```

Change commit message to "We can edit unpushed messages only"

```
~/harry]$ git commit --amend -m "We can edit unpushed messages only"  
[bugfix/release_01_fix1 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
```