The Working Cycle

Exercise I: adding files

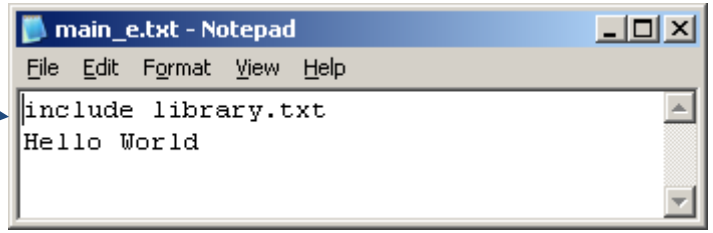
First say to Git who are you, this info will appear on each commit you do. Example:
$ git config --global user.name <YourUSERNAME> #Already, done? check it: git config --global -l
$ git config --global user.email <YourEMAIL>
$ git config --global init.defaultBranch main ## Set **main** branch to be default branch

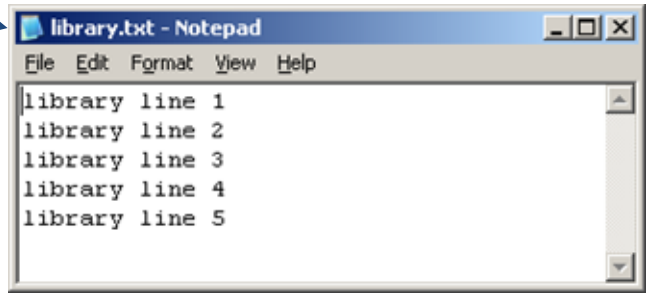
Please create a new (git init) ”bare” repository to be your origin(remote) repository
$ git init --bare --shared ~/git_remote_repo

Then clone it to harry repo: cd .git and overview repository objects
$ git clone ~/**git_remote_repo** ~/**harry**

Go to **harry** folder and create two files in your working directory (fill them with these text lines):
$ cd **~/harry ## Do it in any editor you work with, example: VS Code, Coursor, vim**

`main_e.txt`
`library.txt`





Add your changes to the index
~/**harry**]$ git add *

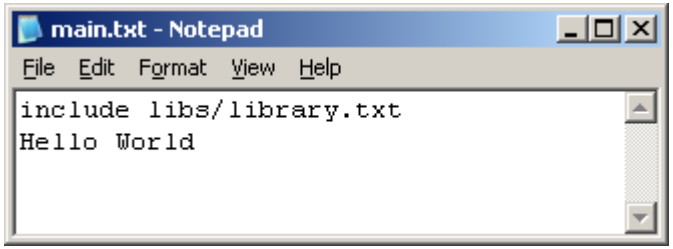
Check your status before committing
$ git status
Commit your changes as "Initial project"
$ git commit -m "Initial project"

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 ## Edit in any editor you prefer
$ git add -u



Check your status and diff before commit
$ git status $ git diff --staged main.txt

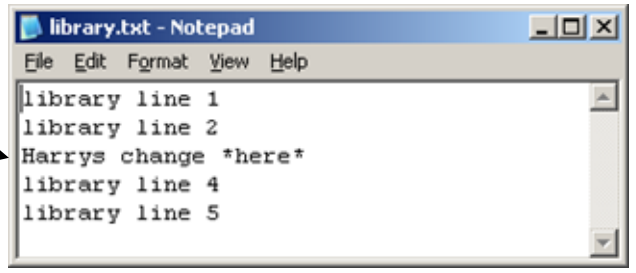
```
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       renamed:    library.txt -> libs/library.txt
#       new file:   main.txt
#       deleted:    main_e.txt
#
```

Commit your changes as "Restructuring project"
$ git commit -m "Restructuring project"

Push all your work to the origin (remote) repository
$ git push

```
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

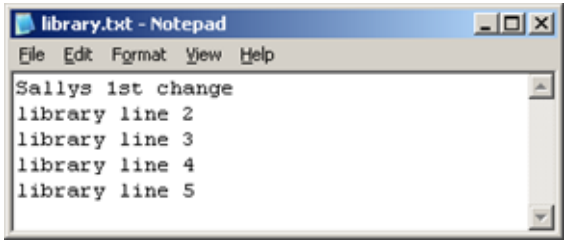


Change the 3rd line in **Harry**'s repo and check the status
~/**harry**]$ git status $ git diff HEAD ## Use any editor you want

Commit **Harry's** changes as "**Harry's** 1st changes in our library."
~/**harry**]$ git commit -a -m "Harrys 1st changes in our library"

Clone the git_remote_repo repo again to ~/**sally** folder, before pushing from **harry** repo
~/**harry**]$ git clone ~/git_remote_repo ~/sally

Push all your work in **Harry**'s repo to the origin (central) repo
~/**harry**]$ git push



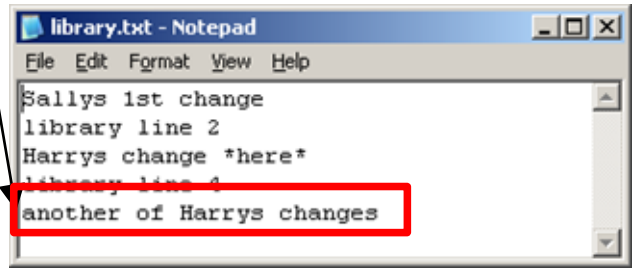
Change the first line in **Sally's** repo, commit it, and try to push
$ cd ~/sally ##Change the first line ~/**sally**]$ git diff
~/**sally**]$ git commit -a -m "Sallys 1st changes in our library"
~/**sally**]$ git push
Why did it fail?
Pull updates to **Sally's** repo prior to push:
~/**sally**]$ git pull --rebase Overview what you got: ~/**sally**]$ git status; git log

~/**sally**]$ git diff HEAD^ HEAD – compare the HEAD commit with the previous commit, do you see both Sally's and Harry's changes?
Push Sally's changes: ~/**sally**]$ git push

The Working Cycle
Exercise IV: conflicts (with rebase)

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

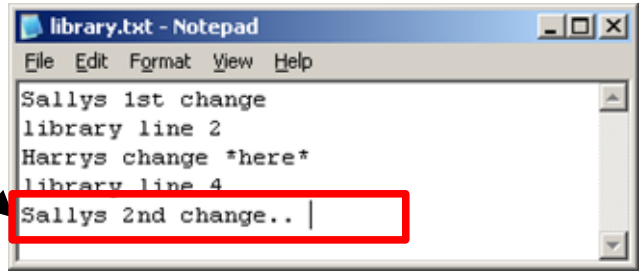
Change the last line of lib/library.txt in Harry's repo, see your change:
$git diff HEAD



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 last line of lib/library.txt in **Sally's** repo, commit, push
$ cd ~/sally
~/**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**]$ Open the file:
libs/library.txt
In you editor: **VS code** or vim, or any editor merge /diff editor you use. Fix conflicts (leave only 1 line or both)
Save the file libs/library.txt after the merge conflicts were solved

```
Sally's first change
Library line 2
Harry's change here
Library line 4
<<<<<<< HEAD
Another of Harrys change
=======
Sally 2nd change
>>>>>>> ad9e470 (+Sally 2nd change)
```

The Working Cycle
Exercise IV: conflicts (with rebase) (Continue)

Run "git add libs/library.txt" to mark resolution and add the file to staging area
~/**sally**]$ git add libs/library.txt

Run git status to see what is next?
~/**sally**]$ git status
rebase in progress; onto fe343dc. You are currently rebasing branch 'main' 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/main' 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 -p -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 all tags to origin repo
~/sally]$ git push --tags

```
* 9ec4c94 (HEAD -> main, tag: Release_01, origin/main, origin/HEAD) + Sally 2nd change
* 9c8fe9e +Another of Harrys change
* 39da746 +Sally's first change
* 0a09798 +Harry's change here
```

Go to Harry’s repo, $ cd ~/harry
You still do not see the tag in Harry’s repo
~/harry]$ git log --graph --oneline --all

Fetch all tags from the Harry’s repository
~/harry]$ git fetch --tags

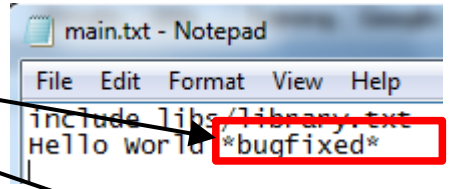
Run git log to see the tag in yellow color
~/harry]$ git log --graph --oneline --all

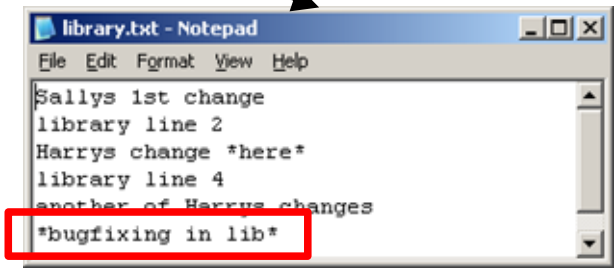
Rebase to align the local main branch with origin/main
Fast-forward rebase
~/harry]$ git rebase

Exercise VI: Branching

Create a branch named **`bugfix/release_01_fix1.`** All in Harry repo only!
~/harry]$ git branch bugfix/release_01_fix1
Check out this new branch:
~/harry]$ git checkout bugfix/release_01_fix1

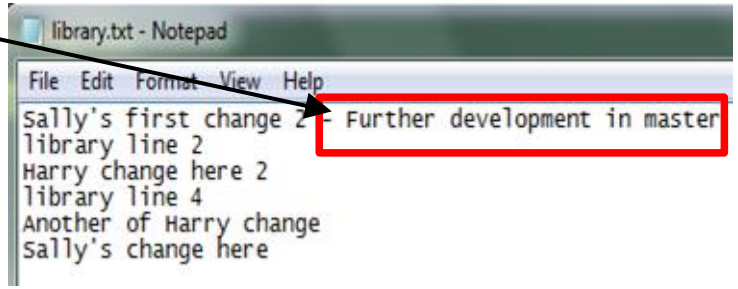
Change Harry's repo main.txt and libs/library.txt:
Then commit with the message "bugfixing" and push
~/harry]$ git commit -a -m "bugfixing"
~/harry]$ git push --set-upstream origin bugfix/release_01_fix1
Check if the remote tracking branch was set, for bugfix/release_01_fix1
~/harry]$ git branch -vv
* bugfix/release_01_fix1 afae908 [origin/bugfix/release_01_fix1]





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

Add a line in **Harry's** repo libs/library.txt main branch,
diff, and commit it as "**making progress in the main**",
Then check the status and push the change
~/harry]$ git diff HEAD



~/harry]$ git commit -a -m "making progress in the main"
See the new branch in the log graph
~/harry]$ git log --graph --oneline --branches

~/harry]$ git status and **~/harry**]$ git push

Exercise VII: Merging

Check if you are on the main branch in the Harry repo

```
~/harry]$ git status
# On branch main
Your branch is up to date with 'origin/main'.
nothing to commit, working tree clean
```

Merge bugfix/release_01_fix1 into the main branch

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

Save the commit message

```
commit ef91551fa6e3a8ed8087905c25ecfccbaf4cd643 (HEAD -> main)
Merge: 2a0a04e 646ecd0
Author: ilya <ilya@domain.com>

    Merge branch 'bugfix/release_01_fix1'
```

Check if the merge adds the bugfix to the main branch, and you see the main change

```
~/harry]$ git log -p
~/harry]$ git diff HEAD^^ HEAD
~/harry]$ git log --graph --oneline --all       ## See the branch merge
```

Push the merge changes to the original repo,
See file annotation

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

```
 2a0a04e1 libs/library.txt (ilya 2016-01-24 12:39:00 +0200 1) Sally change - further dev on main
^5d236de library.txt      (ilya 2016-01-23 21:16:41 +0200   2) library line 2
30a7edb1 libs/library.txt (ilya 2016-01-24 12:00:00 +0200 3) Change harry
^5d236de library.txt      (ilya 2016-01-23 21:16:41 +0200   4) library line 4
a8ac8654 libs/library.txt (ilya 2016-01-24 16:16:52 +0200 5) Another Harry's change
a8ac8654 libs/library.txt (ilya 2016-01-24 16:16:52 +0200 6) Sally's 2nd change
646ecd0a libs/library.txt (ilya 2016-01-24 16:48:48 +0200 7) "bugfixing in lib"
```

Check Out, Branch and Cherry Pick old commit

Exercise VIII

```
commit d01409754a6d77cdb1e7f2c3a72394fee7ddeb0d (tag: Release_01)
Merge: 75c03db [illegible]59572
Author: irokhk[illegible]
Date:   Sun Jan 5 15:20:13 2014 +0200

    Merge of Sally
```

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

Run log to find the 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 the specific revision, Tag Release_01 points to, or its SHA1
~/harry]$ git checkout Release_01 Or ~/harry]$ git checkout d01409754a6d77cdb1e7f2c3a72394fee7ddeb0d

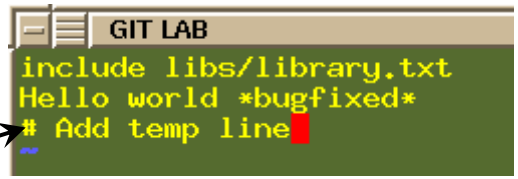
Check out the merge and add the bugfix to the main branch
~/harry]$ git checkout -b bugfix/release_01_fix2 Or ~/harry]$ git branch bugfix/release_01_fix2 Release_01

Cherry Pick commit ("**making progress in the main**") into current branch bugfix/release_01_fix2
~/harry]$ git log main --decorate --name-only and
Copy "**making progress in the main**" commit SHA1
~/harry]$ git cherry-pick <Paste, copied **making progress in the main** SHA1 here>
~/harry]$ git diff HEAD^ HEAD – review check-pick merge results
~/harry]$ git log --graph --oneline --all – Cherry-pick do not

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 the main branch in Harry's repo
~/harry]$ git checkout main
~/harry]$ git status
On branch main
nothing to commit, working tree clean



How to undo modifications

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

Changes not staged for commit:
(use "git add <file>..." to update what will be committed)
(use "git restore <file>..." to **discard changes in working directory**)
modified: main.txt

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

Discard the change in the Working Dir in the file main.txt. Remember that the discard is irreversible
~/harry]$ git diff – Always see what you are going to discard first!!!
~/harry]$ git restore main.txt (discard the change)

Run git status and see the status of the file now
~/harry]$ git status

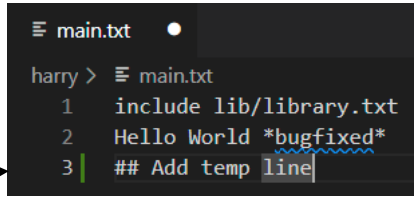
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

How to undo staged changes

Do the below 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 diff --staged (Compare HEAD and staged, see what you are going to discard first!!!)
~/harry]$ git status - See how to unstage
~/harry]$ git restore --staged main.txt (Unstage the file)
~/harry]$ git diff - See what you are going to discard
~/harry]$ git restore main.txt (discard the change) **~/harry**]$ git status - See the status now

Git Reverts of all kinds
Exercise IX Continue



How to revert latest local commit
Do the upper steps again (vi main.txt, git status, git diff)
~/harry]$ git commit -a -m "Adding temp line number 3 commit" - Commit all changed files

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

commit **a0a176a86717** (HEAD -> main)
Author: ilya <ilya@domain.com>

Adding temp line number 3 commit

M main.txt

Run git reset to reset the current branch to the previous commit
~/harry]$ git reset --hard HEAD^
(Be careful --hard resets branch, Index, and Working Tree also)

Run log now to see the previous good commit back to be first
~/harry]$ git log --name-status

commit **ef91551fa6** (HEAD -> main, origin/main, origin/HEAD)
Merge: 2a0a04e 646ecd0
Author: ilya <ilya@domain.com>

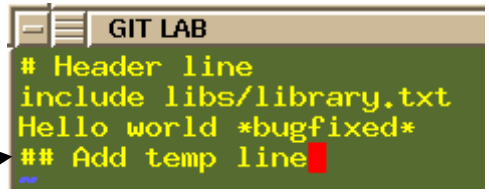
Merge branch 'bugfix/release_01_fix1'

Where is the temp commit? It exists, just not referenced, run reflog to see it:
~/harry]$ git reflog

ef91551 (HEAD -> main, origin/main, origin/HEAD) HEAD@{0}: reset: moving to HEAD^
a0a176a HEAD@{1}: commit: Adding temp line number 3 commit

Git Reverts of all kinds
Exercise IX Continue

How to Revert Remote Commit



Let us reset the main branch back to "Adding temp line number 3 commit" - Copy its SHA1
~/harry]$ git reset --hard <Paste "Adding temp line number 3 commit" SHA1 here>
HEAD is now at a0a176a Adding temp line number 3 commit
Push the commit to the original repo
~/harry]$ git push

See the log, the "Adding temp line number 3 commit" commit must be back in the main branch
~/harry]$ git log -2 -p

Now let us revert the already pushed to origin commit, never mind who did it.
Let us revert the HEAD commit: "Adding temp line number 3 commit."
~/harry]$ git revert HEAD Or $ git revert < "Adding temp line number 3 commit" SHA1>

Revert "Adding temp line number 3 commit"
This reverts commit a0a176a – Save the commit message as is
Run diff of the file main.txt to see what we get after reverting
~/harry]$ git log -2 -p

Run git status and see the revert commit
~/harry]$ git status

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

Git Reverts of all kinds
Exercise IX Continue

How to Revert Remote Commit in a feature branch – replacing it completely

See what local branches you have in harry's repo
~/harry]$ git branch
  bugfix/release_01_fix1
  bugfix/release_01_fix2
* main

Checkout bugfix/release_01_fix1 branch
~/harry]$ git checkout bugfix/release_01_fix1
Let us reset the branch to the previous commit HEAD~1 commit
~/harry]$ git log -2 – See the previous to the HEAD commit we are going to reset to
~/harry]$ git reset --hard HEAD~1
~/harry]$ git log -1 – See the HEAD commit now, bugfix/release_01_fix1 is reset to that commit
Force push the commit to the original repo
~/harry]$ git push --force-with-lease
 ! [rejected] bugfix/release_01_fix1 -> bugfix/release_01_fix1 (non-fast-forward)
Why? Need to explicitly allow non-fast-forward push to our origin, bare repo
First, check our repo origin bare repo path/URL
~/harry]$ git remote -v
Now, let us change the remote bare repo config to allow non-fast-forward push (**-C** - Change Dir)
~/harry]$ git -C **~/git_remote_repo** config receive.denyNonFastForwards false

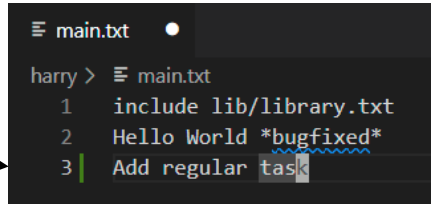
Force push the previous commit. Can you find the reverted commit on the remote? How?
~/harry]$ git push --force-with-lease # -> bugfix/release_01_fix1 (forced update)

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 your work, do an emergency fix, commit it, and pop your stash back

~/harry]$ git checkout main

Do regular tasks in the main branch, change the file main.txt, and add a line "Add regular task"

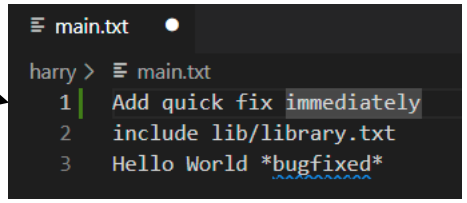


~/harry]$ code main.txt Or $ vi main.txt

~/harry]$ git stash

Saved working directory and index state WIP on main: 00588e9 Revert "Adding temp line number 3 commit"

Do an emergency fix in the main branch, change the file main.txt, and add the first line "Add quick fix immediately"



~/harry]$ code main.txt Or $ vi main.txt

~/harry]$ git commit -a -m "Fix a bug in a quick"

~/harry]$ git stash pop

Auto-merging main.txt

Here can also be merge conflicts, solve it as we already learned

Now continue your work, diff changes, commit, and push

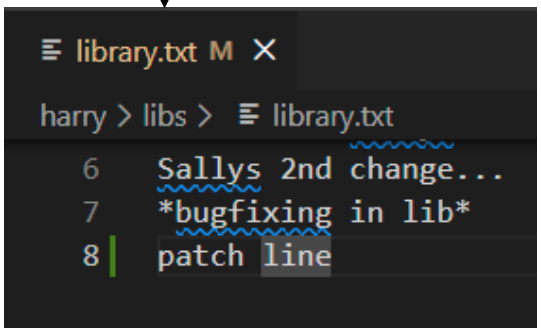
~/harry]$ git diff

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

~/harry]$ git push

Format/Apply Patch
Exercise XI

Go to Harry's repository and add a line to the END of file libs/library.txt – "patch line"
$ cd ~/harry/libs (main)
**~/harry/libs]$** code library.txt Or **~/harry/libs**]$ vi library.txt



Commit the change in Harry's repository
~/harry/libs (main)
~/harry]$ git commit -am "harrys commit for patch"

Format the patch of the last commit
~/harry/libs]$ git format-patch -1 main
0001-harry-s-commit-for-patch.patch

See the patch file on the file system
~/harry/libs]$ ls
0001-harry-s-commit-for-patch.patch library.txt

See the content of the patch file. Does it look familiar?
~/harry/libs]$ cat 0001-harry-s-commit-for-patch.patch

Go to Sally's repository and pull the latest commits
$ **cd ~/sally/libs**
~/sally/libs (main)
~/sally/libs]$ git pull

Format/Apply Patch
Exercise XI Continue

See what you got in the patch
$ ~/sally/libs (main)
~/sally/libs]$ git apply --stat ~/harry/libs/0001-harrys-commit-for-patch.patch
libs/library.txt | 1 +
 1 file changed, 1 insertion(+)

Check if the patch is applicable: "no output, no errors."
~/sally/libs] $ git apply --check ~/harry/libs/0001-harrys-commit-for-patch.patch

Apply the patch with the signature of the patch
~/sally/libs] $ git am --signoff < ~/harry/libs/0001-harrys-commit-for-patch.patch
Applying: harry's commit for patch

~/sally/libs] $ git log -1 -c
commit d3eb9fd709 (HEAD -> main)
    harry's commit for patch
    Signed-off-by: ilya <ilya@domain.com>
diff --git a/libs/library.txt b/libs/library.txt
...
+patch line
(END)

Squash several commits to 1 commit (rebase -i) Exercise XII

In Sally's repository do 2 changes and 2 commits

We want to squash 2 commits 9fb9d06 and 5822495
$ ~/sally (main)
~/sally] $ git log --pretty=oneline -3
9fb9d060ab4ffe34f033d7b61163ce6697894a7b Add 2nd commit for squash
5822495ba795f4b0a18e89d52e77473dcd700c65 Add 1st commit line
d3eb9fd709ed7614f0755a496d05d695acc6114f harry's commit for patch

Run git rebase interactively on the commit **before** 2 commits you want to squash
$ ~/sally (main)
~/sally] $ git rebase -i HEAD^^ OR **~/sally**] $ git rebase -i d3eb9fd709ed76
The first commit **pick** (leave it)
The second **s**quash into the first commit
Save the commit message as is, and exist

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

[detached HEAD e33ede3] Add 1st commit line
2 files changed, 3 insertions(+)
Successfully rebased and updated refs/heads/main.

Run now **~/sally**] $ git log -p -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
~/sally] $ git reflog
~/sally] $ git push

Fetch directly from another repository, not origin. Exercise XIII

See the local branches at sally's repo. What does the * mean?
~/sally/libs (main)
~/sally/libs] $ git branch
* main

Let us check the remote tracking branches. What does **origin** here mean?
~/sally/libs] $ git branch -r

Let us pull from harry'r repo branch directly
~/sally/libs] $ git fetch ~/harry bugfix/release_01_fix1:bugfix/release_01_fix1

Let us see the logs of the 2 repositories again
~/sally/libs] $ git log -1 --oneline bugfix/release_01_fix1
* 55bc145 (origin/bugfix/release_01_fix1, bugfix/release_01_fix1)
~/sally/libs] $ git **-C ~/harry** log -1 --oneline bugfix/release_01_fix1
* 55bc145 (origin/bugfix/release_01_fix1, bugfix/release_01_fix1)

Check out the branch and see list of branches
~/sally/libs] $ git checkout bugfix/release_01_fix1
~/sally/libs] $ git branch

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 (main)
$ vi library.txt
```

Commit the change in Sally's repository

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

Push the change to origin

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

Go to Harry's repository

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

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

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

Rebase (Already present in TeamWork)
Exercise VIII Continue

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

Fetch the Sally's commit
$ ~/harry/libs (main)
$ git fetch
From c:/Users/Ilya/git_remote_repo
   06cfb7a..4a14da9 main   -> origin/main

See the log, where is the Sally's commit and where is Harry's commit, are they on the same line?
$ ~/harry/libs (main)
$ git log --graph --oneline --branches

Now run rebase
$ ~/harry/libs (main)
$ 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 (main)
$ git log --graph --oneline --branches

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

Change sally's file

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

$ git diff HEAD

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

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

$ ~/sally/libs (main)
$ git push ~/harry main:main

Worked? Why not? Go to Harry's repository
$ ~/sally/libs (main)
$ cd ~/harry/libs

Check another branch:
~/harry]$ git checkout bugfix/release_01_fix1

Back to Sally's repository $ cd ~/sally

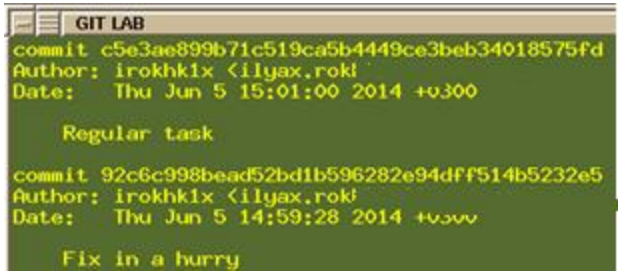
$ ~/sally (main)
$ git push ~/harry main:main
To c:/Users/Ilya/harry 93dffff..fde4fd3 main -> main

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

Advanced working with branches
Exercise XI

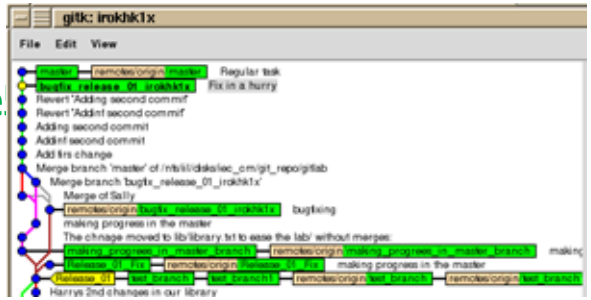
How to take changes that happened on the main in the meanwhile

Let us check out bugfix/release_01_fix1 branch
~/harry]$ git checkout bugfix/release_01_fix1

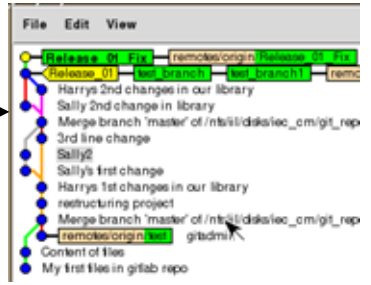


Run git log --all (refs) and copy „Fix in a hurry" commit sha1
~/harry]$ git log –all
Now let us merge our branch bugfix/release_01_fix1 with commit HEAD^ on main branch
92c6c998bead52bd1b596282e94dff514b5232e5

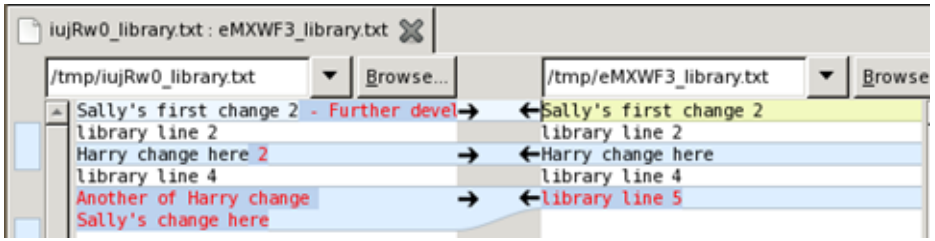
~/harry]$git merge 92c6c998bead52bd1b596282e94dff514b5232e
Updating 0ac136b..92c6c99
Fast-forward
ilyar/libs/library.txt | 2 +-
ilyar/main.txt | 3 ++-
2 files changed, 3 insertions(+), 2 deletions(-)



Compare my branch to the main (as it was when I last updated from main)
~/harry]$ git checkout bugfix/release_01_fix1
Run Git log to see when last time was merged from main?
~/harry]$ git log –oneline –graph --all



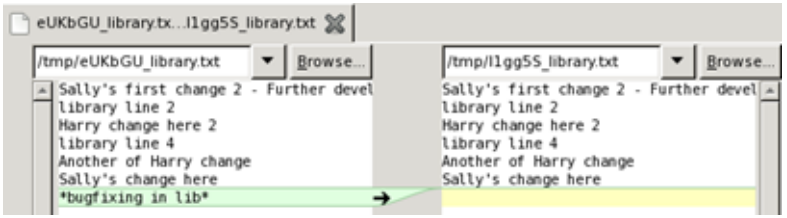
Advanced working with branches
Exercise XI Continue



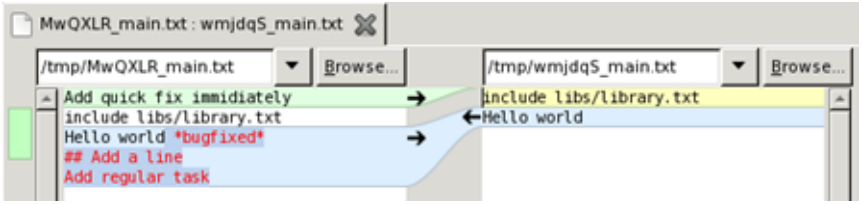
~/harry]$ git diff HEAD bccdb778dc129c990a0c09147c914a34eff12dda

How to Compare two branches

Let us diff main and bugfix_release_01_fix1 branches
~/harry]$ git difftool main bugfix/release_01_fix1



~/harry]$ git diff main 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 immidiately
 include libs/library.txt
-Hello world *bugfixed*
-## Add a line
-Add regular task
+Hello world

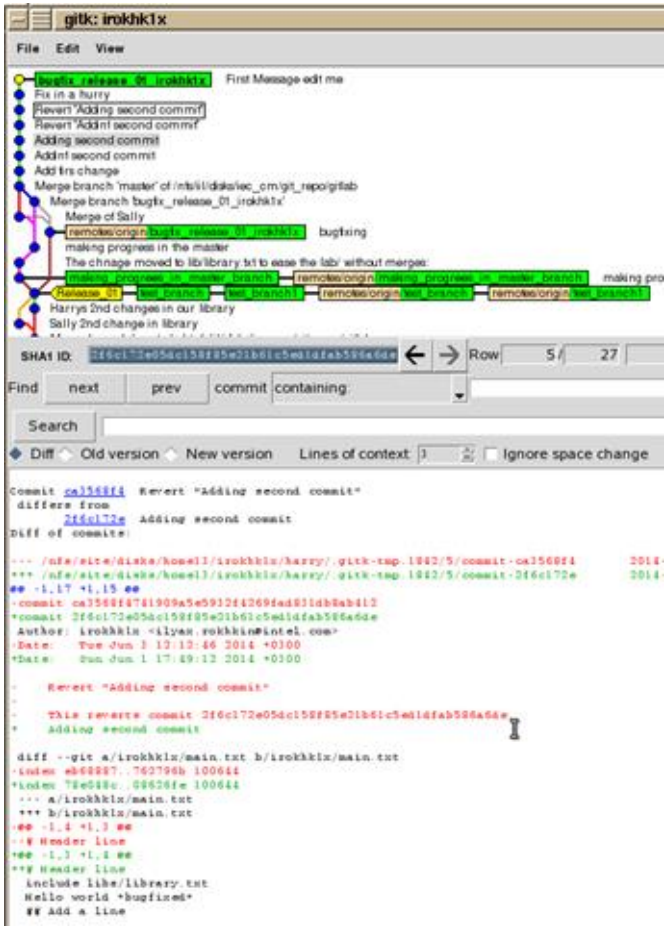


Using the git log
Exercise XII

Git log in cmd

This is the log command in cmd to see merge info + files, that were changed + all branches
~/harry]$ git log -p --branches

~/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
```