Merge Practice

Some examples of merging and conflict resolution.

A Common Problem

1. Developer A clones a repo from Github, or "pulls" latest rev from github. Now his local copy is up to date!
2. Developer A starts work on his local copy.
3. Developer B (same person or other) directly edits a file on Github -- quite often its README.md, and saves it. That is a new commit.
4. Developer A finishes his work, commits it, and does "git push" to Github.

What Happens?

What Happens?

```
dev-A> git commit -m "add tests for ..."
dev-A> git push
! [rejected]   master -> master (fetch first)
error: failed to push some refs to https://github.com/...
hint: Updates were rejected because the remote
hint: contains work that you do not have locally.
hint: This is usually caused by another repository
hint: pushing to the same ref.
```

Try It!

Use your `unittesting` repo on Github.

Dev A: On your own computer...

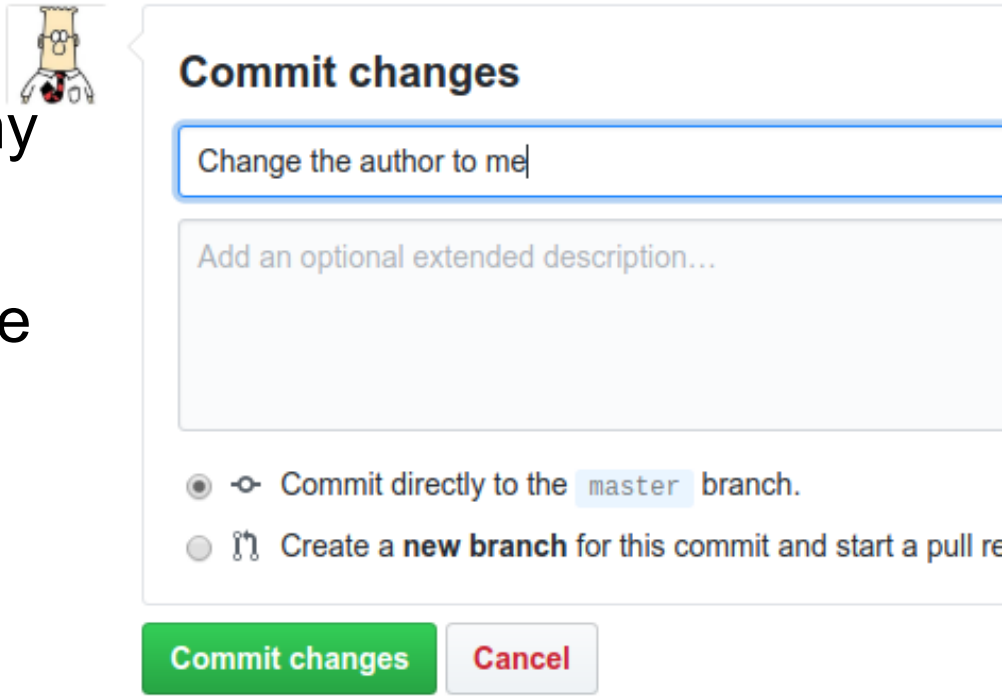
1. Make sure your local repo and Github are in sync.

```
cmd> cd workspace/unittesting  (wherever you cloned)
cmd> git status
On branch master
Your branch is up to date with 'origin/master'.
cmd> git push
Everything up-to-date   (no changes to push)
```

Alternative: Create a fresh clone from Github
cmd> cd workspace
cmd> git clone [github-unittesting-url] unittesting-merge

Dev B: Make a change on Github

1. Browse to https://github.com/ISP19/unittesting-your_github_id
2. The README.md file is shown at bottom. Click edit icon to edit it.
3. Change the author -- any change is OK.
4. Write a commit message and click [Commit changes]



Dev A: Commit work your local repo

1. Edit README.md and change some lines:

```
## Unit Testing Assignment
Copyright by Bill Gates and Fatalai Jon
```

2. Commit the changes to your repo `git commit -a -m "..."`

3. Push your changes to Github

```
cmd> git push
To https://github.com/ISP19/unittesting-fatalaijon.git
 ! [rejected]     master -> master (fetch first)
error: failed to push some refs to 'https://github.com/ISP19/
  unittesting-fatalaijon.git'
hint: Updates were rejected because the remote contains
hint: work that you do not have locally. This is usually
hint: caused by another repository pushing to the same ref.
```

What Happened? What to Do?

Your local "master" is "behind" the master on Github.

Your work is based on a previous commit to master.

Need to update your local repo before you can push.

```
cmd> git pull
remote: Total 3 (delta 2), reused 0 (delta 0), pack-
  reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/ISP19/unittesting-fatalaijon
   81fcef6..855692a  master     -> origin/master
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit
  the result.
```

Understand the conflict

"git status" might help...

```
cmd> git status
On branch master
Your branch and 'origin/master' have diverged,
and have 1 and 1 different commits each, respectively.
  (use "git pull" to merge the remote branch into yours)

You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
both modified:   README.md
```

Fixing conflicts

1. For text files, conflicts are written into your working copy. Example in next slide.
2. You have to edit the files and fix the conflicts.
   - or use a GUI merge tool and "`git mergetool`"
3. After fixing conflicts use "`git commit`" to resolve the conflict and save changes.

--- or ---

3b. If you can <u>not</u> fix conflicts, abort the merge operation:

```
git merge --abort
```

this resets your working copy to before the merge.

View the conflicts

Edit the file to see what has changed.

```
cmd> edit README.md
++<<<<<<< HEAD
+by Bill Gates and Fatalai Jon
++=======
+by Fatalai Jon.
++>>>>>>> 855692a4b30b46645e51999bb27e72adbfbaca9d
```

Understanding diffs

"**`diff`**" is a Unix command to show differences between text files. It show:

- lines changed (differences)
- lines added in one file
- lines deleted in one file

may show surrounding identical lines for *context*.

Example: make 2 copies of a text file. Change one copy (add lines, change lines, delete lines). Run diff:

```
cmd>  diff  a.txt  b.txt
```

Git diffs

Git displays differences between files using "diff" notation.

Example: Go to a repository you already have.

1. Verify your working copy is "clean": **`git status`**

2. Edit README.md (on any text file, including *.py).

2. What has changed?

`cmd>` **`git diff`**

Output of git diff

```
diff --git a/README.md b/README.md
index ff3ac4b..1434aa0 100644
--- a/README.md
+++ b/README.md
@@ -1,6 +1,6 @@
 ## Unit Testing Assignment

-by Bill Gates.
+by Bill Gates.   ARE YOU REALLY BILL GATES???
```

"git pull" = "git fetch" + "git merge"

"git pull" performs two commands:

`git fetch` - fetch updates from a remote repository.
It saves the remote in a separate branch named:

`origin/master` or `origin/branchname`

`git merge` - merge two development histories.

If you don't specify which branches to merge,

the default is HEAD and origin/tracking_branch_name

git fetch and diff

To see what has changed *before* you merge do this:

1. fetch the remote branch: **`git fetch`**
2. in your local repo, the branch you just fetched is named **`origin/master`** or **`origin/branch-name`**
3. view differences between working copy and remote:

   `git diff origin/master`

== or ==

4. view differences between local HEAD and remote:

   `git diff HEAD origin/master`

Example

```
cmd>  git fetch
remote: Total 3 (delta 2), reused 0 (delta 0), ...
Unpacking objects: 100% (3/3), done.
From github.com:ISP19/unittesting 92448bb..5f828df
  master -> origin/master
cmd>  git diff HEAD origin/master
diff --git a/README.md b/README.md
index ff3ac4b..1434aa0 100644
--- a/README.md
+++ b/README.md
@@ -1,6 +1,6 @@
 ## Unit Testing Assignment

-by Bill Gates.
+by Bill Gates.   ARE YOU REALLY BILL GATES???
```

Example merge

Viewing the "diffs" can help you decide how to merge. You may decide to use the remote version, use your version, or combine them using an editor or mergetool.

```
cmd> git merge
```

(resolve the conflicts using editor or mergetool)

Then run:

```
cmd> git commit -m "Merged. Conflicts resolved"
```

Visual Merge Tools

1. PyCharm and PyDev (Eclipse) have builtin GUI tool for viewing diffs and merging branches
2. meld and diffuse are good tools known by git.

```
cmd> git help mergetool
```