

*This Linux contribution guide was created by Theodore from the Q&A community and ported to the wiki by StevenPuttemans.*

## OpenCV Contribution Guide in Linux Environment

### Prerequisites

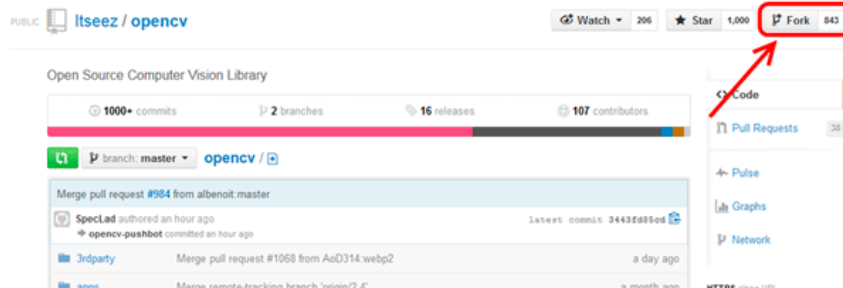
- Linux x64 bit operating system (e.g. Arch Linux / Ubuntu / OpenSUSE / RedHat / ...)
- Git (git-2.2.2-1)

### Setting Up

1. Register at Github
2. Install Git
  - `pacman -S git` (Arch Linux)
  - `sudo apt-get install git` (Ubuntu)

### Forking OpenCV repository

1. Go to GitHub
2. Login to your account using your Github username and password
3. Open the Itseez OpenCV GitHub page
4. Press the fork icon on the right hand top side of the page which will help you to make a personal copy of the OpenCV sourcecode:



5. When you are done return back to your own github page where you should



find the copied repository:

6. This means that you successfully forked the OpenCV repository to your personal account. We are ready now to clone the forked repository to our

local environment, where will make the actual changes to the source code.

## Cloning the forked OpenCV repository on your local machine

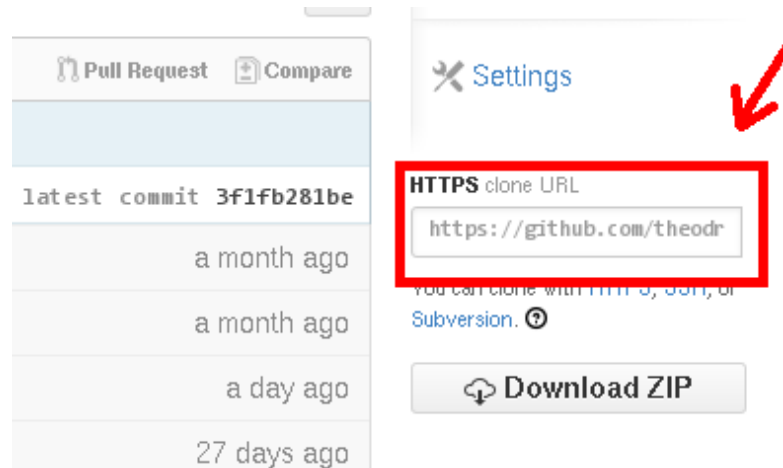
1. Go to your local desktop environment and open your favourite terminal emulator. (*I consider that you know what a terminal is and how to execute commands on the command prompt.*) Browse to the path/folder where you want to sync up the OpenCV source code from your fork. In my case I decided to locate it into the `/home/theodore/documents/git` folder.

```
theodore ~ $ cd /home/theodore/documents/git
theodore ~/documents/git $ |
```

2. We are ready now to clone the forked repository in our GitHub account to the current folder. Just type:

```
git clone <link_to_your_forked_repository_in_your_GitHub_account>
<name>
```

you will find the **link** on your account in GitHub right here:



and for the **name** just choose something that you like and correlates that you have the source code of OpenCV library. I choose `opencv_source`.

```
theodore ~/documents/git $ git clone https://github.com/theodr/opencv.git opencv_source
Cloning into 'opencv_source'...
remote: Counting objects: 166856, done.
remote: Compressing objects: 100% (57/57), done.
Receiving objects: 37% (62775/166856), 198.36 MiB | 504.00 KiB/s
```

If everything went right, you should end up with something like this:

```
theodore ~/documents/git $ git clone https://github.com/theodr/opencv.git opencv_source
Cloning into 'opencv_source'...
remote: Counting objects: 166856, done.
remote: Compressing objects: 100% (57/57), done.
remote: Total 166856 (delta 23), reused 16 (delta 1)
Receiving objects: 100% (166856/166856), 420.70 MiB | 649.00 KiB/s, done.
Resolving deltas: 100% (114112/114112), done.
Checking connectivity... done.
theodore ~/documents/git $
```

## Git Configuration

1. Before we start making any changes into the source code let's configure some global git attributes. Every commit you make will have your name and email address to identify the 'owner' of the commit, so you should start by giving it those values. To do so, go into the source folder:

```
theodore ~/documents/git $ cd opencv_source/
```

and run these commands:

```
git config --global user.name "Your Name"
```

```
git config --global user.email "your@email.com"
```

```
[theodore@tt ~/documents/git/opencv_source]$ git config --global user.name "theodore"
[theodore@tt ~/documents/git/opencv_source]$ git config --global user.mail "theodore@gmail.com"
[theodore@tt ~/documents/git/opencv_source]$
```

2. Install the default pre-commit hook by renaming `opencv_source/.git/hooks/pre-commit.sample` to `opencv_source/.git/hooks/pre-commit` - this will prevent you from committing whitespace errors.

```
theodore ~/documents/git/opencv_source $ mv -v .git/hooks/pre-commit.sample .git/hooks/pre-commit
'.git/hooks/pre-commit.sample' -> '.git/hooks/pre-commit'
theodore ~/documents/git/opencv_source $
```

## Making actual changes to the sourcecode

1. First let's see how is the status of our local repository and in which branch we are. We can achieve that by giving:

```
git status
```

```
git branch
```

```
git branch -a
```

```

theodore ~/documents/git/opencv_source $ git status
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working directory clean
theodore ~/documents/git/opencv_source $ git branch
* master
theodore ~/documents/git/opencv_source $ git branch -a
* master
  remotes/origin/2.4
  remotes/origin/2.4.10.x-prep
  remotes/origin/2.4.8.x-prep
  remotes/origin/2.4.9.x-prep
  remotes/origin/HEAD -> origin/master
  remotes/origin/dist_transf_tutorial
  remotes/origin/master
theodore ~/documents/git/opencv_source $

```

We see at the moment that the master branch is active and that it is up-to-date with the `origin/master`. Moreover, with the last command we see which other branches are available.

2. According to the official contribution page you can choose a base branch for your work. You have two options:
  1. 2.4 - the branch that will be used for future minor releases in the 2.4.x line. Choose it when fixing a bug that's reproducible in this branch, or when making performance optimizations relevant to it. 2.4 has been feature-frozen; as such, new functionality will not be accepted.
  2. master - the branch that will be used for the next major release of OpenCV (tentatively, 3.0). Choose it when adding new functionality, or for bugfixes/optimizations that don't apply to 2.4.

If you decide that the 2.4 branch is the one that you should go, you need to shift to this branch. The way to do that is by typing:

```
git checkout 2.4
```

```

theodore ~/documents/git/opencv_source $ git checkout 2.4
Branch 2.4 set up to track remote branch 2.4 from origin.
Switched to a new branch '2.4'
theodore ~/documents/git/opencv_source $ git status
On branch 2.4
Your branch is up-to-date with 'origin/2.4'.
nothing to commit, working directory clean
theodore ~/documents/git/opencv_source $ git branch -a
* 2.4
  master
  remotes/origin/2.4
  remotes/origin/2.4.10.x-prep
  remotes/origin/2.4.8.x-prep
  remotes/origin/2.4.9.x-prep
  remotes/origin/HEAD -> origin/master
  remotes/origin/dist_transf_tutorial
  remotes/origin/master
theodore ~/documents/git/opencv_source $

```

3. I decided to work with the master branch. Be sure that you are on the master branch with the *status* option, if you are not switch to it with:

```
git checkout master
```

```

theodore ~/documents/git/opencv_source $ git checkout master
Switched to branch 'master'
Your branch is up-to-date with 'origin/master'.
theodore ~/documents/git/opencv_source $ |

```

Once, you have made sure that you are on the master branch create a new branch on your Git repository, which derives from master, to apply changes. Do not forget to create a new branch for every single update you want to do! These branches are local, but will be pushed to GitHub after editing the source code. Choose a correct name for your branch, which will be representative to the changes that you want to make. To actually create a new branch, add the **name** of your new branch after the command:

```
git branch <name>
```

I chose to name it *bugfix\_1*, and then I verified that it has correctly been created.

```

theodore ~/documents/git/opencv_source $ git status
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working directory clean
theodore ~/documents/git/opencv_source $ git branch bugfix_1
theodore ~/documents/git/opencv_source $ git branch -a
2.4
bugfix_1
* master
remotes/origin/2.4
remotes/origin/2.4.10.x-prep
remotes/origin/2.4.8.x-prep
remotes/origin/2.4.9.x-prep
remotes/origin/HEAD -> origin/master
remotes/origin/dist_transf_tutorial
remotes/origin/master
theodore ~/documents/git/opencv_source $ |

```

Then switch to the newly created branch. We are ready now to start making changes into the source code.

`git checkout <branch_name>`

```

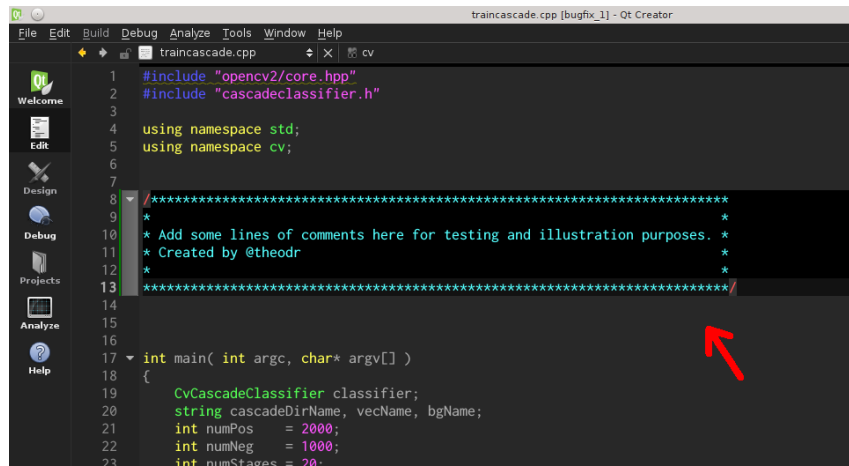
theodore ~/documents/git/opencv_source $ git checkout bugfix_1
Switched to branch 'bugfix_1'
theodore ~/documents/git/opencv_source $ git status
On branch bugfix_1
nothing to commit, working directory clean
theodore ~/documents/git/opencv_source $ |

```

or both creation and switching can be shorted with:

`git checkout -b <name>`

4. Open now your desktop file manager, and go to the path where you have cloned the source code (i.e. here `/home/theodore/documents/git/opencv_source`). Select the file you want to adapt (in our case it's an adding extra comment to a source code file). In our case, we open up the file `apps/traincascade/traincascade.cpp`, and we modify it.



```
1 #include "opencv2/core.hpp"
2 #include "cascadeclassifier.h"
3
4 using namespace std;
5 using namespace cv;
6
7
8 /*****
9  *
10 * Add some lines of comments here for testing and illustration purposes. *
11 * Created by @theodr *
12 *
13 *****/
14
15
16
17 int main( int argc, char* argv[] )
18 {
19     CvCascadeClassifier classifier;
20     string cascadeDirName, vecName, bgName;
21     int numPos = 2000;
22     int numNeg = 1000;
23     int numStages = 20;
```

Save the file so that your adaptations are stored onto the system. Then again on your terminal type:

```
git status
```

to see that indeed the file is modified.

```
git add .
```

```
git commit -m '<representative_message_for_the_commit>'
```

to commit these changes and

```
git push origin <name>
```

or

```
git push origin --all
```

to push these changes, from a specific branch that you will name with the *name* attribute or from all the available branches to GitHub.

```

theodore ~/documents/git/opencv_source $ git status
On branch bugfix_1
Your branch is up-to-date with 'origin/bugfix_1'.
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:   apps/traincascade/traincascade.cpp

no changes added to commit (use "git add" and/or "git commit -a")
theodore ~/documents/git/opencv_source $ git add .
theodore ~/documents/git/opencv_source $ git commit -m 'My first commit into bugfix_1'
[bugfix_1 7be5f7f] My first commit into bugfix_1
1 file changed, 1 deletion(-)
theodore ~/documents/git/opencv_source $ git push origin bugfix_1
Counting objects: 5, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 425 bytes | 0 bytes/s, done.
Total 5 (delta 4), reused 0 (delta 0)
To https://github.com/theodr/opencv.git
   eb59ddd..7be5f7f  bugfix_1 -> bugfix_1
theodore ~/documents/git/opencv_source $
theodore ~/documents/git/opencv_source $
theodore ~/documents/git/opencv_source $ git push origin --all
Everything up-to-date
theodore ~/documents/git/opencv_source $ |

```

during the push procedure bear in mind that you might be asked to insert your GitHub login info (i.e. username and password).

---

**Edit(update):** (“squash” commits with *rebase* command)

As it is stated from the devs it is good to try not to include “oops” commits - ones that just fix an error in the previous commit or similar. If you have those or if you asked to merge any kind of commits, this can be achieved either before submitting these commits or if it is already late afterwards. In *git* this known as **squash**. This means that you can squash those fixes directly into the commits where they belong. You can achieve this with the *rebase* command:

- First be sure that you are in the correct branch with the `git status` command
- Obtain the full amount of commits you have done within that branch by:

```
git rev-list --count master..<your_branch_name>
```

- Finally access those commits with:

```
git rebase -i HEAD~<number_obtain_from_previous_step>
```

```

theodore ~/documents/git/opencv_source $ git status
On branch morph_lines_detection_tut
nothing to commit, working directory clean
theodore ~/documents/git/opencv_source $ git rev-list --count master..morph_lines_detection_tut
9
theodore ~/documents/git/opencv_source $ git rebase -i HEAD~9

```

- If everything went fine you will get something similar to the following, where you will be able to see the history of your commits prefixed



(i.e. heads) with the *pick* option.

```
pick 52535b3 Fix for assert in compute when subimage is used (see issue #4149)
pick 098389d fix opencv_run_all_tests_unix.sh script:
pick a857499 Morphology Lines Extraction Tutorial v.1
pick 391d58b Morphology Lines Extraction v.2, warning fixes
pick 5851f9a Morph lines detection tutorial fixes, plus cover images addition
pick 3f4af2a bold italic and asterisks issues
pick ce63417 bold italic issue
pick e954ddf bold italic and asterisks issues
pick 314a85 minor spellchecking fix
pick c6bf01c checkspelling fixes

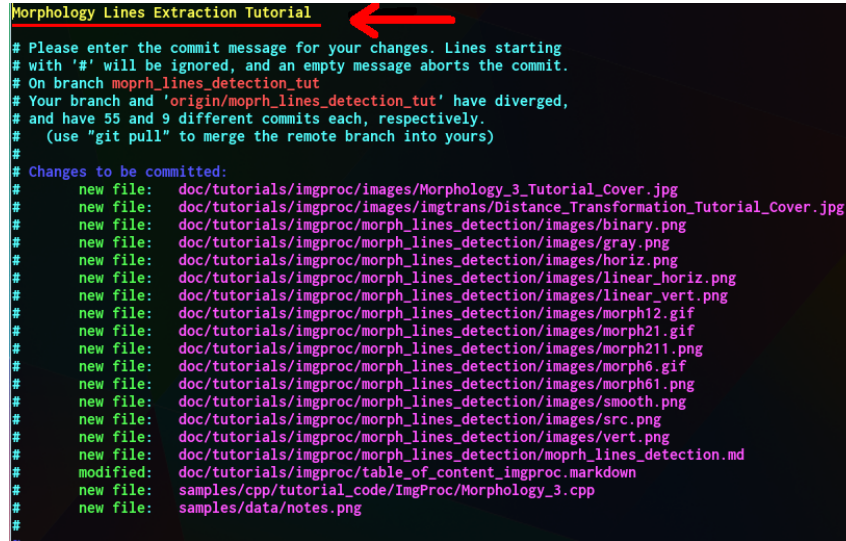
# Rebase 1e47c6c..c6bf016 onto 1e47c6c (10 TODO item(s))
#
# Commands:
# p, pick = use commit
# r, reword = use commit, but edit the commit message
# e, edit = use commit, but stop for amending
# s, squash = use commit, but meld into previous commit
# f, fixup = like "squash", but discard this commit's log message
# x, exec = run command (the rest of the line) using shell
#
# These lines can be re-ordered; they are executed from top to bottom.
#
# If you remove a line here THAT COMMIT WILL BE LOST.
#
# However, if you remove everything, the rebase will be aborted.
#
# Note that empty commits are commented out
```

- Replace all commit heads (first word) with a *s* or *squash* prefix:

```
pick 52535b3 Fix for assert in compute when subimage is used (see issue #4149)
squash 098389d fix opencv_run_all_tests_unix.sh script:
squash a857499 Morphology Lines Extraction Tutorial v.1
squash 391d58b Morphology Lines Extraction v.2, warning fixes
squash 5851f9a Morph lines detection tutorial fixes, plus cover images addition
squash 3f4af2a bold italic and asterisks issues
squash ce63417 bold italic issue
squash e954ddf bold italic and asterisks issues
squash 314a85 minor spellchecking fix
squash c6bf01c checkspelling fixes

# Rebase 1e47c6c..c6bf016 onto 1e47c6c (10 TODO item(s))
#
# Commands:
# p, pick = use commit
# r, reword = use commit, but edit the commit message
# e, edit = use commit, but stop for amending
# s, squash = use commit, but meld into previous commit
# f, fixup = like "squash", but discard this commit's log message
# x, exec = run command (the rest of the line) using shell
#
# These lines can be re-ordered; they are executed from top to bottom.
#
# If you remove a line here THAT COMMIT WILL BE LOST.
#
# However, if you remove everything, the rebase will be aborted.
#
# Note that empty commits are commented out
```

- Then make a global commit message:



```

Morphology Lines Extraction Tutorial
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# On branch morph_lines_detection_tut
# Your branch and 'origin/morph_lines_detection_tut' have diverged,
# and have 55 and 9 different commits each, respectively.
# (use "git pull" to merge the remote branch into yours)
#
# Changes to be committed:
#   new file:   doc/tutorials/imgproc/images/Morphology_3_Tutorial_Cover.jpg
#   new file:   doc/tutorials/imgproc/images/imgtrans/Distance_Transformation_Tutorial_Cover.jpg
#   new file:   doc/tutorials/imgproc/morph_lines_detection/images/binary.png
#   new file:   doc/tutorials/imgproc/morph_lines_detection/images/gray.png
#   new file:   doc/tutorials/imgproc/morph_lines_detection/images/horiz.png
#   new file:   doc/tutorials/imgproc/morph_lines_detection/images/linear_horiz.png
#   new file:   doc/tutorials/imgproc/morph_lines_detection/images/linear_vert.png
#   new file:   doc/tutorials/imgproc/morph_lines_detection/images/morph12.gif
#   new file:   doc/tutorials/imgproc/morph_lines_detection/images/morph21.gif
#   new file:   doc/tutorials/imgproc/morph_lines_detection/images/morph211.png
#   new file:   doc/tutorials/imgproc/morph_lines_detection/images/morph6.gif
#   new file:   doc/tutorials/imgproc/morph_lines_detection/images/morph61.png
#   new file:   doc/tutorials/imgproc/morph_lines_detection/images/smooth.png
#   new file:   doc/tutorials/imgproc/morph_lines_detection/images/src.png
#   new file:   doc/tutorials/imgproc/morph_lines_detection/images/vert.png
#   new file:   doc/tutorials/imgproc/morph_lines_detection/morph_lines_detection.md
#   modified:   doc/tutorials/imgproc/table_of_content_imgproc.markdown
#   new file:   samples/cpp/tutorial_code/ImgProc/Morphology_3.cpp
#   new file:   samples/data/notes.png
#

```

- As a final step push these changes to corresponding branch in GitHub by:

```
git push origin <name>
```

or

```
git push origin --all
```

## Opening a Pull Request (PR)

1. Now in order to find out if GitHub got sync'd up with our local environment, open a browser and login into your account in GitHub. If everything is correct, the branch you created for the fix should be visible in the branch setup.

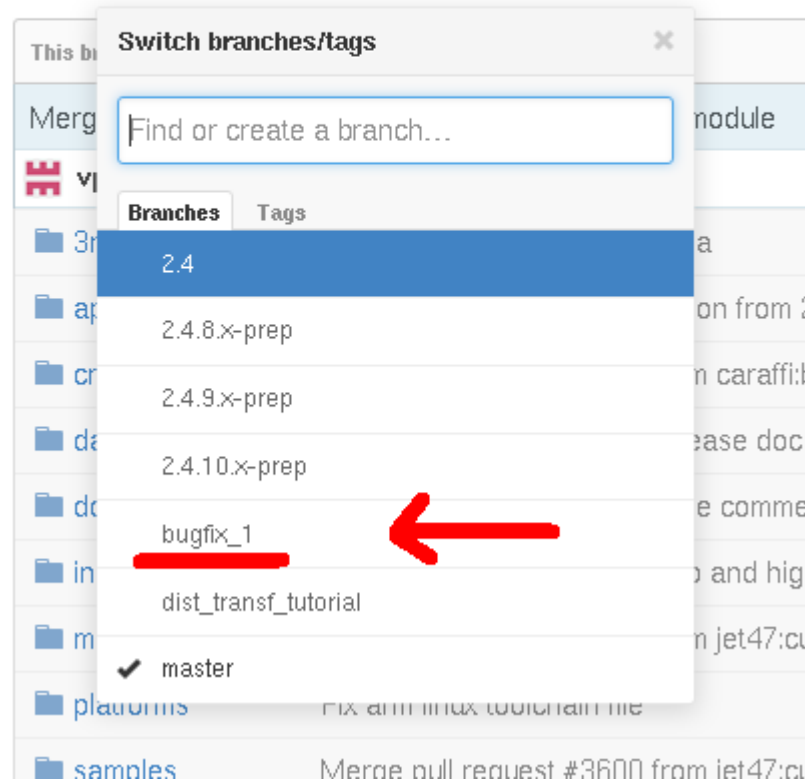
Your recently pushed branches:

 **bugfix\_1** (less than a minute ago)



branch: **master** ▾

**opencv** / +



Select the one you just created. Then browse within to find and check the file. If adaptations worked, you should be able to see them.

This repository Search Explore Gist Blog Help theodr +

theodr / opencv  
forked from Itseez/opencv

branch: bugfix\_1 opencv / apps / traincascade / traincascade.cpp

theodr 12 minutes ago My first commit into bugfix\_1  
6 contributors

126 lines (118 slocc) | 4.278 kb Raw Blame History

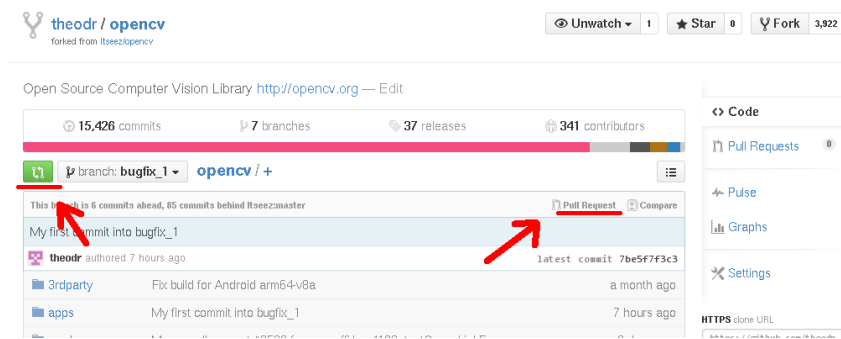
```

1 #include "opencv2/core.hpp"
2 #include "cascadeClassifier.h"
3
4 using namespace std;
5 using namespace cv;
6
7 .....
8 *
9 * Add some lines of comments here for testing and illustration purposes. *
10 * Created by @theodr *
11 *
12 .....
13
14
15 int main( int argc, char* argv[] )
16 {
17     CvCascadeClassifier classifier;
18     string cascadeDirName, vecName, bgName;
19     int numPos = 2000;
20     int numNeg = 1000;

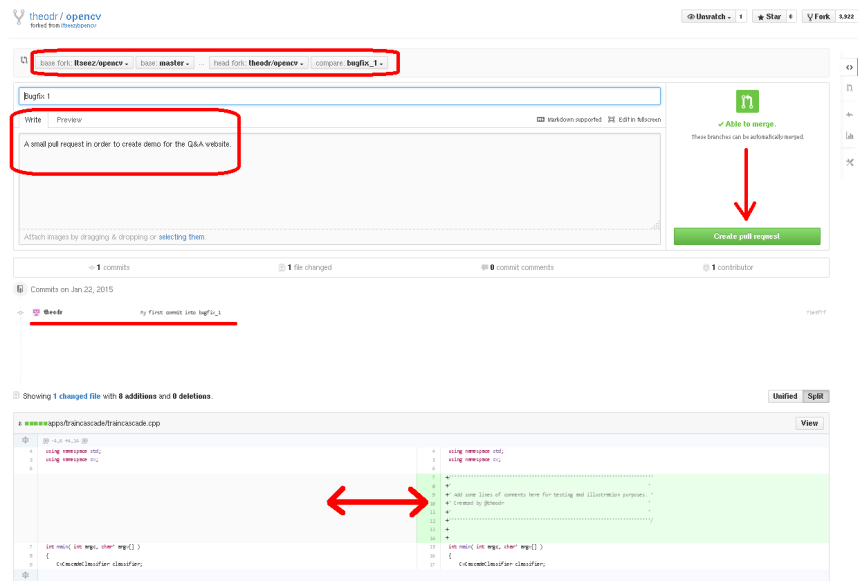
```

perfect!!!!

2. We are ready not to create a **PR**. Just click on the **PR** buttons:



Immediately we can see, that there is a single file found, which is different between the current 'master' branch on *Itseez/opencv* and the 'bugfix\_1' branch on *theodr/opencv*. Moreover, we can see the commit that we did earlier, and the actual differences between the two files. We now assign a pull request at this change (which is basically asking to upload your code to the sourcecode of OpenCV). Click to create a **PR** at the designated button. Do not forget to add some explanation on what your add-on does exactly. Then click the create **PR** button:



This all will result in an actual PR being pushed towards the source code manager. Congratulations!!! Now wait and go check from time to time (most likely you if you have set up everything correct you will also get a mail notification), to make sure the developers don't ask you to change something drastically.

If you want there is also the option to follow the status of your **PR** on the OpenCV's official GitHub repo buildbot. When an error occurs in your build (error build status on **PR**), this will appear on the latter. Shortly:

- Look up in the table your **PR** and take a look at what system builds are failed (i.e. red color)
- Press the build specific tab and open up the report
- Look for the step that is red in the list
- Click on *stdio* if you want to see the exact error output
- Sometimes a report is also generated as a second point containing all error messages separately

Id	Author	Title & description	Build status									
			Linux x64	Win x64	Mac	Android	OpenCL	Linux x64 no optimizations	Docs	Win32	ARM	iOS
3607 master	soyersoyer / mshabunin	Build Mac is too old. Build Linux x64 is too old. Build Android is too old. Build Win x64 is too old. Build OpenCL is too old. Build Docs is too old. Support uncompressed capturing for webcams which support the YUYV pixel format. After the read we can convert to BGR: <code>cvtColor(m.out_CV_YUV2BGR_YUVV);</code>	2263 SUCCESS	1936 SUCCESS	1938 SUCCESS	2161 SUCCESS	1856 SUCCESS	284 SUCCESS	2242 SUCCESS	not_queued	not_queued	not_queued
3603 master	alalek / None	OCL: Shared Virtual Memory (SVM) support  first version of BGM for PR	2335 SUCCESS	2084 SUCCESS	1938 SUCCESS	2250 SUCCESS	1942 SUCCESS	486 SUCCESS	2334 SUCCESS	not_queued	not_queued	not_queued
3601 master	Gillevi / None	Unsuccessful builds Integrating BGM/BinBoost descriptor into OpenCV (the code itself is under opencv_contrib, opencv contains only the tutorial).  The descriptor is based on: 1. Trzcinski, Tomasz, et al. "Learning image descriptors with the boosting trick." Advances in neural information processing systems. 2012. 2. Trzcinski, Tomasz, et al. "Boosting binary keypoint descriptors." Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on. Ieee, 2013.	2246 FAILURE	1980 FAILURE	1932 FAILURE	2133 FAILURE	1841 FAILURE	233 FAILURE	2221 FAILURE	not_queued	not_queued	not_queued

### BuildSlave:

[linux-slave-x64](#)

### Reason:

#3601 (a9f235be6c43ec61c56c59f972faa221576aab46) on precommit\_linuxx64

### Steps and Logfiles:

1. [init init](#) ( 0 secs )
  1. [stdio](#)
2. [Fetch opencv opencv update opencv](#) ( 5 secs )
  1. [stdio](#)
3. [Fetch extra opencv extra update opencv\\_extra](#) ( 4 secs )
  1. [stdio](#)
4. [Merge opencv with test branch merge opencv](#) ( 2 secs )
  1. [stdio](#)
5. [Merge opencv\\_extra with test branch merge opencv\\_extra skipped](#) ( 0 secs )
  1. - no logs -
6. [Fetch opencv\\_contrib opencv\\_contrib update opencv\\_contrib](#) ( 4 secs )
  1. [stdio](#)
7. [Merge opencv\\_contrib with test branch merge opencv\\_contrib](#) ( 4 secs )
  1. [stdio](#)
8. [cmake cmake](#) ( 17 secs )
  1. [stdio](#)
9. [compile release compile release failed](#) ( 2 mins, 14 secs )
  1. [stdio](#)
10. [test\\_core](#)
  1. - no logs -
11. [test\\_imgproc](#)

3. A last step you should consider, is going back to the Issue tracker and add the link to your **PR** at the actual bugfix task. The bug will be then “*automatically*” closed, once your **PR** has been merged (for some reason this is not working at the moment, therefore you will need to do it manually).
4. Once your **PR** got accepted, a button to delete that branch will appear at the end of the **PR** comments. and you can delete your local branch, too. Make sure that you are back to the original branch where your bugfix branch derived from:

`git checkout master (or 2.4)`

and delete it by typing:

```
git branch -D <branch_name> (this applies locally in your computer)
```

```
git push origin --delete <branch_name> (this applies for the forked repository in Github)
```

same procedures applies, if it got rejected or for any reason you do not want to work on this branch/bugfix anymore. The branches in your GitHub account can also be removed online. Just browse to see all your available branches, then find the branch that you are interested in and click to delete it:

The screenshot shows the GitHub repository page for 'theodr / opencv', which is a fork of 'Itseez/opencv'. The repository has 15,505 commits, 6 branches, 37 releases, and 341 contributors. The '6 branches' tab is selected, and a red arrow points to the 'bugfix\_1' branch in the 'Your branches' section. The 'bugfix\_1' branch is highlighted with a red box. Another red arrow points to the 'New pull request' button for 'bugfix\_1'.

Branch	Updated	By	Commits	Files	New pull request	Delete
master	Updated 4 days ago	vpisarev	5810	375	New pull request	Delete
bugfix_1	Updated 4 days ago	theodr	85	5	New pull request	Delete
bugfix_1	Updated 4 days ago	theodr	85	5	New pull request	Delete
2.4	Updated 5 days ago	vpisarev	5810	375	New pull request	Delete
2.4.10.x-prep	Updated 3 months ago	asmorkalov	5810	214	New pull request	Delete
2.4.8.x-prep	Updated 7 months ago	asmorkalov	6366	68	New pull request	Delete
2.4.9.x-prep	Updated 5 months ago	asmorkalov	5970	20	New pull request	Delete

Finally, to update(merge) your local repository, just do a:

```
git pull upstream master (or 2.4)
```

and

```
git push origin --all
```

to keep up to date the fork in GitHub, as well (see next section for more info).

## Keeping your Fork Up-to-Date

1. When creating a fork to the GitHub repository, you fork it as it is on that moment. However, when adding pull requests, on a later moment, you might want to add changes to merged stuff. Then you notice all of a sudden your fork doesn't contain new commits from OpenCV repository, so you need to update your fork.

Browse to the folder where you have cloned the repository:

```
theodore ~ $ cd /home/theodore/documents/git/opencv_source/  
theodore ~/documents/git/opencv_source $ |
```

Make sure that you are on the “*master*” or “*2.4*” branch with the *git status* or *git branch* command, if you are not then switch to it with the *git checkout master (or 2.4)* command. Once you are into the parent branch, create the upstream connection to it in order to be able to retrieve all changes by typing:

```
git remote add --track master upstream https://github.com/opencv/opencv.git  
(for master)
```

```
theodore ~/documents/git/opencv_source $ git remote add --track master upstream https://github.com/Itseez/opencv.git
```

or

```
git remote add --track 2.4 upstream https://github.com/opencv/opencv.git  
(for 2.4)
```

```
theodore ~/documents/git/opencv_source $ git remote add --track 2.4 upstream https://github.com/Itseez/opencv.git
```

2. Retrieving the changes of the remote master to your local master branch is easy by the following command:

```
git pull upstream master
```

```
theodore ~/documents/git/opencv_source $ git pull upstream master  
remote: Counting objects: 735, done.  
remote: Compressing objects: 100% (311/311), done.  
remote: Total 735 (delta 455), reused 574 (delta 402)  
Receiving objects: 100% (735/735), 729.89 KiB | 420.00 KiB/s, done.  
Resolving deltas: 100% (455/455), done.  
From https://github.com/Itseez/opencv  
* branch      master      -> FETCH_HEAD  
* [new branch] master      -> upstream/master  
Updating 95ecd3..3f1fb28  
Fast-forward  
 CMakeLists.txt          | 11 +  
 cmake/OpenCVCompilerOptions.cmake | 25 +
```



Identical for the 2.4 branch of the remote to your local 2.4 branch. Do not forget to first checkout to the 2.4 branch, else this will screw up your master:

```
git checkout 2.4
git pull upstream 2.4
```

```
theodore ~/documents/git/opencv_source $ git checkout 2.4
Switched to branch '2.4'
Your branch is up-to-date with 'origin/2.4'.
theodore ~/documents/git/opencv_source $ git pull upstream 2.4
remote: Counting objects: 7, done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 7 (delta 1), reused 3 (delta 0)
Unpacking objects: 100% (7/7), done.
From https://github.com/Itseez/opencv
* branch          2.4          -> FETCH_HEAD
Updating 2ca39c9..2fabe56
Fast-forward
 samples/gpu/opencv.cpp | 2 --
 1 file changed, 2 deletions(-)
```

3. Finally, what remains is to update your fork. You can accomplish that with the *git push origin -all* command that we have seen earlier:

```
git push origin --all
```

```
theodore ~/documents/git/opencv_source $ git push origin --all
Username for 'https://github.com': theodr
Password for 'https://theodr@github.com':
Counting objects: 742, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (265/265), done.
Writing objects: 100% (742/742), 145.17 KiB | 0 bytes/s, done.
Total 742 (delta 575), reused 620 (delta 455)
To https://github.com/theodr/opencv.git
 2ca39c9..2fabe56  2.4 -> 2.4
 95ecdc3..3f1fb28  master -> master
theodore ~/documents/git/opencv_source $
```