Detailed Example Contribution

Overview

The following is a step-by-step of a contribution. The contribution is a trivial fix to some commenting that is incorrect in the `Notepad_plus_msgs.h` file. It is written from the first person point of view with annotations to explain why I'm doing what I'm doing and what my choices where.

Formatted: Font: (Default) Courier New

The step-by-step begins *after* the set up of the local environment's git configuration, Github account sign-up, and Lighthouse account sign-up, and Visual Studio 2008 Express, and assumes you have ssh validation working for the command line.

Comment [JL1]: Which reminds me: do we have a "get up and running" document for N++ that explains all this?

Setting up the Github and Lighthouse accounts~~.~~

Github repository fork~~.~~

Using Firefox or Chrome (*IE doesn't seem~~s~~ to play well with Github*):

- Log into Github.
- Visit the npp-community page on Github.
- Select the npp-community repository.
- Copy the 'Public Clone URL' to the clipboard.
- Click the `fork` button.

Lighthouse service hook setup~~.~~

- Log into Lighthouse and go to the user profile by clicking the user name.
- In the drop-down menu on the right for `Create an API Token` I selected `all projects` and copied it to the clipboard.
- Go to the Github forked repository page.
- Select `admin` ⟶ `Service Hooks` ⟶ `Lighthouse`
- Enter `nppcommunity` for the *subdomain*.
- Enter `38932` for the *project id*.
- Paste the `Lighthouse Token` into *token*.
- Mark the *Active* box.

The service hook should be all ready to go now.

Getting the source and and setting up the local environment~~.~~

Using msysgit bash prompt:

```
$ cd ~/datastore/repos/git
$ mkdir npp-community && cd npp-community
$ git init
```

This is good time to make sure the Git user.name and user.email values match the ~~GitHub~~Github account value.

```
$ git config --get-regexp "user.*"
#~~//~~ if it isn't what is wanted then...
$ git config user.name 'My Name'
$ git config user.email my@email.com
```

This is also a good time to set some other git config values that are useful.

```
$ git config core.autocrlf true    ~~//~~# convert lf to crlf on file checkout
$ git config core.safecrlf warn    ~~//~~# let me know when line endings are
being converted
$ git config rerere.enabled true    ~~//~~# remember conflict resolutions
$ git config apply.whitespace fix    ~~//~~# fix whitespace errors when applying
patches
```

And back to the step-by-step…

Make sure you have your key loaded into pageant if you use putty;~~,~~ if you use openssh it helps to have ssh-agent setup and running.

```
$ git remote add npp-community git://github.com/npp-community/npp-
community.git  ~~//~~# pasted from clipboard
$ git remote add thell-nppcr git@github.com:almostautomated/npp-community.git
$ git fetch npp-community
$ git checkout -b master npp-community master
$ git submodule init
$ git submodule update
$ git checkout -b next npp-community/next
$ git submodule update      ~~//~~# just to see if the submodule is different on
this branch.
$ git checkout -b pu npp-community/pu
$ git submodule update      ~~//~~# just to see if the submodule is different on
this branch.
$ git checkout -b thell-master Thell-nppcr/master
$ git submodule update      ~~//~~# reset the submodule to whatever it was on
master
```

What I did above was setup my local environment to have master, next, and pu match up with what npp-community has, then setup my own master for *my* working environment. Right now they are the same, but later they may be different because each of us likes things our own way, right? The submodule commands initialize Google Test and Google Mock.

After that, open Visual Studio Express 2008, load and build projects in the following order: (*I used the debug build configuration for each*)

- Google Test (`google_test/msvc/gtest.9.sln`)
- Google Mock (`google_mock/msvc/gmock.9.sln`).
- Notepad++ (`notepadPlus.9.sln` in the root directory).

Everything ~~is~~ should be setup and good to go now.

Making the trivial changes~~.~~

Create the ticket~~.~~

Every contribution should have a ticket:~~.~~

- Log into Lighthouse
- Visit the npp-community project
- Click the `Create New Ticket` button.
- Subject → Notepad_plus_msgs.h: trivial comment fixes
- Body → Some minor cleanup of copied comments.
- Make myself the person responsible.
- Create the ticket.

Make note of the ticket #. In this case it is #11

Create a topic branch for the change~~.~~

```
$ git checkout -b tf/LH-11/trivial-comment-fixes master
$ git push thell-nppcr tf/LH-11/trivial-comment-fixes
```

The naming convention `initials/LH-ticket#/topic` makes working with local branches easy and makes it obvious what the branch is for ~~to~~ other users when you push it to Github. When the branch is merged into the main repository the branch name may be different, but that is OK, because once it is merged into the main repository the personal branch isn't needed anymore and ~~should~~ can be deleted to keep things clean.

Comment [JL2]: Should we strongly suggest a way to separate words? Either with dashes '-', underscores '_' or MixedCaps? I know I must have used them all, but a suggested standard would be nice IMO.

Comment [JL3]: You could indicate how to delete a branch from your own depot and from github (I know I search for it every time I need to delete a branch there).

Pushing the newly created branch before you make any changes to it allows Github to cache the branch so later pushes to it will trigger the service hook.

Make the changes~~.~~

Edit `Notepad_plus_msgs.h`.

Formatted: Font: (Default) Courier New

```
Line: 120
from:         //BOOL WM_SWITCHTOFILE(0, 0)
to:           //NPPM_SAVECURRENTFILE(0,0)

Line: 374
from:         //scnNotification->nmhdr.code = NPPN_FILEBEFOREOPEN;
to:           //scnNotification->nmhdr.code = NPPN_FILEBEFORESAVE;
```

```
Line: 416
from:         //scnNotification->nmhdr.code = NPPN_FILEBEFOREOPEN;
to:           //scnNotification->nmhdr.code = NPPN_FILEBEFORELOAD;
```

Pretty darned trivial isn't it? Rebuild anyway just to make sure I didn't typo and remove a comment indicator!
Everything still looks good, time to commit.

```
$ git status    ~~//~~ # make sure the only file changed is the msgs file
$ git add .
$ git commit -s
```

Added the subject and body~~,~~ then saved the commit.

```
$ git log -1
commit 9033d9d1cf7691ca214a6f9926fa92c9e4bb4e0c
Author: Thell Fowler <git@tbfowler.name>
Date:   Fri Oct 16 18:41:26 2009 -0500

    Notepad_plus_msgs.h: trivial comment corrections

    * Looks like some comment blocks where just copied but not changed.

    [#11 state:needs_ack responsible:npp-community]

    Signed-off-by: Thell Fowler <git@tbfowler.name>
```

Commit messages are an art, and we aren't pedantic about it, just a few things to take notice of:
* The topic has the convention of filename or area changed, a ':', and a *short* description of about 50 chars.
* A blank line.
* Then an explanation of why the change is being made. Keep the lines < 8~~7~~0 chars.
* A blank line.
* Lighthouse ticket info in brackets. This only needs to be on the last commit on the branch being submitted. (*For example a series of 10 commits to accomplish one thing only needs this on the last, but a series of ten that closes three related tickets would have the ticket info in the commit that closes that item.*)
* A blank line.
* Signed-off-by:

Comment [JL4]: Is the format for it explained thoroughly somewhere?

You may also have noticed that this commit will change the state to `needs_ack` directly from `new`. This is common, especially for short trivial updates.

Push the change.

```
git push thell-nppcr tf/LH-11/trivial-comment-fixes
```

Sometimes it might take a few moments for Lighthouse to process the low-priority work queue, but you can either take a look at the page or check your email to verify the ticket updated.

Now wait for someone else to change the state to acked. The reviewer may download your branch, test it out, and amend the last commit by adding and 'Acked-by:' line and changing the ticket state to acked in the commit; or they might just change it on Lighthouse.

Pushing version 2 of the topic branch~~.~~

I cheated for this since it is so trivial and did it myself since no-one else was around, yet when I did the review of it I found an error.~~ :D~~
Looking at the ticket thread you will see an entry from npp-community re-assigning the ticket back to me and leaving a comment on the ~~GitHub~~Github commit stating what was wrong. This gives an ideal chance to explain resubmitting a branch…

Think about this for a moment. If I just make the correction and amend the commit and re-push it then there really isn't any way for someone just looking at my Github repository to know that I am submitting a new version of my topic branch unless they go into the commit and look at the change. So to be kind to others I do the following…

```
$ git checkout -b tf/LH-11v2/trivial-comment-fixes tf/LH-11/trivial-comment-
fixes
$ git push thell-nppcr tf/LH-11v2/trivial-comment-fixes
$ vim PowerEditor/src/MISC/PluginsManager/Notepad_plus_msgs.h
~~//~~ # Fix my spacing error, write and quit
$ git add PowerEditor/src/MISC/PluginsManager/Notepad_plus_msgs.h
$ git commit --amend
$ git push --force thell-nppcr tf/LH-11v2/trivial-comment-fixes
```

Comment [JL5]: git add? Wasn't this file already in the repo?

Essentially all I did was create a new 'v2' branch from the 'v1' branch, push that to Github, make my change, amend the commit, and push it again using the force option. This makes a new version branch *and* updates the Lighthouse ticket.

After waiting again for another ack, it is then time to wait for the update to get merged onto the *pu* branch.

Proposed update branch~~.~~

This part is up to the maintainer, and describes the actions ~~the maintainer~~he takes~~.~~

Formatted: Font: Italic

Formatted: Font: Italic

```
$ git remote add tf-pu git://github.com/almostautomated/npp-community.git
$ git fetch tf-pu
$ git checkout -b tf/LH-11v2/npp-trivial-comment-fixes tf-pu/tf/LH-
11v2/trivial-commit-fixes
$ git log master..     ~~//~~ # Verify that the only changes between the branches
are limited to the change ticket subject.
$ git log -p     ~~//~~ # Verify whitespace errors are not being introduced
~~//~~ #  do build test
$ git checkout pu
$ git merge --log --no-ff tf/LH-11v2/npp-trivial-comment-fixes
$ git commit --amend
~~//~~ # add lighthouse ticket state update
$ git push npp-community pu
```

Now your update is on the *pu* branch.

Moving on to the *next* branch~~.~~

After being tested and getting feedback from the community a milestone will be set. The ticket will also get merged into the *next* branch.

This part is up to the maintainer, and describes the actions ~~the maintainer~~ he takes~~. W~~when moving topics from* pu *to next

```
$ git checkout next
$ git log --first-parent next..pu ~~//~~ # get a list of the branches merged to
pu

~~//~~ # For each branch in that list that is ready for the next branch
$ git merge --log --no-ff $(topic_branch)  ~~//~~ # relying on rerere to resolve
conflicts
$ git commit --amend
~~//~~ # add lighthouse ticket state updates for newly moved branches
$ git checkout~~o~~ pu
$ git reset --hard next
~~//~~ # re-merge the branches that stayed on pu
$ git merge --log --no-ff $(topic_branch)
$ git push npp-community next
$ git push --force npp-community pu
```

Moving on to the *master* branch~~.~~

As the update sits on the *next* branch the project continues to move forward to the next milestone, when the time comes the update is merged on to master and the ticket is marked as resolved.

***This part is up to the maintainer, and describes the actions ~~the maintainer~~he takes~~. W~~when moving topics from* next *to* master**

```
$ git checkout master
$ gitk --all    ~~//~~ #  I like to get a visual feel for what needs to be done.
$ git branch --no-merged

~~//~~ # For each branch in that list that is ready for the master branch
$ git merge --log --no-ff $(topic_branch)  ~~//~~ # relying on rerere to resolve
conflicts
$ git commit --amend
~~//~~ # add lighthouse ticket state updates for newly moved branches

$ git checkout~~o~~ next && git merge --log --no-ff master
~~//~~ # Check to see if any branches on pu are ready to be integrated into next
```

Formatted: Font: Italic

Formatted: Font: Italic

Formatted: Font: Italic

Formatted: Font: Italic

Formatted: Font: Italic

Comment [JL6]: The user most probably won't have the co alias installed. However it would be nice to give the set of aliases and settings you gave me in this document. I think this would sit nicely with the part where you mention to setup the user name and email address.

Formatted: Font: Italic

Formatted: Font: Italic

Formatted: Font: Italic

Formatted: Font: Italic

Formatted: Font: Italic

Formatted: Font: Italic

~~//~~ # if so the same steps as outline in the 'Moving to next' section are followed.

$ git checkout~~o~~ pu
$ git reset --hard next
~~//~~ # re-merge the branches that stayed on pu
$ git merge --log --no-ff *$(topic_branch)*
$ git push npp-community next
$ git push --force npp-community pu

Formatted: Font: Italic