Submitting Pull Requests

Getting involved is easy! Simply send an email to the dev@ mailing list introducing yourself, and the people in the community can help you get familiar with Apache Mynewt OS.

Patches

Apache Mynewt can accept patches as proposed in Apache's workflow here. Basically, you submit your initial edits for your peers to review and merge. After a few successful patches under your belt, you will get access to commit code directly to the module you are working on in the repository. And over time, you will get full access to all the repositories, where you can commit code to any module on the apache git repository (and not the github mirror).

Pull Requests

Apache Mynewt will also accept Pull Requests (PR) against its github repositories. We recommend the route of the Pull Request. The steps are given below, followed by an explanatory diagram of how the remote and local repositories interact. The "master" branch is the main branch for code development and "feature" branches handle code for long-lived features that several people in the community collaborate on. An example of a feature branch is "bluetooth5". The master branch is kept fairly up-to-date by merging commits periodically from such "feature" branches into "master". Small changes or edits that are not related to any of these features being tracked in "feature" branches can be merged into the "master" directly without requiring the creation of a "feature" branch.

There is typically an owner of a feature branch. The owner is generally the one to ensure that the master gets periodically merged into the feature branch. Others working on a feature branch can also pull and rebase from master to sync up.

Thus, the "master" is the branch on Mynewt's repository that contains the most recent changes made by the community of developers. Contributions from you need to go into the "master" branch if it is a small change or a change that cuts across multiple features. Otherwise, you issue your PR against a specific "feature" branch. The essential steps are the following:

1. Fork the Mynewt repository to create your very own repo on github.
2. Clone the Mynewt repository on your machine using the Newt tool. Set up the remotes correctly as shown in the diagram. This is key to making sure you correctly pull the latest code from Mynewt's "master" and push to your working branch on github.

```
$ newt new devproject
$ cd devproject
$ newt install
$ cd repos/apache-mynewt-core
$ git status
On branch mynewt_1_0_0_tag
nothing to commit, working tree clean
$ git checkout -b mybranch
Switched to a new branch 'mybranch'
$ git remote -v
origin        https://github.com/apache/incubator-mynewt-core.git (fetch)
origin        https://github.com/apache/incubator-mynewt-core.git (push)
$ git remote add fork https://github.com/<user>/incubator-mynewt-core
$ git remote -v
origin https://github.com/apache/incubator-mynewt-core.git (fetch)
origin https://github.com/apache/incubator-mynewt-core.git (push)
fork https://github.com/<user>/incubator-mynewt-core (fetch)
fork https://github.com/<user>/incubator-mynewt-core (push)
$ git pull --rebase origin master
```

If you are working on a long-term feature (and hence using the code in a "feature" branch), then you should create your branch and pull the latest code from that branch instead. Note lines 8 and 20 below:

```
$ newt new devproject
$ cd devproject
$ newt install
$ cd repos/apache-mynewt-core
$ git status
On branch mynewt_1_0_0_tag
nothing to commit, working tree clean
$ git checkout <feature-branch-name>
$ git checkout -b mybranch
Switched to a new branch 'mybranch'
$ git remote -v
origin        https://github.com/apache/incubator-mynewt-core.git (fetch)
origin        https://github.com/apache/incubator-mynewt-core.git (push)
$ git remote add fork https://github.com/<user>/incubator-mynewt-core
$ git remote -v
origin https://github.com/apache/incubator-mynewt-core.git (fetch)
origin https://github.com/apache/incubator-mynewt-core.git (push)
fork https://github.com/<user>/incubator-mynewt-core (fetch)
fork https://github.com/<user>/incubator-mynewt-core (push)
$ git pull --rebase origin <feature-branch-name>
```

3. Make your code changes and commit them locally. Do all this in your local branch ("mybranch" in this example).

```
$ git checkout mybranch
<make your code changes>
$ git add .
$ git commit –m “your message about your code changes”
```

4. Always sync up with the latest code in the "master" branch and resolve any merge conflicts before pushing any changes to remotes. Again, if you are working on a feature branch, always sync up with the "feature" branch.

```
$ git pull --rebase origin master

OR

$ git pull --rebase origin <feature-branch-name>
```

5. Push your commits to your remote working branch on github.

```
$ git push fork mybranch
```

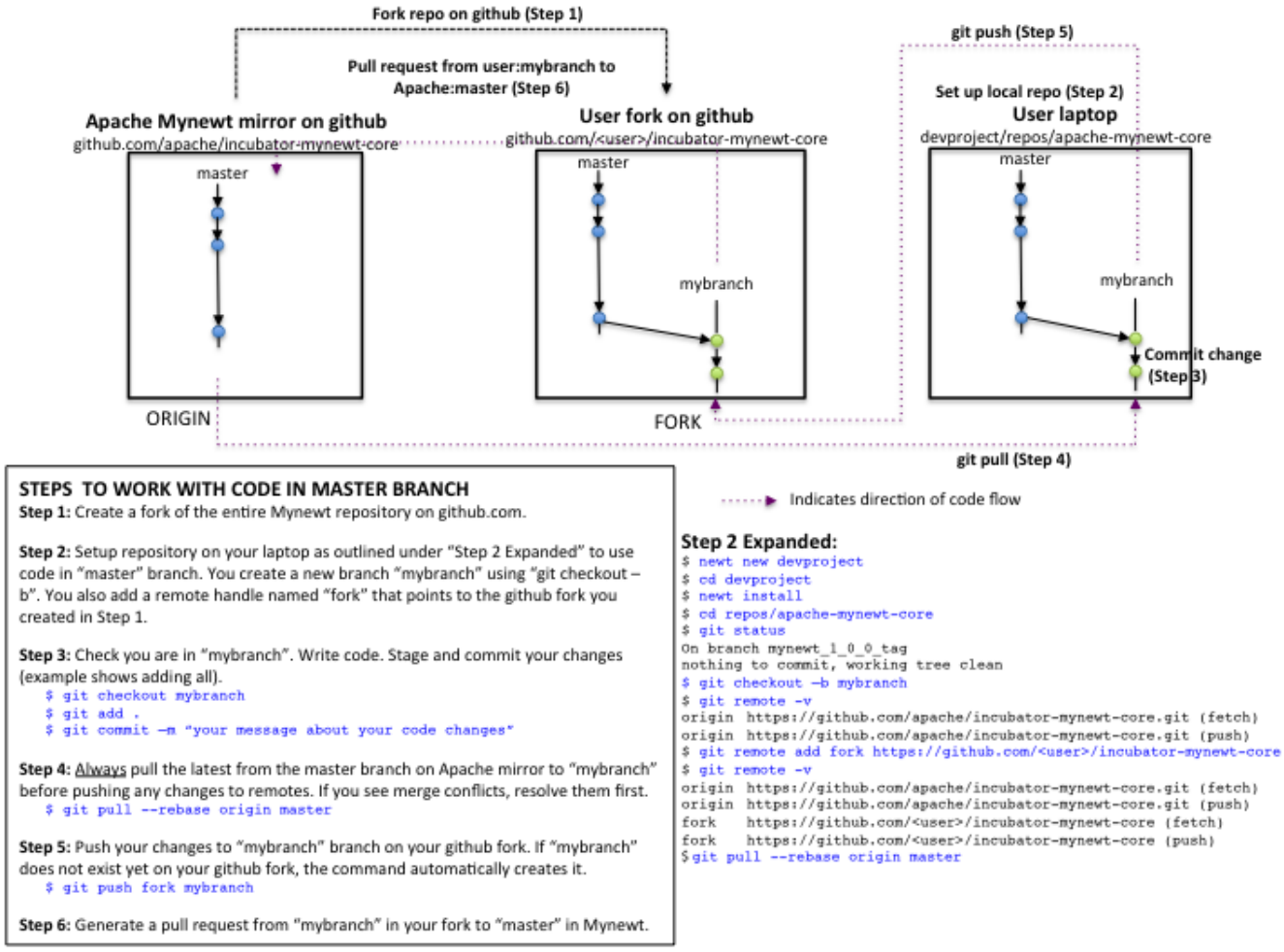
6. a) Submit a pull request against the "master" branch of the appropriate Apache Mynewt repository (incubator-mynewt-core if you are working on the Mynewt OS codebase, incubator-mynewt-newt if you are working on the Newt Tool) by clicking the green "New pull request" button on your github fork page.

   OR

   b) Submit a pull request against the appropriate "feature" branch if your changes are for the functionality in the "feature" branch.

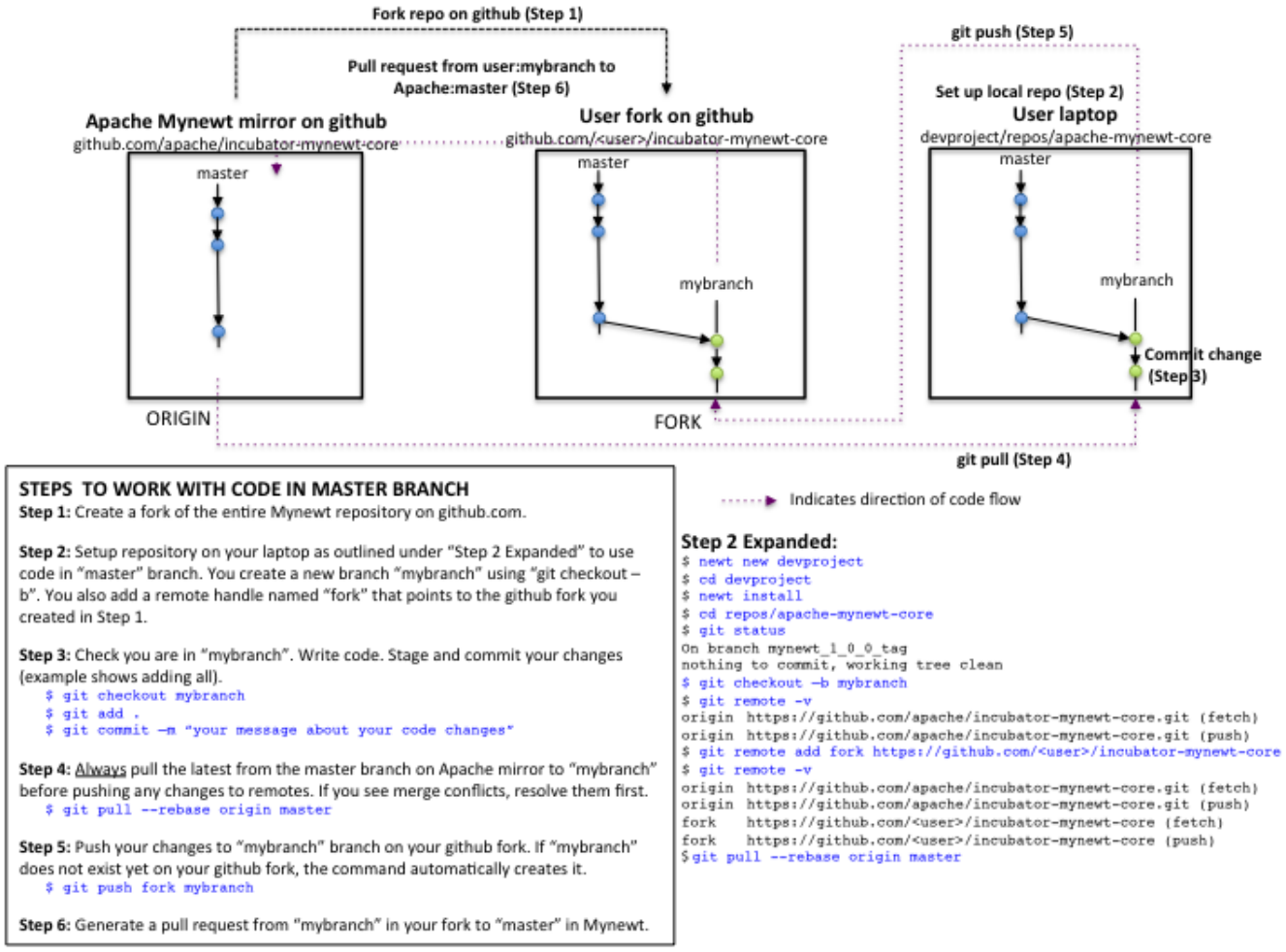
Explanatory diagram

If you are working with the master branch:



Here is the PDF of the diagram: mynewt master cycle recommended.pdf

If you are working with a feature branch:



Here is the PDF of the diagram: mynewt feature cycle recommended.pdf

Notes

Remember there are four repositories to work with in the Apache Mynewt project. Only the documentation repo has a "develop" branch to issue all technical documentation Pull Requests against.

- Apache Mynewt core repo mirrored on github
- Apache Mynewt newt tool repo mirrored on github
- Apache Mynewt newtmgr tool repo mirrored on github
- Apache Mynewt documentation repo mirrored on github

The bottomline is to work with the "master' branch for small changes that is applicable to the OS in general and to work with "feature" branches for changes specific to that area of code. In the comment for the pull request, include a description of the changes you have made and why. Github will automatically notify everyone on the commits@mynewt.incubator.apache.org mailing list about the newly opened pull requests. You can open a pull request even if you don't think the code is ready for merging but want some discussion on the matter.

There is no need to open a new pull request if you have already submitted a pull request against a branch on the mirror but have modified your code futher (e.g. after some feedback from the community or another clever idea popping into your head). The old pull request will get updated with your changes.

Upon receiving notification, one or more committers will review your work, ask for edits or clarifications, and merge when your proposed changes are ready.