Merging Github Pull Requests

This section documents the process of merging code changes contributed via Github Pull Requests. It assumes you have a clone of ZooKeeper's Git repository and the user has committer access rights.

zk-merge-pr.py is a script that automates the process of accepting a code change into the project. It directly squashes the commits in the pull request, rewrites the commit message in the squashed commit to follow a standard format including information about each original commit. The script sends a Github API request to squash and merge the pull request, automatically closing the corresponding PR. Note that the script will prompt the user before executing remote updates (i.e., git push and closing JIRA ticket), allowing flexibility for users who wish to skip those steps.

This script is a modified version of an Apache Kafka tool: https://github.com/apache/kafka/blob/trunk/kafka-merge-pr.py that, in its turn, is a modified version of an Apache Spark tool: https://github.com/apache/spark/blob/master/dev/merge_spark_pr.py

Step-by-step guide

Setting Up:

1. Add aliases for the remotes expected by the merge script (if you haven't already):

   ```
   $ cd $ZOOKEPER_BASE_DIR
   $ git remote add apache-github https://github.com/apache/zookeeper.git
   $ git remote add apache https://gitbox.apache.org/repos/asf/zookeeper.git
   ```

   You can easily checkout pull requests using the following. Here I'm fetching pull #9 and creating a local "pr/9" branch.

   ```
   $ git fetch apache-github  pull/9/head:pr/9
   ```

   However you can also (instead of the fetch above - this part is optional) add the following refspec as a shortcut

   ```
   $ git config --add remote.apache-github.fetch '+refs/pull/*/head:refs/remotes/apache-github/pr/*'
   ```

   after which:

   ```
   $ git fetch apache-github
   ```

   and then just:

   ```
   $ git checkout pr/9
   ```

2. Before starting using the script it’s required to setup environment variables below:

   PR_REMOTE_NAME - points to Github **mirror** of Apache project (default git-remote name: apache-github)

   PUSH_REMOTE_NAME - points to Apache Git repo (default git-remote name: apache)

   ```
   $ export PR_REMOTE_NAME=apache-github
   $ export PUSH_REMOTE_NAME=apache
   ```

3. Install jira-python and requests:

   ```
   sudo easy_install jira
   sudo easy_install requests
   ```

   Or

   ```
   sudo pip install jira
   sudo pip install requests
   ```

4. Setup environment variables to JIRA credentials:

JIRA_USERNAME & JIRA_PASSWORD - apache JIRA credentials

```
$ export JIRA_USERNAME=myname
$ export JIRA_PASSWORD=mypassword
```

OR

Use JIRA_ACCESS_TOKEN which is recommended over exposing your Jira password.

Go to https://issues.apache.org/jira/secure/ViewProfile.jspa -> Personal Access Tokens for

```
export JIRA_ACCESS_TOKEN=myaccesstoken
```

If you don't execute steps 3 and 4 then the script will not be able to automatically close the JIRA after merging the PR.

5. (Optional) Setup Github OAUTH token:

   GITHUB_OAUTH_KEY (optional) - if you exceed Github API rate limit then set this variable to allow it to surpass this limit as the script comment states:

   ```
   $ export GITHUB_OAUTH_KEY=<your-github-oauth-key>
   ```

   “OAuth key used for issuing requests against the GitHub API. If this is not defined, then requests will be unauthenticated. You should only need to configure this if you find yourself regularly exceeding your IP's unauthenticated request rate limit. You can create an OAuth key at https://github.com/settings/tokens. This script only requires the "public_repo" scope.”

Once the pull request is ready to be merged (it has been reviewed, feedback has been addressed, CI build has been successful and the branch merges cleanly into trunk):

1. Run the merge script:

   ```
   python zk-merge-pr.py
   ```

2. Answer the questions prompted by the script.
   When the script asks “Would you like to squash the commit messages? (y/n): ”, answer n.

   If there is any error during cherry-picking then the process can be aborted without running the clean up routine. In this case, make sure you delete the PR_TOOL_* branches before re-running the script.

Under the hood

1. Merging and/or cherry-picking
   a. If the Github PR has already been merged into Apache Git repo, the script will ask if you want to cherry-pick the PR commits to other branches (to backport the changes to other branches, for example). ***After we are done cherry-picking (the script asks if we want to continue), no further steps from this script are executed.***

   b. Otherwise, the script will try to merge the PR into the target reference (if the PR is targeting the master branch it will try to merge the PR on master, if targeting branch-3.5 it will try to merge on branch-3.5, etc). It will ask for authors and reviewers names/emails, among other bits of info that compose the commit log.

      i. After executing step b, the script will ask if you want to backport (i.e., cherry-pick) the PR into other branches, suggesting the latest branch it has already found (say, branch-3.6), but you are free to choose other branches. After each cherry-pick it will ask if you want to backport the PR to another branch until you choose not to (in a nutshell, the same procedure as step b).

c. During this step, the script will update the JIRA entry, marking it as closed. It will try to fill the fixed versions accordingly. The committer should provide the JIRA ID. *If the JIRA credentials are not set up or jira-python lib is not installed*, the script will merge the PR and push to Apache git repo, but won't close the JIRA issue, so make sure the correspondent JIRA issue was closed after merging the PR.

   It sends an API request to squash and merge the pull request.

1. Answering Squashing Question:

   Respond to the question 'Would you like to squash the commit messages? (y/n): ' by typing 'n' as suggested.

2. Reviewing Checks:

   Check for disapproval reviews and prompt confirmation if changes are requested. Also, verify if the PR has approved reviews; if not, prompt confirmation for the merge.

3. Author and Closing Line:

   Gather information about the PR author and create a closing line for the commit message.

4. Checking GitHub Checks:

   Check the status of GitHub checks for the latest commit. If all checks pass, proceed; otherwise, prompt confirmation if there are pending or unsuccessful checks.

5. GitHub API Request - Merge:

   Send a request to the GitHub API to merge the PR using the specified squash method, including the commit title and modified commit message.

6. Handling API Response:

   If the merge is successful, print the merged PR information and return the merge commit SHA. If unsuccessful, print an error message, details, and exit the script.

Troubleshooting

Error

```
urllib2.URLError: <urlopen error [SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed (_ssl.c:727)>
```

This happened either with a new OSx update, or during OSx python upgrade. My certificates were all gone. The fix is to install the certs manually:

Fix

```
/Applications/Python\ 2.7/Install\ Certificates.command
```

Related articles

- Merging Github Pull Requests