Code Submission Process - RDK Central GitHub

Introduction

GitHub is a Git repository hosting service, but it adds many of its own features. While Git is a command line tool, GitHub provides a Web-based graphical interface.

GitHub Enterprise is the on-premises version of GitHub.com and is available on VMware, AWS, and OpenStack KVM, on your own servers or in a private cloud. GitHub Enterprise operates on your infrastructure with your existing information security controls from firewalls and VPNs, to IAM and monitoring systems.

CMF GitHub Organisations

There are 2 RDKM Code Management organisations, namely RDKcentral and RDKcmf. The latter organisation primarily hosts open source mirrors of projects that reside on CMF Gerrit. These projects pertain to RDK-V, RDK-B and RDK-C profiles.

- https://github.com/rdkcmf/
- https://github.com/rdkcentral/

GitHub Repositories on RDKcmf organisation

There are approximately 145 mirrors on the GitHub RDKcmf organisation, these are primarily mirrors of Gerrit opensourced repositories. Gerrit repository active branches are replicated to GitHub. Pull Requests are not supported on these repositories as indicated in the repository CONTRIBUTING.md files.

RDK Central GitHub Components & its Product Branches

Please refer to this link to see all the repositories Source Code Repositories

Component	Product Branch	License	Description
RDKServices	current default branch	Apache	
Lightning	master	Apache	
Lightning-CLI	master	Apache	
Lightning-SDK	master	Apache	
Lightning-UI-Components	master	Apache	
OCDM-Clearkey	master	Apache	
OCDM-Nagra	master	Apache	
OCDM-Playready	master	Apache	
OCDM-Widevine	master	Apache	
RDKShell	master	Apache	
RDKSplashScreen	master	Apache	
Thunder	master	Apache	
ThunderJS	master	Apache	

ThunderNanoServices	master	Apache	
ThunderUI	master	Apache	
Dobby	master	Apache	
android-remote	master	Apache	
meta-turris	master	Apache	
opensync-vendor-rdk-turris	master	Apache	
rdkb-turris-hal	master	Apache	
rdkcryptoapi	master	Apache	
sample-licensing	master	Apache	

GitHub Pull Requests

Pull requests let you tell others about changes you've pushed to a branch in a repository on GitHub. Once a pull request is opened, you can review the potential changes with collaborators and add follow-up commits before your changes are merged into the base branch. Anyone with read permissions to a repository can create a pull request, but you must have write permissions to create a branch. If you want to create a new branch for your pull request and don't have write permissions to the repository, you can fork the repository first. Pull requests can only be opened between two branches that are different.

GitHub Fork

A ‘fork’ is a personal copy of another user's repository that lives on your GitHub account. Forks allow you to freely make changes to a project without affecting the original. A forked project also remains attached to the original, allowing you to submit a pull request to the original's author to update with your changes, ensuring you’re always working off a recent or up-to-date codebase.

GitHub Workflow Steps

- Create a Fork by simply clicking on the 'fork' button of the repository page on GitHub.
- Clone your Fork, the clone command creates a local git repository from your remote fork on GitHub.
    - git clone https://github.com/USERNAME/REPOSITORY.git
- Modify the Code in your local clone, and commit the changes to your local clone using the git commit command.
- Push your Changes by invoking the git push command, from your workspace, to upload your changes to your remote fork on GitHub.
- Create a Pull Request by clicking the 'pull request' button on the GitHub page of your remote fork.

Configure your Github access token

In the recent past support for direct password authentication was removed from Github. You will need to generate a Github personal token to push your code changes RDK Central Github.

To create your personal token, you have to go to github.com -> Settings -> Developer Settings -> Personal Access Token -> Generate New Token.

Note - While creating a new token, it will ask for Github configuration options selection – Select everything.

Once the Github token is generated successfull, you will need to add an entry an entry to the ~/.netrc file OR you can directly use this token as your Github password in the command line to push the code changes.

Example: how to add Github credential on ~/.netrc file
machine github.com login your-github-handle-name password ghp_BCy09kNYxg82no6OnliSJQVngGi9K1234567

GitHub Protected Branches

Protected branches ensure that collaborators on your repository cannot make irrevocable changes to branches. Enabling protected branches also allows you to enable other optional checks and requirements, like required status checks and required reviews.

A custom CMF branch protection scheme is deployed in each repository in order to enforce the desired workflows. This scheme imposes the following rules:

- Require pull request reviews before merging
- Require status checks to pass before merging
    - blackduck
    - copyright
    - license/cla
    - component-build

Required status checks ensure that all required CI tests are passing before collaborators can make changes to a protected branch. Status checks are based on external processes, such as continuous integration builds, code compliance scanning, which run for each push you make to a repository. You can see the pending, passing, or failing state of status checks next to individual commits in your pull request.

Contributor License Agreement (CLA)

The RDK CLA facilitates the acceptance and sharing of RDK contributions within the community.

When you contribute to an RDK open source project on GitHub via a new pull request, a bot will evaluate whether you have signed the CLA. The bot will comment on the pull request, including a link to accept the agreement.

CLA assistant enables contributors to sign CLAs from within a pull request. The CLA is stored as a GitHub Gist file and linked with the repository /organisation in CLA assistant.

CLA assistant:

- Comments on each opened pull request to ask the contributor to sign the CLA
- Allows contributors to sign a CLA from within a pull request
- Authenticates the signee with his or her GitHub account
- Updates the status of a pull request when the contributor agrees to the CLA
- Automatically asks users to re-sign the CLA for each new pull request in the event the associated Gist & CLA has changed
- Repository owners can review a list of users who signed the CLA for each version of it.

Note - CLA assistant is provided by SAP as a free hosted offering under: https://cla-assistant.io/

Compliance Scanning

CMF uses BlackDuck (Protex) to check incoming contributions for license compliance. BlackDuck is normally a very manual tool but a significant level of automation has been developed by the team to reduce manual intervention, but it still requires a human to oversee it.

Compliance scanning is looking for several things:

- Addition of source code with a conflicting license (e.g. LGPL code in an Apache component).
- Modification of an opensource component with code of a conflicting license.
- Incorrect or proprietary copyright attribution.

The key points are as follows:

- Scan contribution in GitHub Pull Request.
- Scan is automatically triggered by a webhook.
- Results in scan of the contribution only, i.e. only the changes.
- Required Status Check, associated with the Blackduck scan is updated.
- Summary of the scan including any code matches are provided via a link to an associated Gist.

OSS Engineer and interested parties are notified of scan failures (violations, pending identifications or reviews etc) via AWS Mailing list and Slack.

git-secrets Scanning

git-secrets is a tool created by AWS Labs that scans commits and commit messages and aims to prevent passwords and other sensitive information being committed to a git repository. The tool can also scan files or folders to look for secrets such as an AWS Access Key ID and AWS Secret Access Keys in a repository. git-secrets scans commits, commit messages, and merge commits to prevent adding secrets into your git repositories. If a commit, commit message, or any commit in merge history matches one of the configured prohibited regular expression patterns, then the commit is rejected.

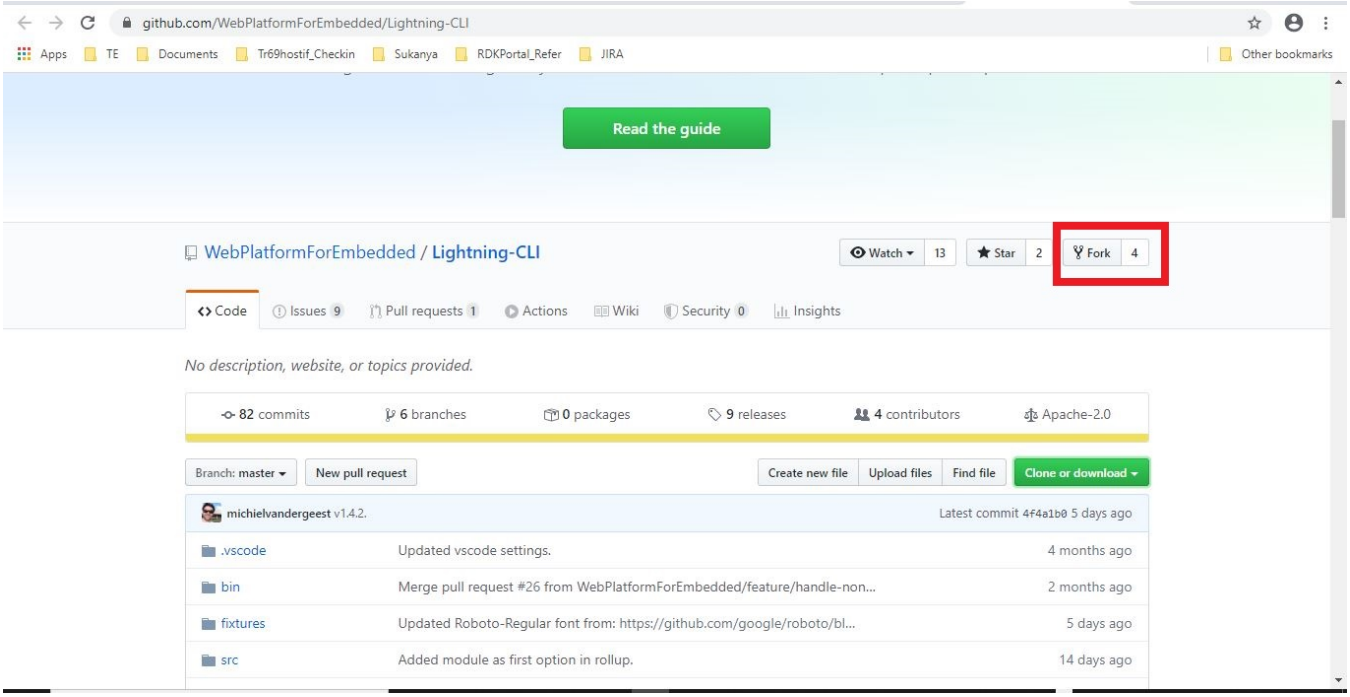
Example of how to push the code changes

Step 1 : Fork the component from GitHub

Sign-in to github with your own credentials.

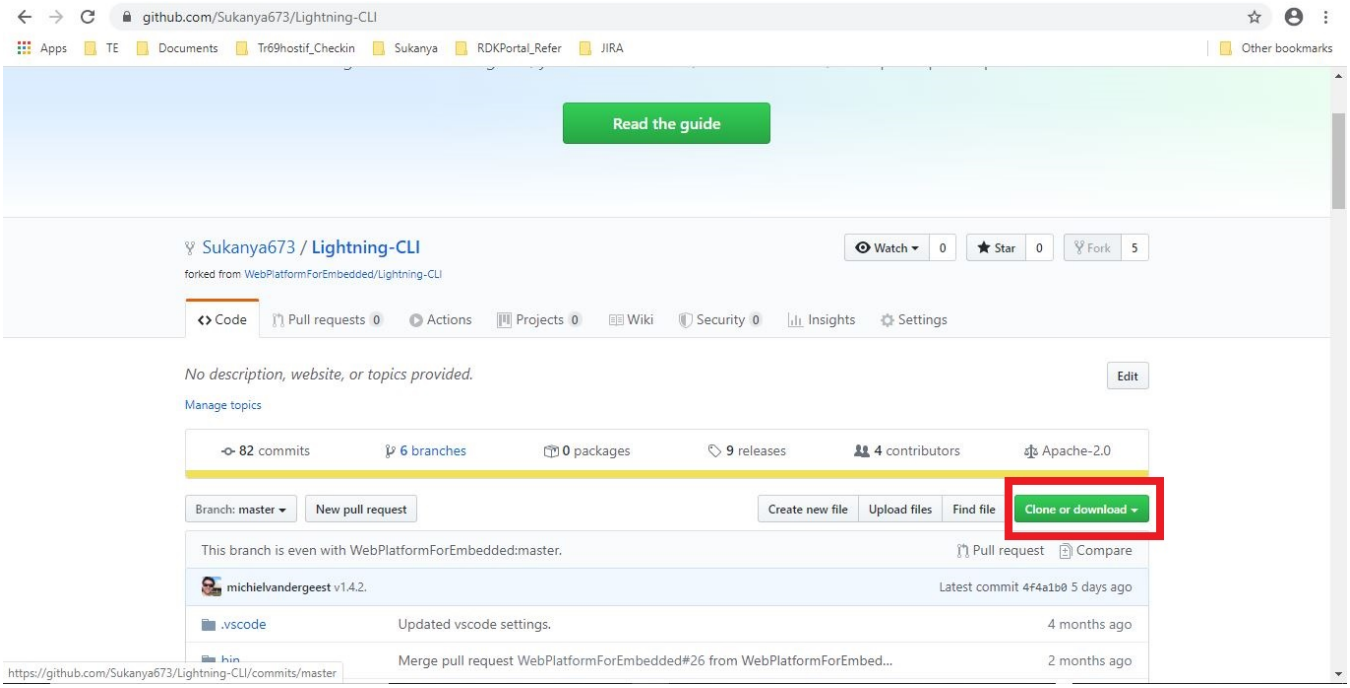
Search for the Component.

Fork the component from github. Forking will create a copy(i.e, your own WORKSPACE) of an original component to work.



Step 2 : Clone the Component

Click on the "Clone or download" button to get the clone url from github. Ensure your github username present in the url to start work with your own workspace.



```
sukanya@tellinux025:~/lightning_cl$ git clone https://github.com/Sukanya673/Lightning-CLI.git .
Cloning into '.'...
remote: Enumerating objects: 152, done.
remote: Counting objects: 100% (152/152), done.
remote: Compressing objects: 100% (88/88), done.
remote: Total 704 (delta 74), reused 120 (delta 59), pack-reused 552
Receiving objects: 100% (704/704), 718.53 KiB | 955.00 KiB/s, done.
Resolving deltas: 100% (369/369), done.
sukanya@tellinux025:~/lightning_cl$
```

Step 3 : Work on changes and Gerrit commands to push the changes

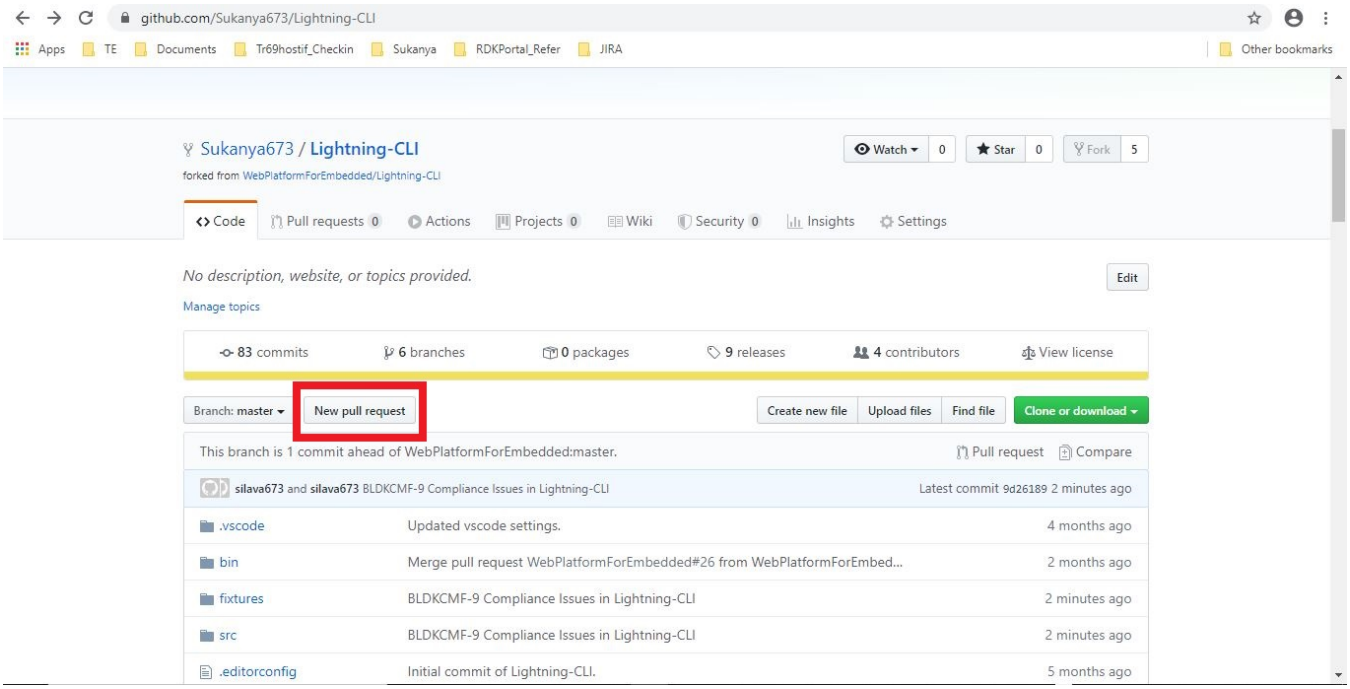
Make the code changes, and commit the changes

```
$ git add .
$ git status
$ git commit -m "<JIRA_TICKET_ID> <COMMIT_DESCRIPTION>"
$ git push
```

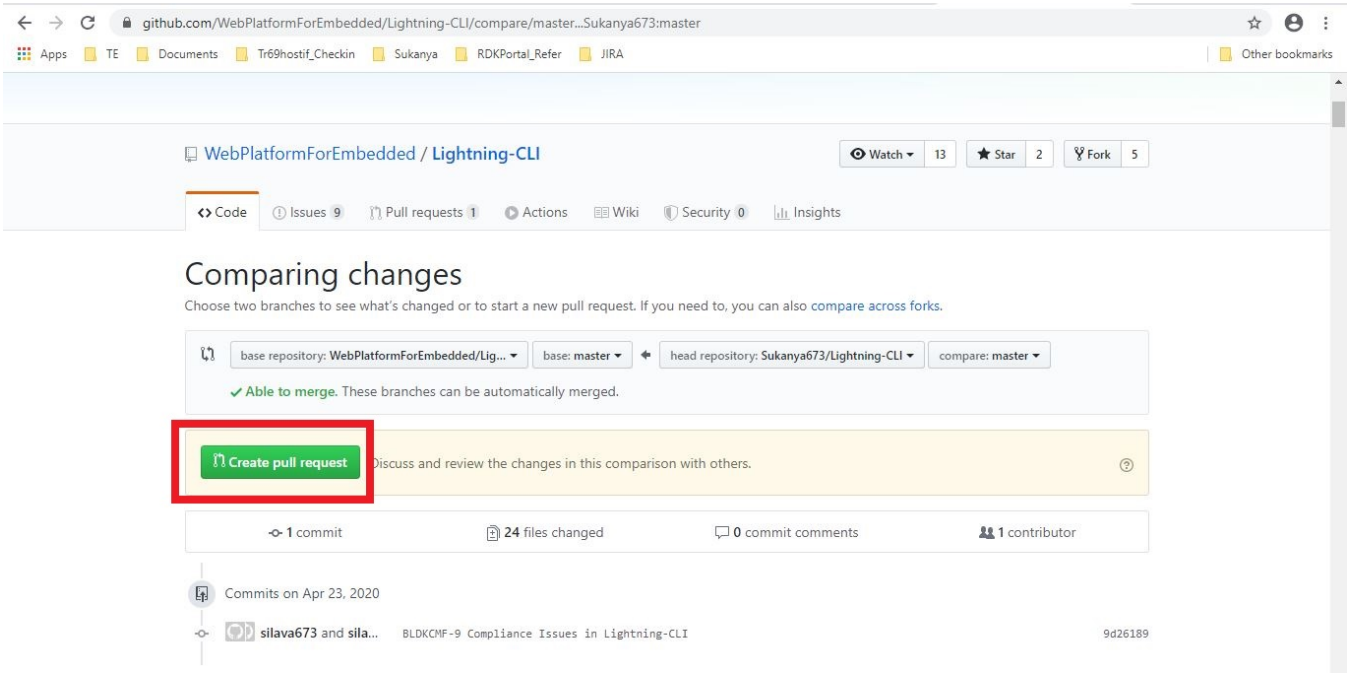
Step 4 : Create pull request for review the changes

Once submitted the changes need to create pull request from github for review. **Pull requests** let you tell others about changes you've pushed to a branch in a repository on GitHub. Once a **pull request** is opened, you can discuss and review the potential changes with collaborators and add follow-up commits before your changes are merged into the base branch.

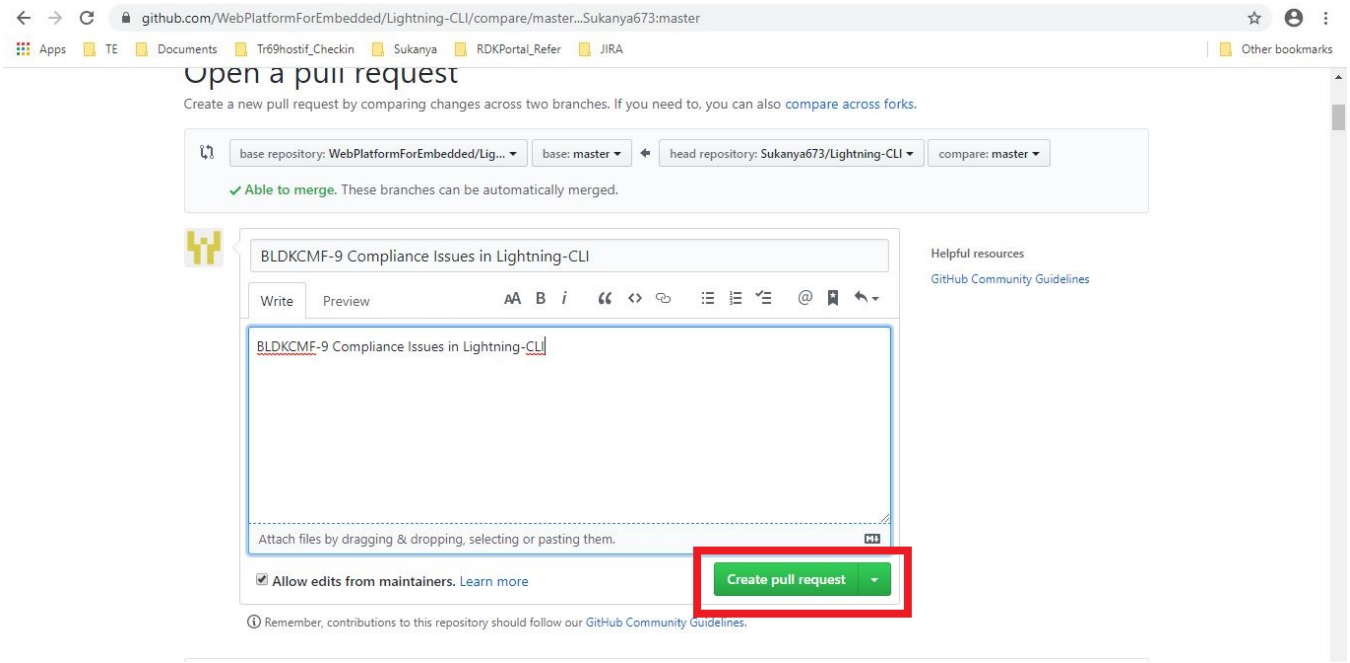
- Click on "New pull request" button.



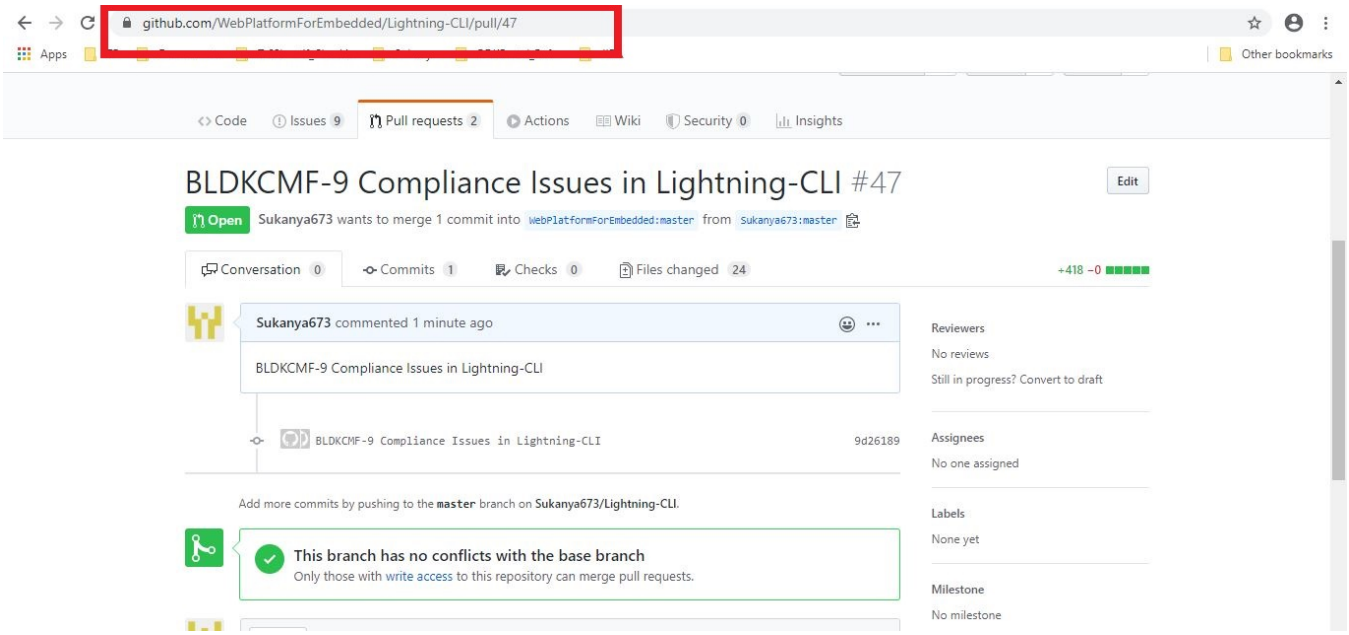
- Click on "Create pull request" button.



- Provide the commit request and Click on "Create pull request" button.



Pull Request page will be created.



Step 5 : Working on review comments

Once you've created a pull request, you can push commits from your workspace to add them to your existing pull request. These commits will appear in chronological order within your pull request and the changes will be visible in the "Files changed" tab.

Other contributors can review your proposed changes, add review comments, contribute to the pull request discussion, and even add commits to the pull request.