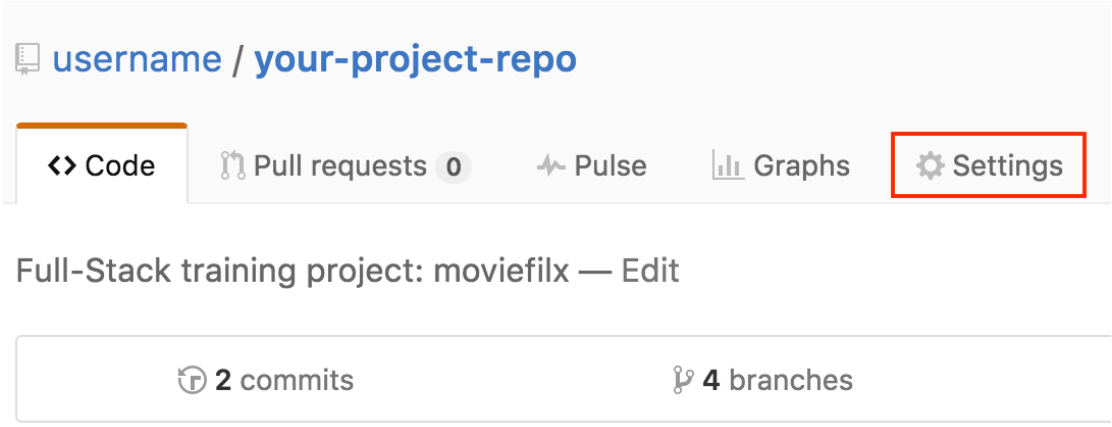
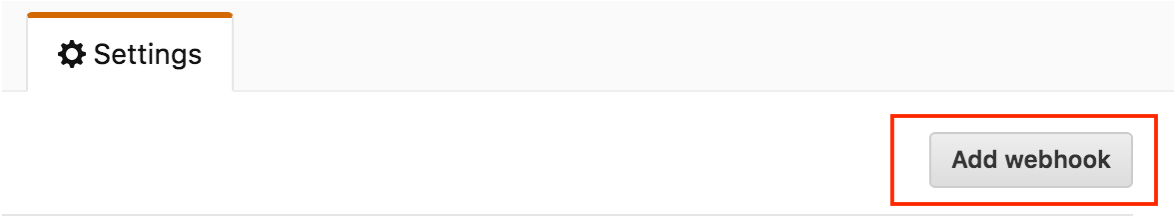
Project Submission Workflow via GitHub

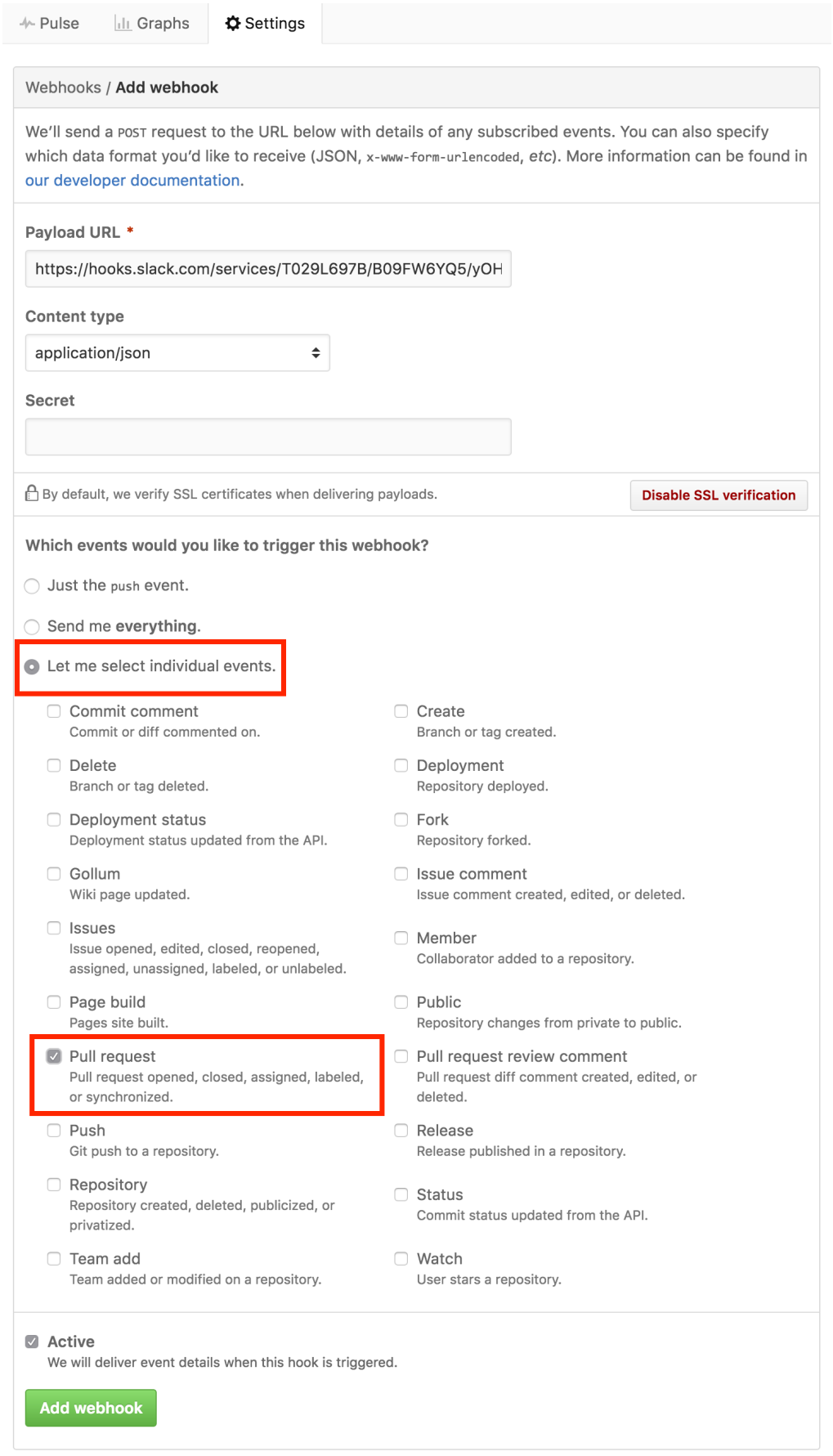
STEP 1 In your GitHub account, go to the repository that you have created for the project. Click on the **Settings** tab in the top navigation.



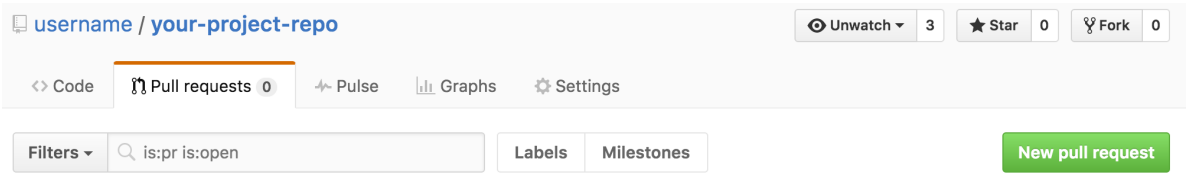
STEP 2 Click on **Webhooks & Services** in the left navigation, and then press the **Add webhook** button.



STEP 3 Enter following URL in the **Payload URL** field:
https://hooks.slack.com/services/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
Ensure that the **Content type** is **application/json** and that **SSL verfication** is enabled.
Click on the radio button for **Let me select individual events** and check **Pull Request** option. Make sure only the **Pull Request** event is selected.
Select **Active** if not selected already.
Press the **Add webhook** button when done.
A few seconds later, GitHub will ping this external webhook and enable it.

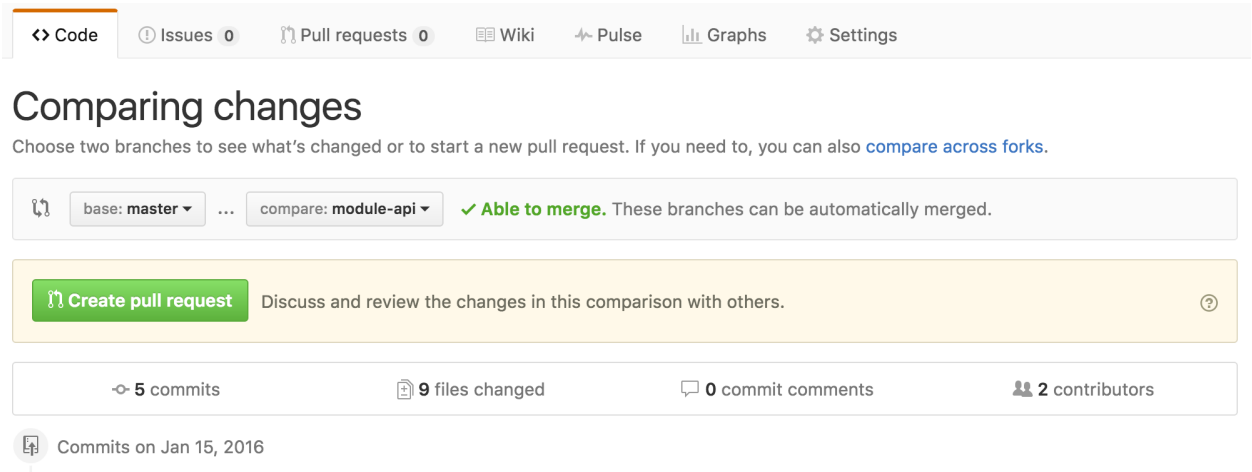

Pulse
Graphs
Settings
Webhooks / Add webhook
We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in our developer documentation.
Payload URL *
https://hooks.slack.com/services/T029L697B/B09FW6YQ5/yOH
Content type
application/json
Secret
By default, we verify SSL certificates when delivering payloads.
Disable SSL verification
Which events would you like to trigger this webhook?
Just the push event.
Send me everything.
Let me select individual events.
Commit comment
Commit or diff commented on.
Create
Branch or tag created.
Delete
Branch or tag deleted.
Deployment
Repository deployed.
Deployment status
Deployment status updated from the API.
Fork
Repository forked.
Gollum
Wiki page updated.
Issue comment
Issue comment created, edited, or deleted.
Issues
Issue opened, edited, closed, reopened, assigned, unassigned, labeled, or unlabeled.
Member
Collaborator added to a repository.
Page build
Pages site built.
Public
Repository changes from private to public.
Pull request
Pull request opened, closed, assigned, labeled, or synchronized.
Pull request review comment
Pull request diff comment created, edited, or deleted.
Push
Git push to a repository.
Release
Release published in a repository.
Repository
Repository created, deleted, publicized, or privatized.
Status
Commit status updated from the API.
Team add
Team added or modified on a repository.
Watch
User stars a repository.
Active
We will deliver event details when this hook is triggered.
Add webhook

STEP 4 When you want to submit any of the three modules for the project, create a **Pull Request (PR)** by navigating to this tab and click the button **New Pull Request.**



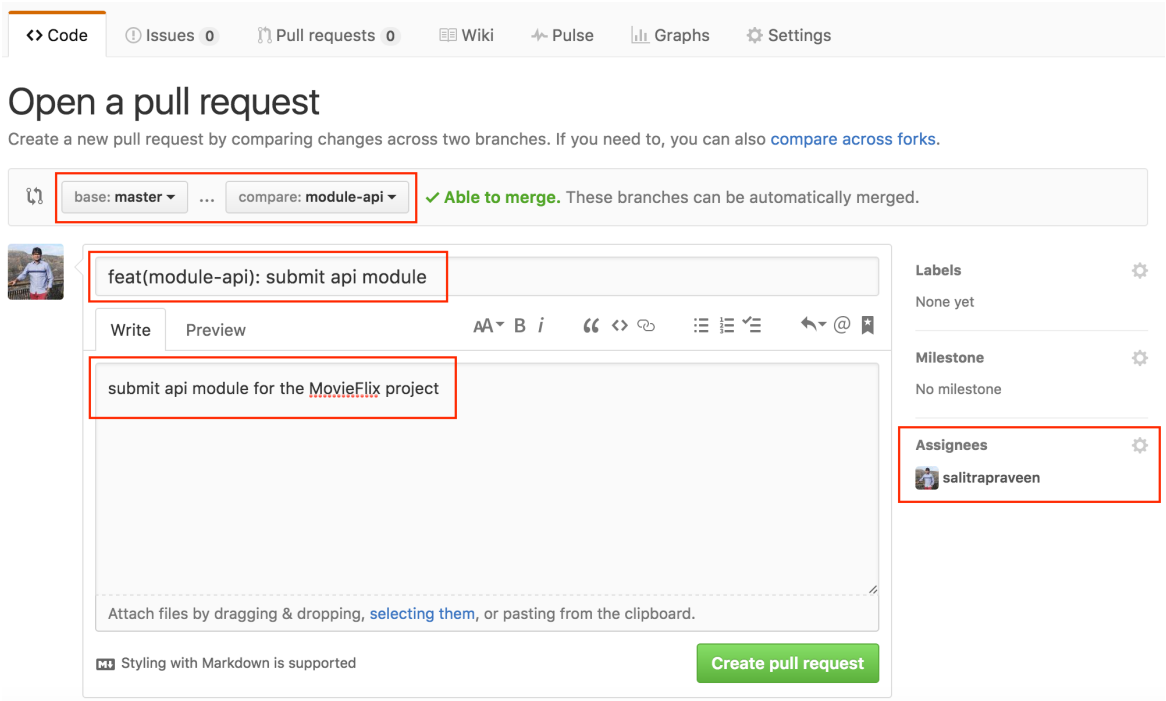
STEP 5 Select **master** for the base branch and respective module branch (**module-***) for the compare branch.

In the example below, a PR is being created from the **module-api** branch into the **master** branch. You will do the same for all the 3 modules submissions.
Click on the button **Create pull request.**



STEP 6 Fill the necessary details as shown in the image. Optionally, you can also assign this PR to **salitrapraveen** for review (But not required).

Follow the message format as shown in the example below.



STEP 7 When you create a PR on this repo, it will send an alert to our reviewers. They will review your submission and comment on the issues, if any.

NOTE **STEP 1 – STEP 3** are one time updates only, unless you delete your repository.

A **PR** between two branches can only be created if they have a difference in their history. If both the branches are identical, you cannot create a PR. So, make sure, you are always working in one of the **module-*** branch and not the **master** branch.

GitHub Commit Message Guidelines

Borrowed from https://github.com/angular/

WHY? We have very precise rules over how our git commit messages can be formatted. This leads to more **readable messages** that are easy to follow when looking through the **project history**.

FORMAT Must be one of the following:

feat: A new feature
fix: A bug fix
docs: Documentation only changes
style: Changes that do not affect the meaning of the code (white-space, formatting, missing semi-colons, etc.)
refactor: A code change that neither fixes a bug nor adds a feature
perf: A code change that improves performance
test: Adding missing tests
chore: Changes to the build process or auxiliary tools and libraries such as documentation generation.

The subject contains succinct description of the change:

1. use the imperative, present tense: "change" not "changed" nor "changes"
2. don't capitalize first letter
3. no dot (.) at the end

EXAMPLES

feat(module-client): add bootstrap to the styling
feat(module-api): init api module
feat(module-api): implement user authentication
test(module-client): add jasmine/karma config
style(module-client): format JavaScript code
refactor(module-client): use directory structure specified by John Papa
docs(module-client): update readme with how to run