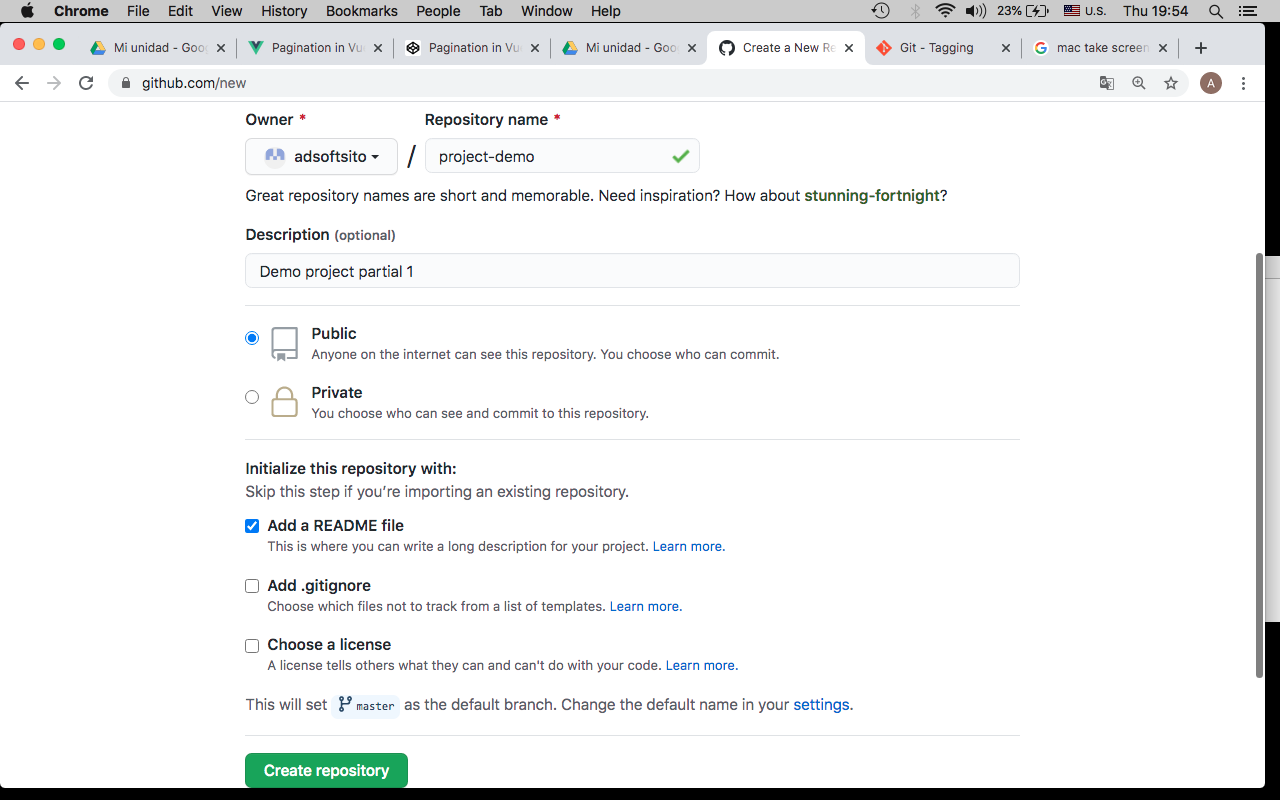
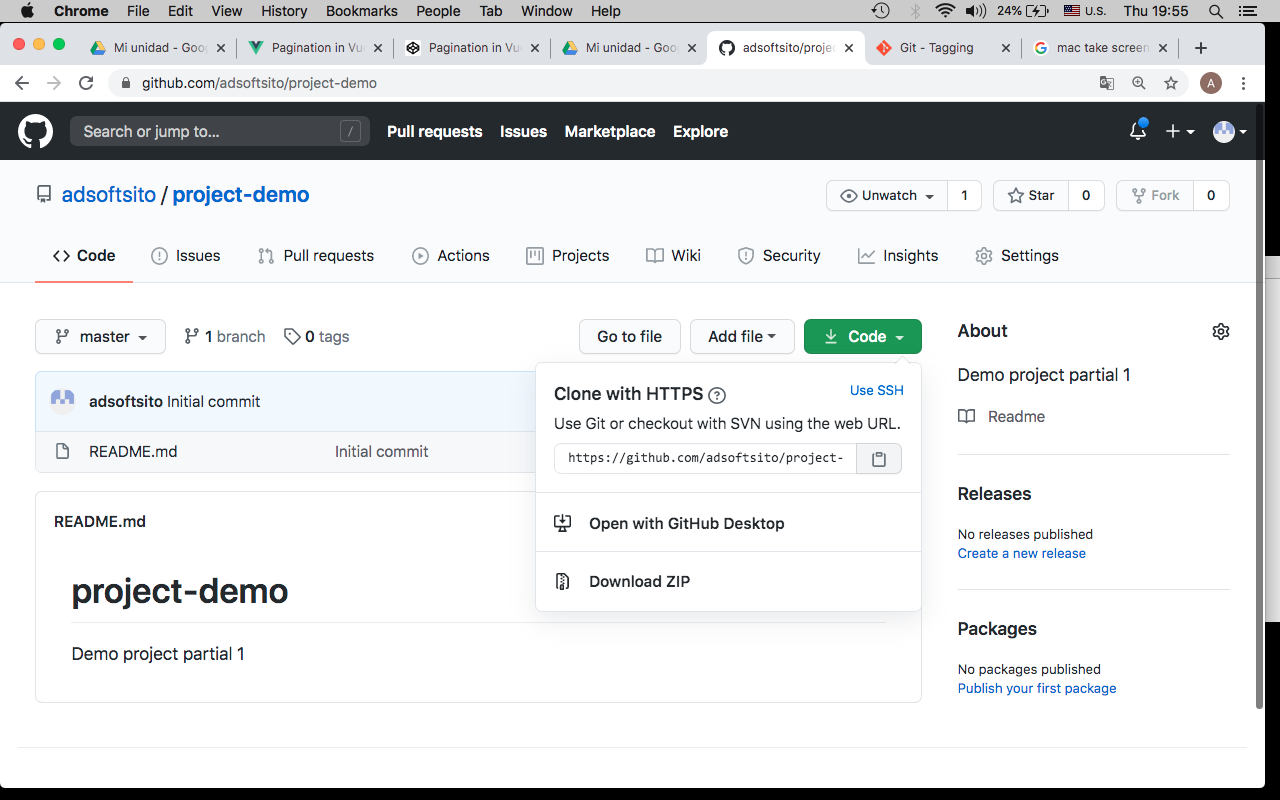
**Git branching model & Automated kanban boards**

1.- Go to <https://github.com>

2.- Create new repository **project-demo**



3.- Copy url repository



4.- Clone repository in wokstation

*$ git clone* [*https://github.com/<gituser>/project-demo.git*](https://github.com/%3cgituser%3e/project-demo.git)

5.- Change to repo directory

*$ cd project-demo*

6.- Check url repository

*$ pwd*

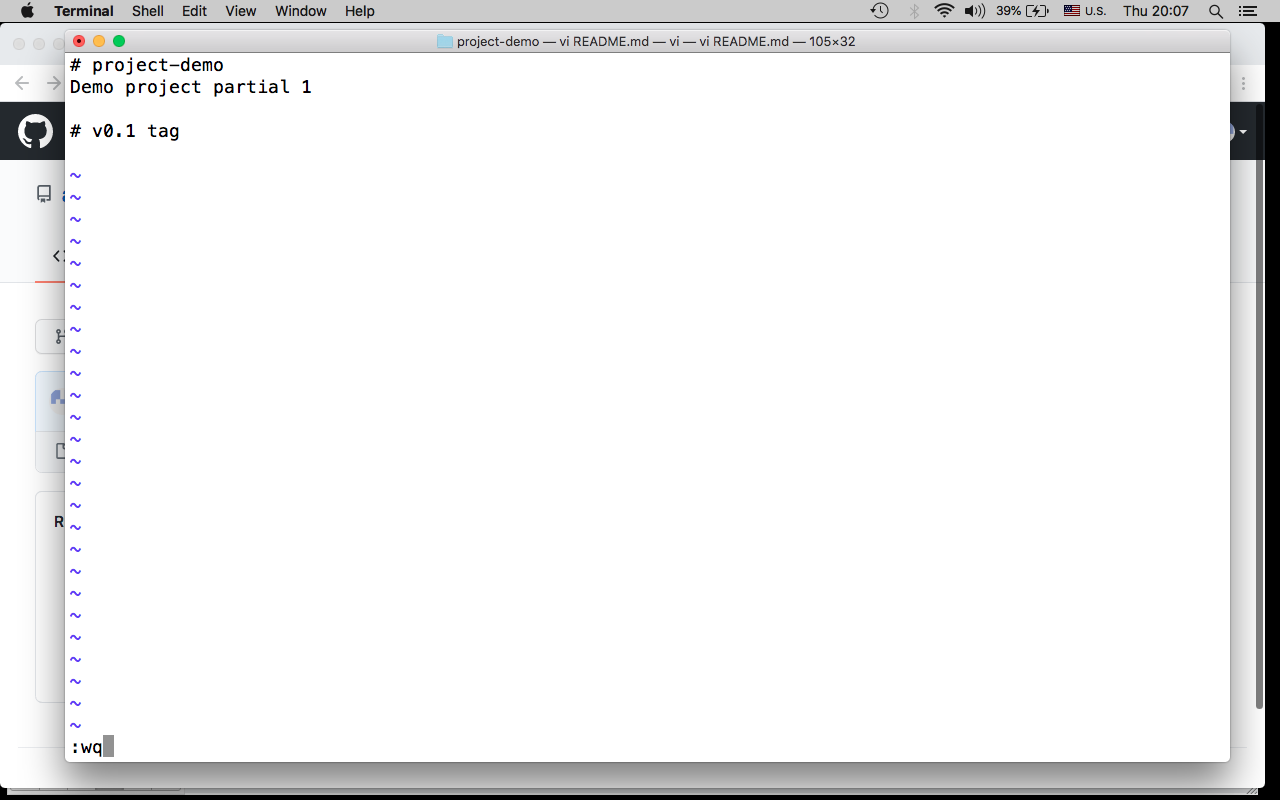
*$ git remote –v*

*origin https://github.com/adsoftsito/project-demo.git (fetch)*

*origin https://github.com/adsoftsito/project-demo.git (push)*

7.- Edit README.md, add # v0.1 tag

*$ vi README.md*



8.- Commit, push changes

*$ git add .*

*$ git commit -m "update README with v0.1 tag"*

*$ git push –u origin master*

9.- Tag base code with v0.1

*$ git tag -a v0.1 -m "my version 0.1"*

*$ git tag*

*$ git show v0.1*

9.- Publish tag

*$ git push origin v0.1*

10.- create branch hotfixes, release and develop

*$ git checkout master*

*$ git branch hotfixes*

*$ git checkout hotfixes*

*$ vi HOTFIXES.md*

*# hotfixes branch here*

*:wq*

*$ git add .*

*$ git commit –m “init hotfixes branch”*

*$ git push –u origin hotfixes*

##############################

$ git checkout master

$ git branch release

$ git checkout release

*$ vi RELEASE.md*

*# release branch here*

*:wq*

*$ git add .*

$ git commit –a –m “init release branch”

$ git push –u origin release

#############################

*$ git checkout master*

*$ git branch develop*

*$ git checkout develop*

*$ vi DEVELOP.md*

*# develop branch here*

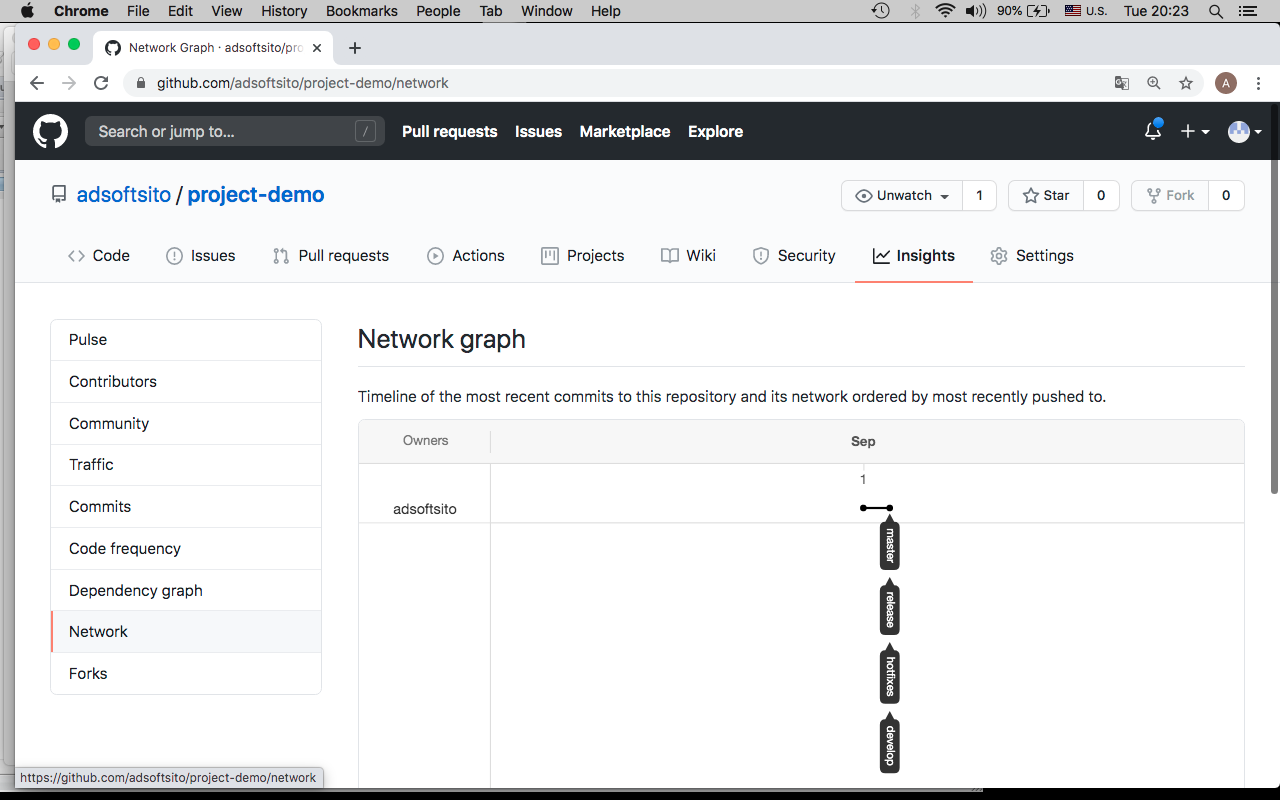
*:wq*

*$ git add .*

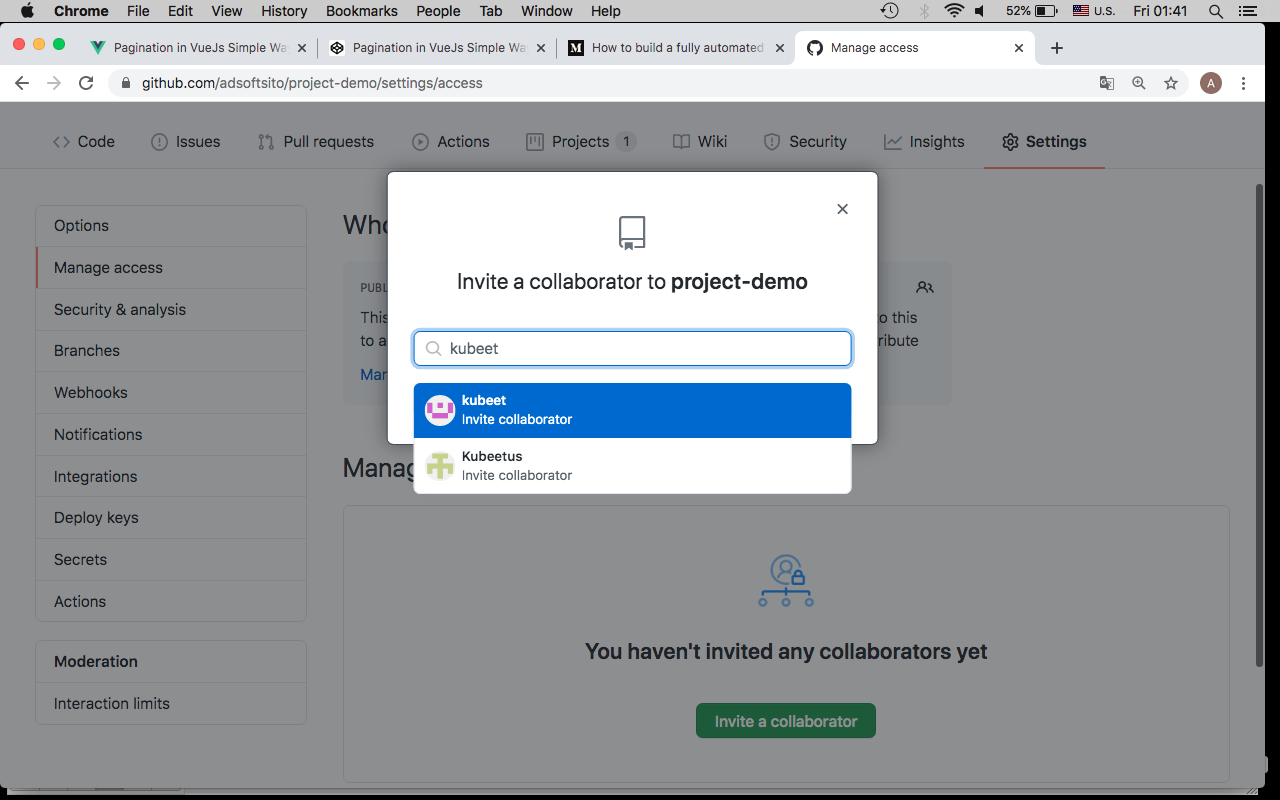
*$ git commit –a –m “init develop branch”*

*$ git push –u origin develop*

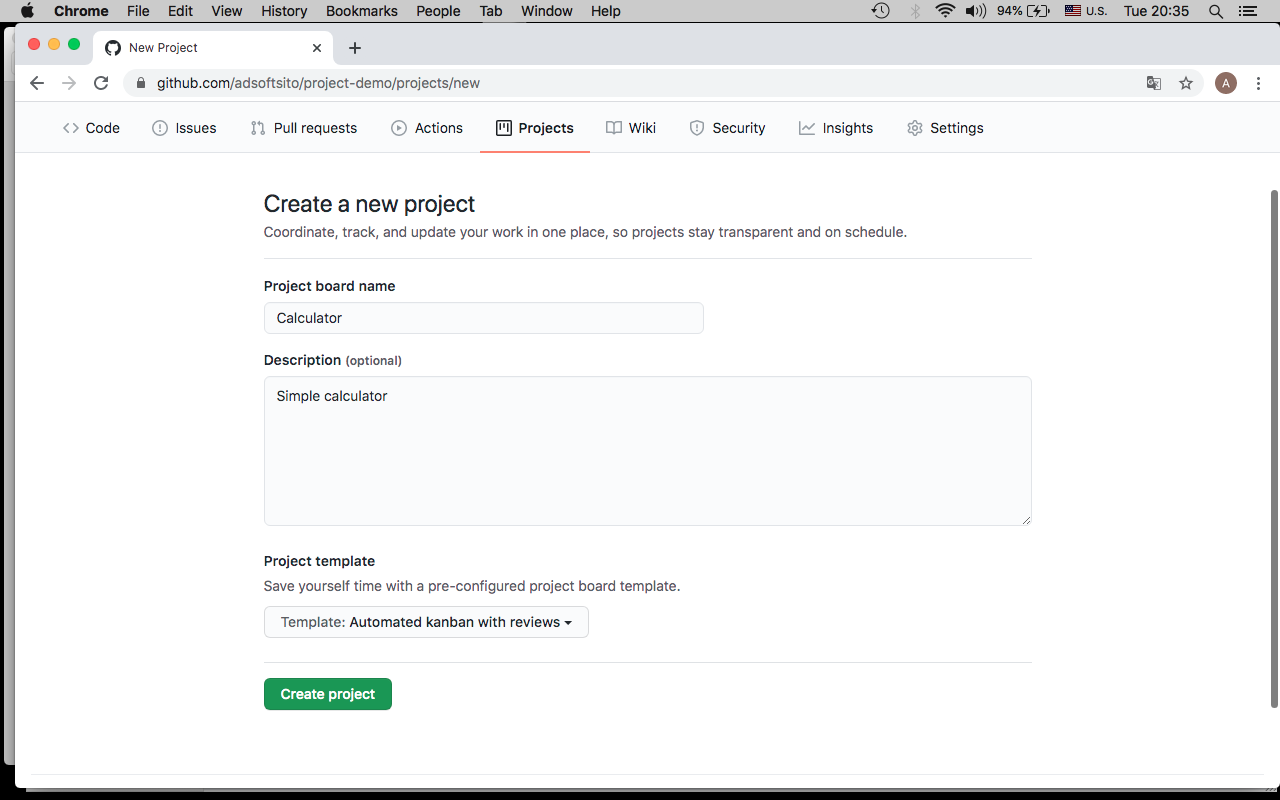
11.- Visualize Network graph in Insights menu



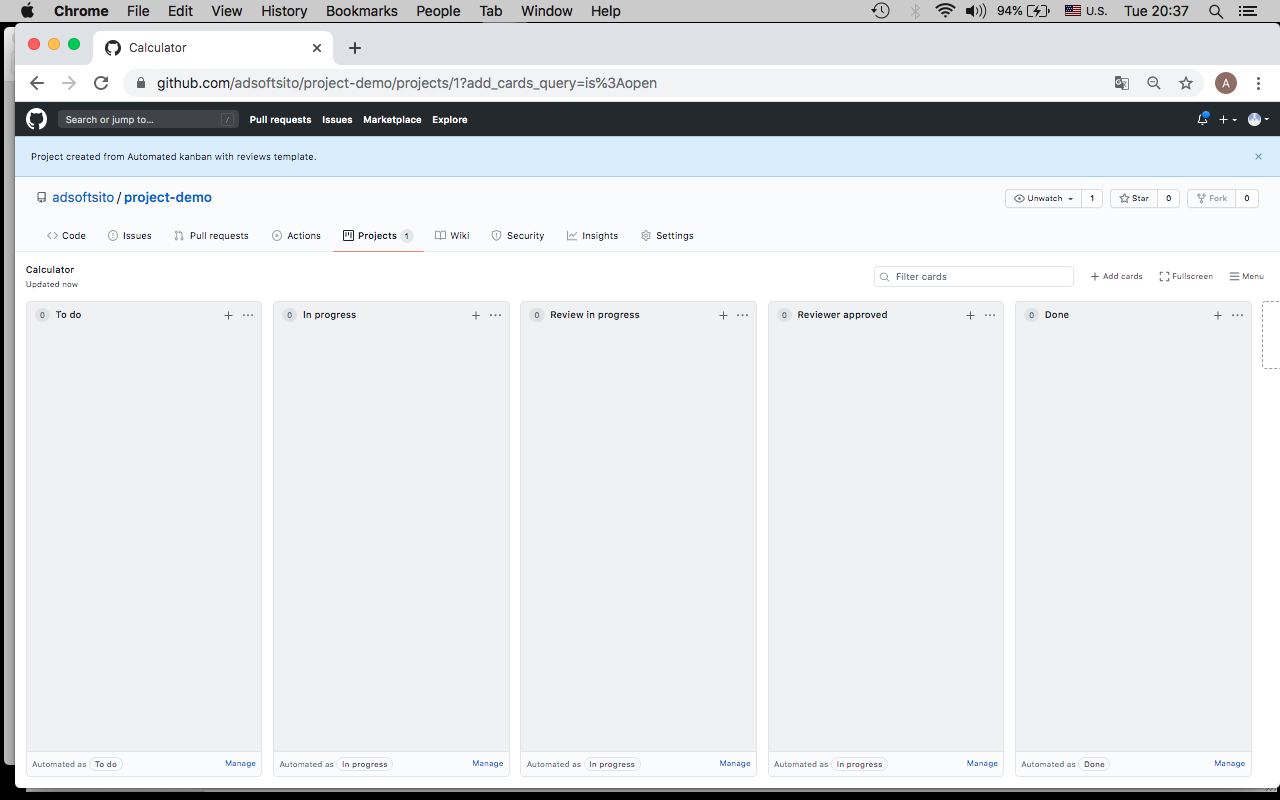
12.- Add team members to repository



13.- Create a Project in github.com, select Automated Kanban with Reviews template.



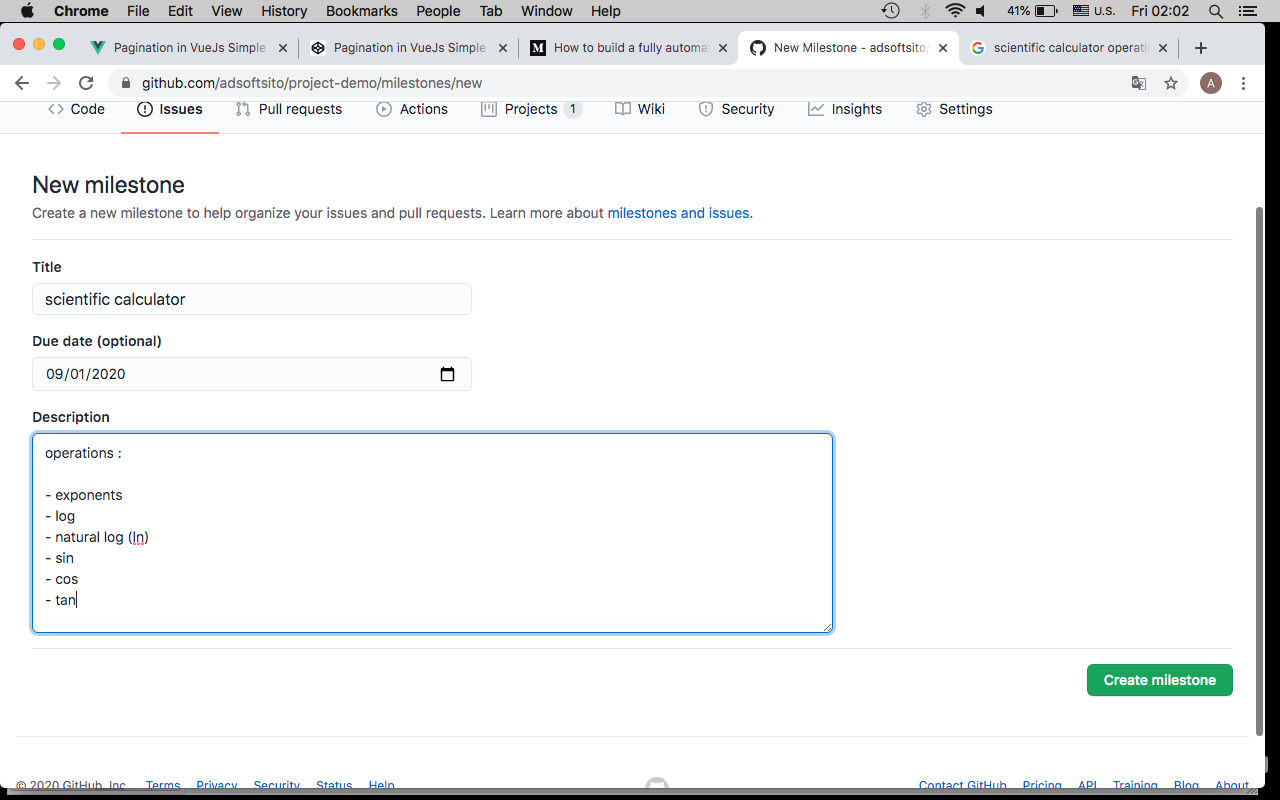
14.- Delete default cards in “To do” Column



15.- créate “simple calculator” milestone

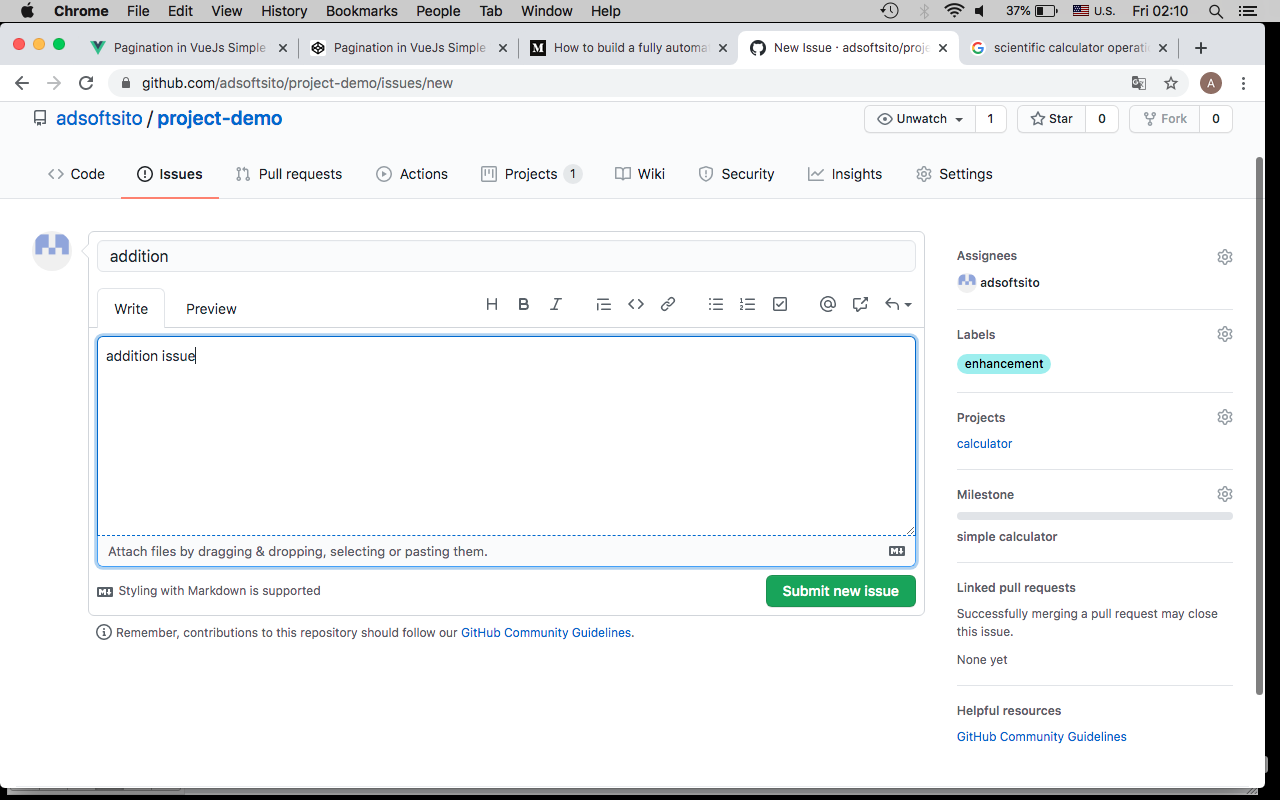


16.- créate “scientific calculator” milestone



17.- Add addition issue’s to calculator Project

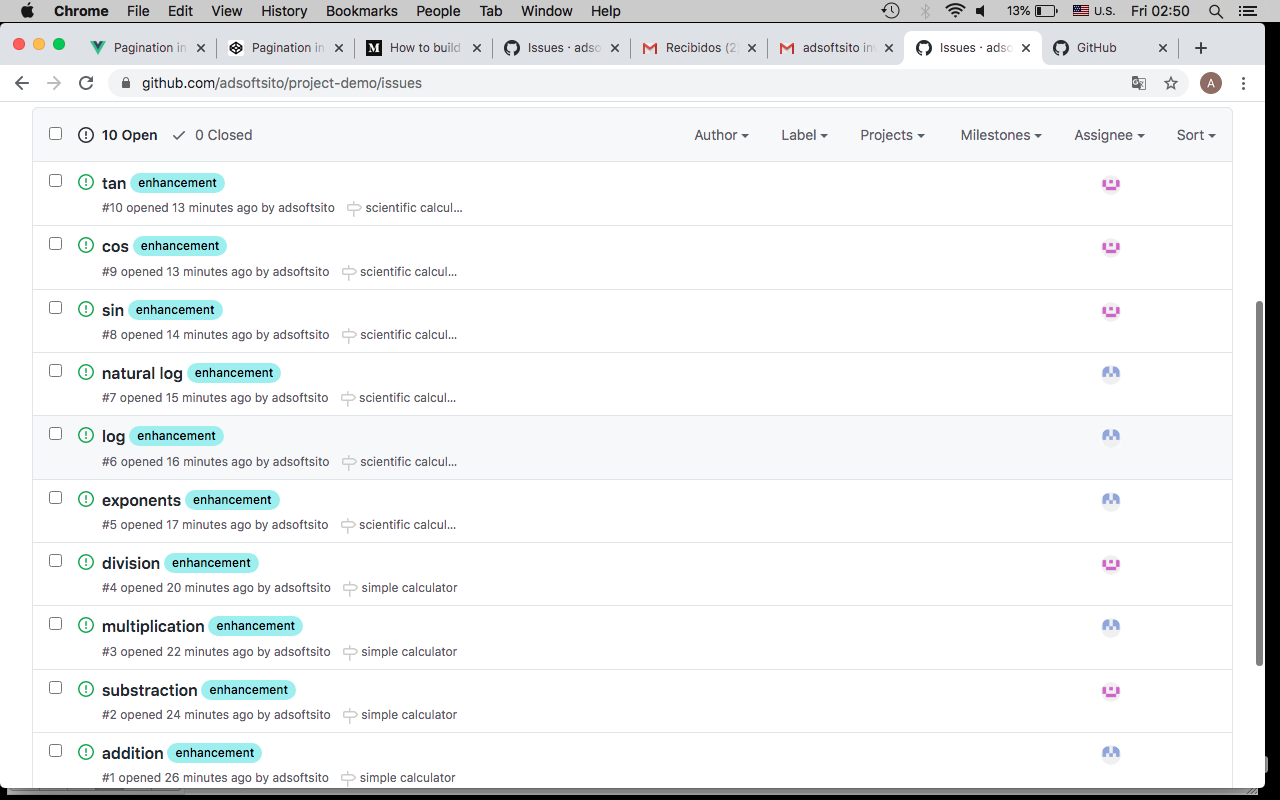
­­



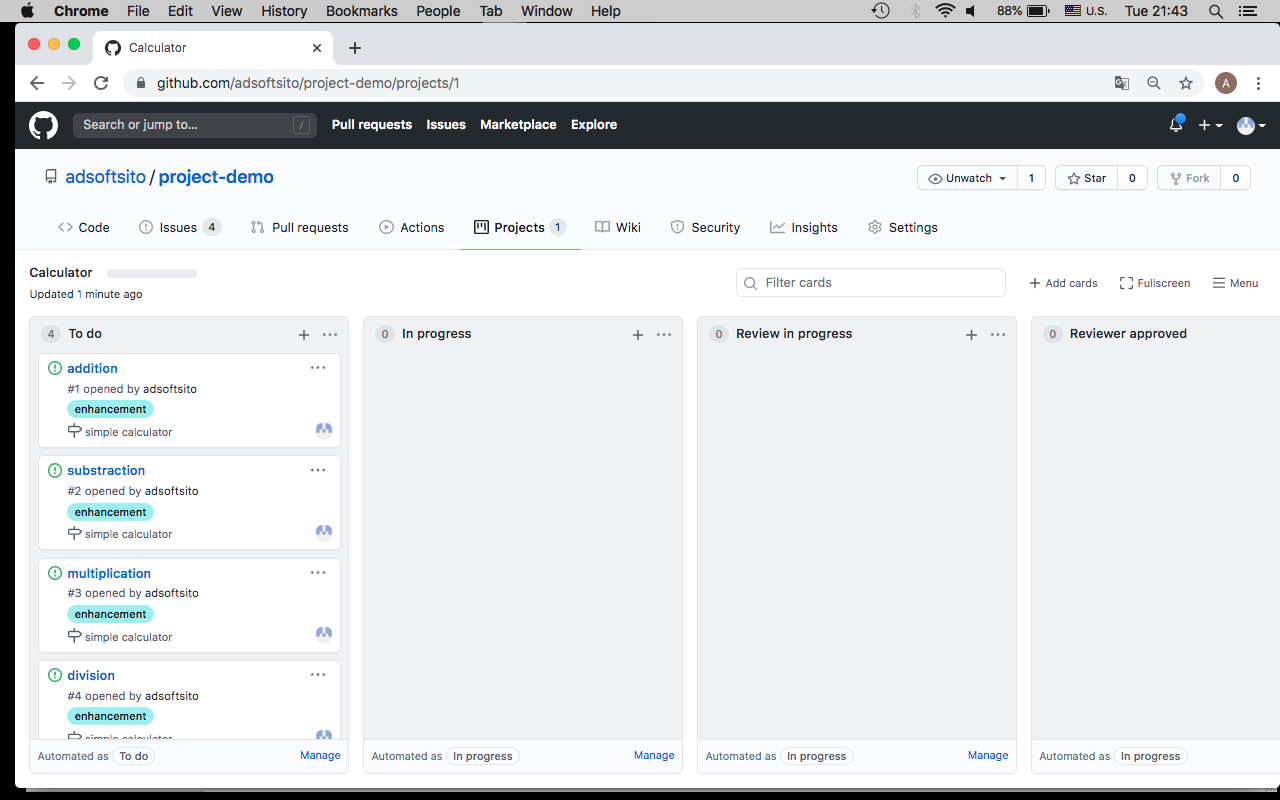
18.- complete issue’s according the next table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **issue name** | **write** | **assignees** | **labels** | **Project** | **milestone** |
| addition | add .. | developer1 | enhancement | calculator | simple calculator |
| substraction | sub … | developer2 | enhancement | calculator | simple calculator |
| multiplication | mul … | developer3 | enhancement | calculator | simple calculator |
| division | div … | developer4 | enhancement | calculator | simple calculator |
| exponents | exp … | developer1 | enhancement | calculator | scientific calculator |
| log | log … | developer2 | enhancement | calculator | scientific calculator |
| natural log | ln … | developer3 | enhancement | calculator | scientific calculator |
| sin | sin … | developer4 | enhancement | calculator | scientific calculator |
| cos | cos … | developern | enhancement | calculator | scientific calculator |
| tan | ln … | developern | enhancement | calculator | scientific calculator |

19.- issue’s should looks like :



21.- Calculator project board looks like :



22.- Create a branch for each issue assigned to you

*$ git checkout develop*

*$ git branch <issue>*

# example : git branch addition

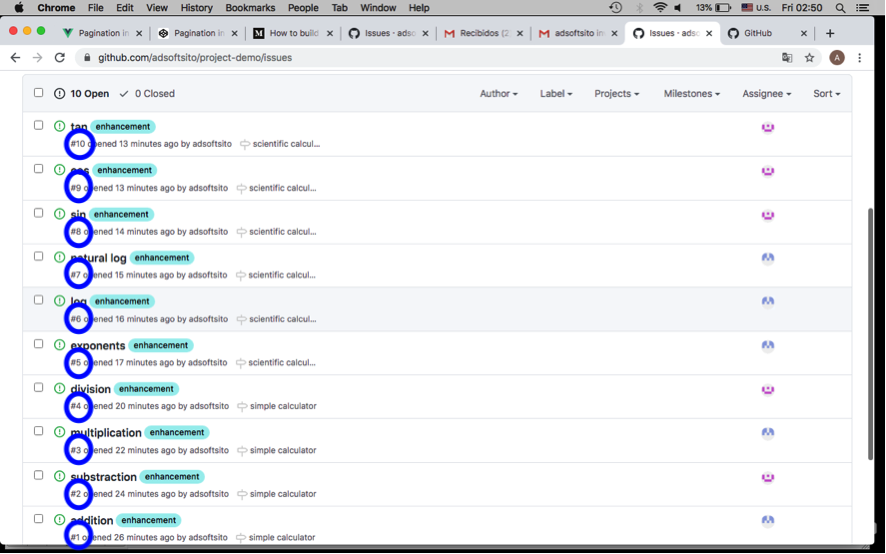
*$ git checkout <issue>*

# example: git checkout addition

*# show current branch*

*$ git branch*

23.- Add some commits with issue reference #issuenumber, check issue number before



*$ vi <issue>.java*

# example: vi addition.java

/\*

insert code here…

\*/

:wq

*$ git add <issue>.java*

# example: git add addition.java

*$ git commit –m “add <issue> base code #<issuenumber>”*

$ example: git commit –m “add addition base code #1”

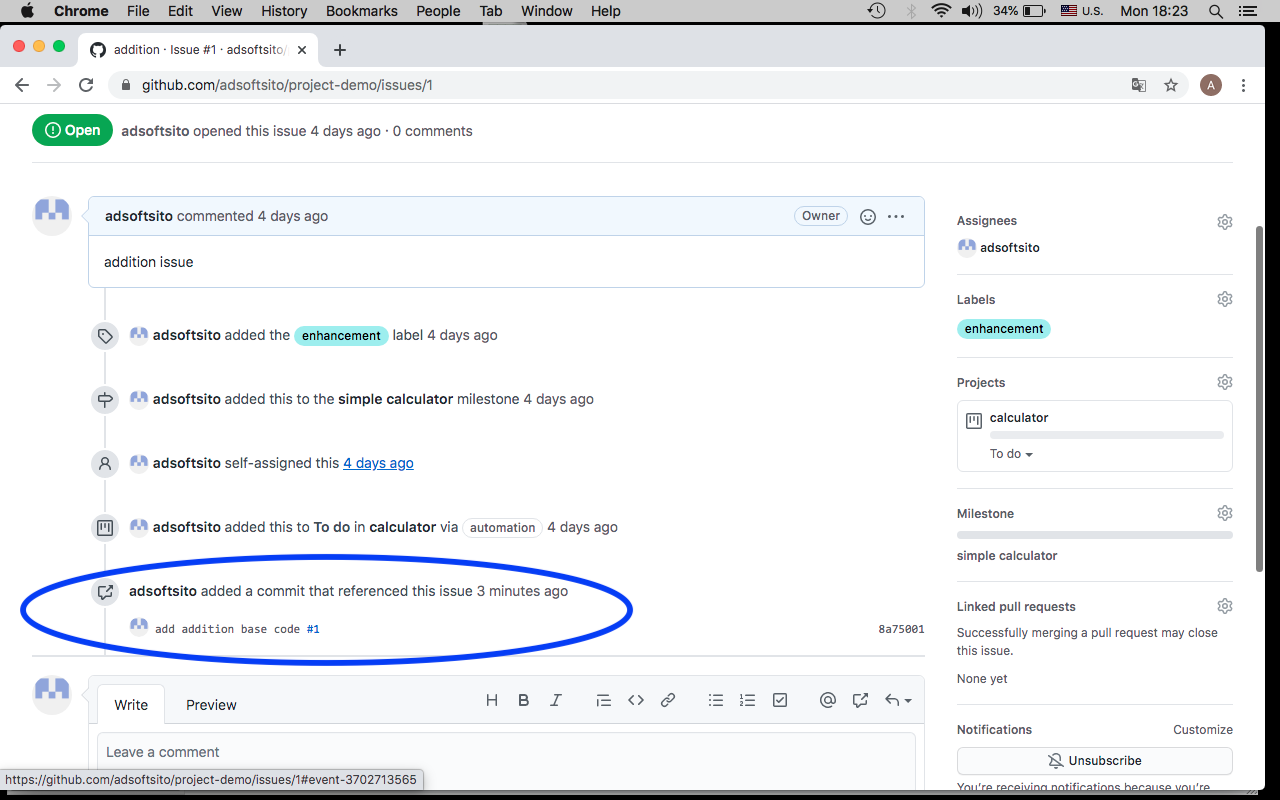
# check local status vs remote status

*$ git status*

*$ git push –u origin <issue>*

$ example: git push –u origin addition

24.- Check issue log



25.- Add more commits with #<issuenumber> and verify issue status

*$ vi <issue>.java*

# example: vi addition.java

/\*

Author: adsoft

Date: Sep 2, 2020

\*/

/\*

insert code here…

\*/

:wq

*$ git commit –a –m “add author and date #<issuenumber>”*

*$ example: git commit –a –m “add autor an date #1”*

*$ git status # check local status vs remote status*

###############################

*$ vi <issue>.java*

*# example: vi addition.java*

/\*

Author: adsoft

Date: Sep 2, 2020

\*/

/\*

insert code here…

\*/

/\*

code is ready !

\*/

:wq

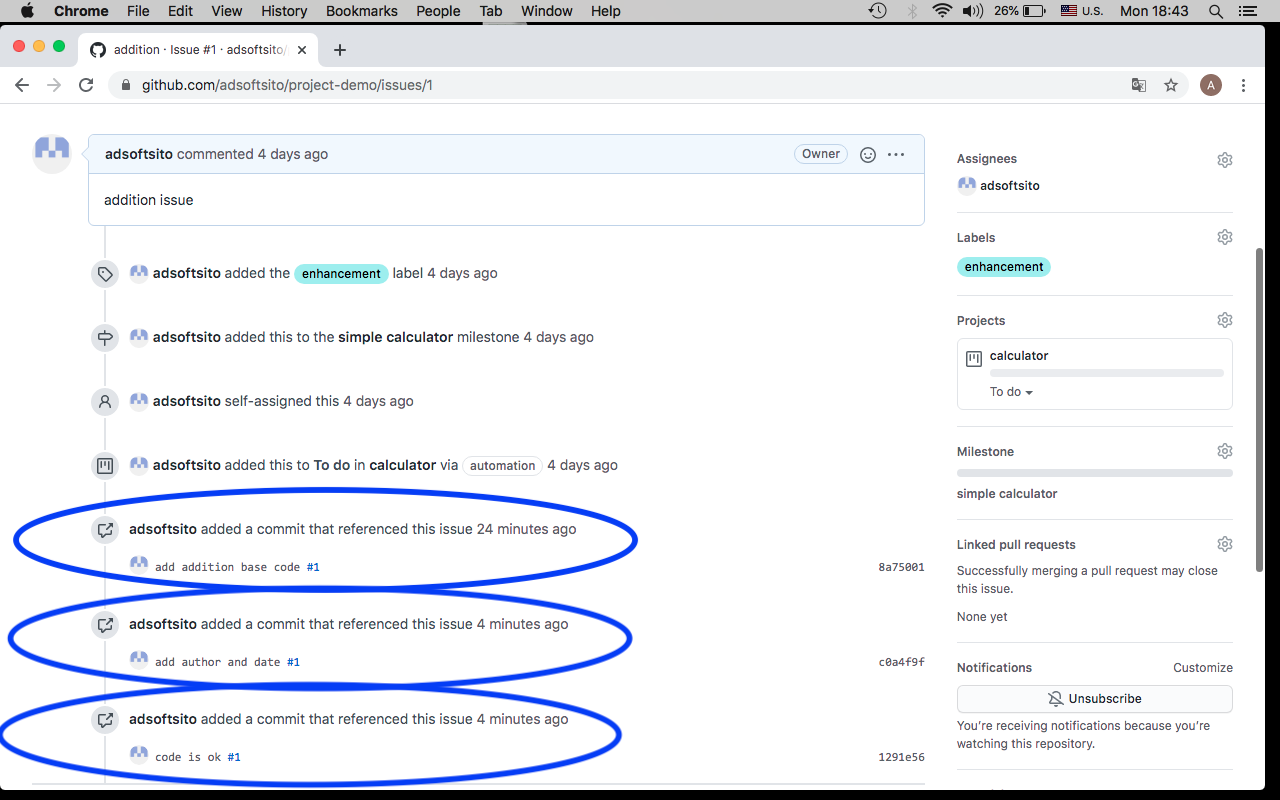
*$ git commit –a –m “code is ok #<issuenumber>”*

$ example: git commit –a –m “code is ok #1”

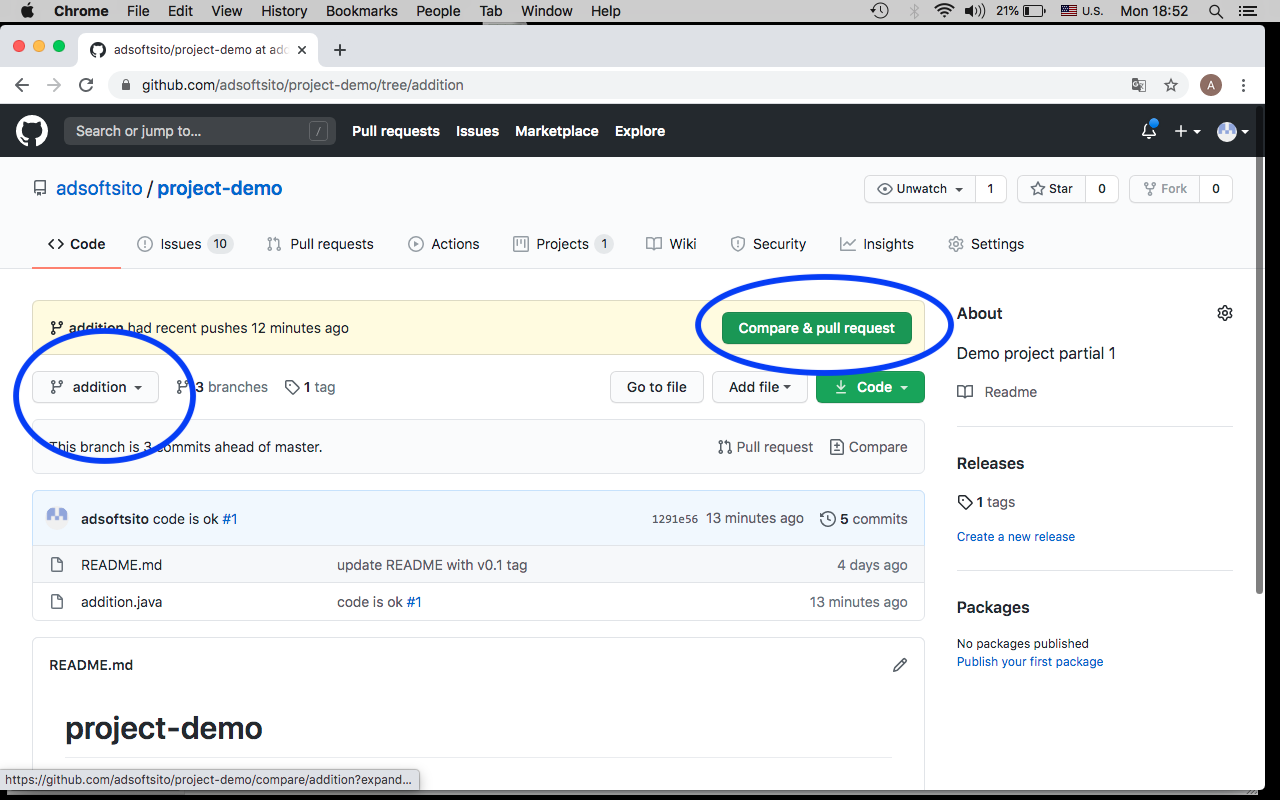
*$ git status # check local status vs remote status*

*$ git push –u origin <issue>*

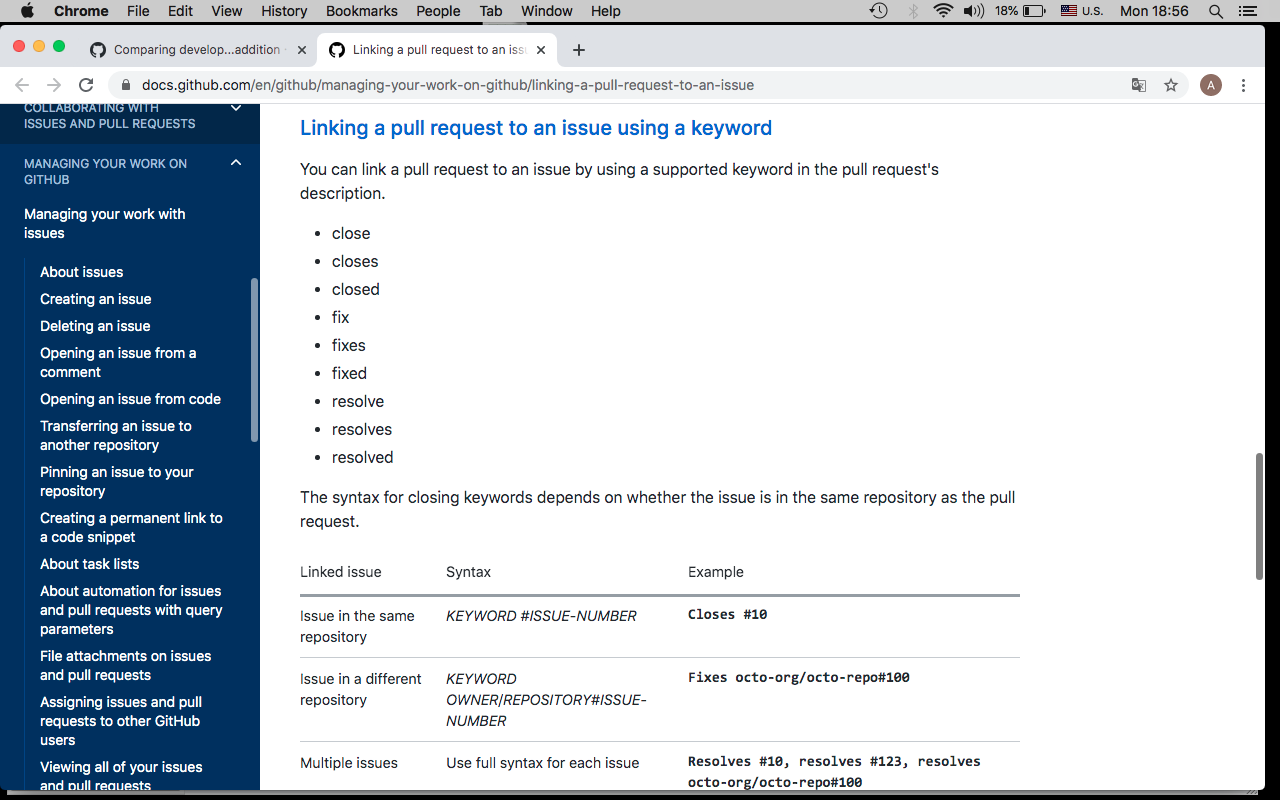
$ example: git push –u origin addition



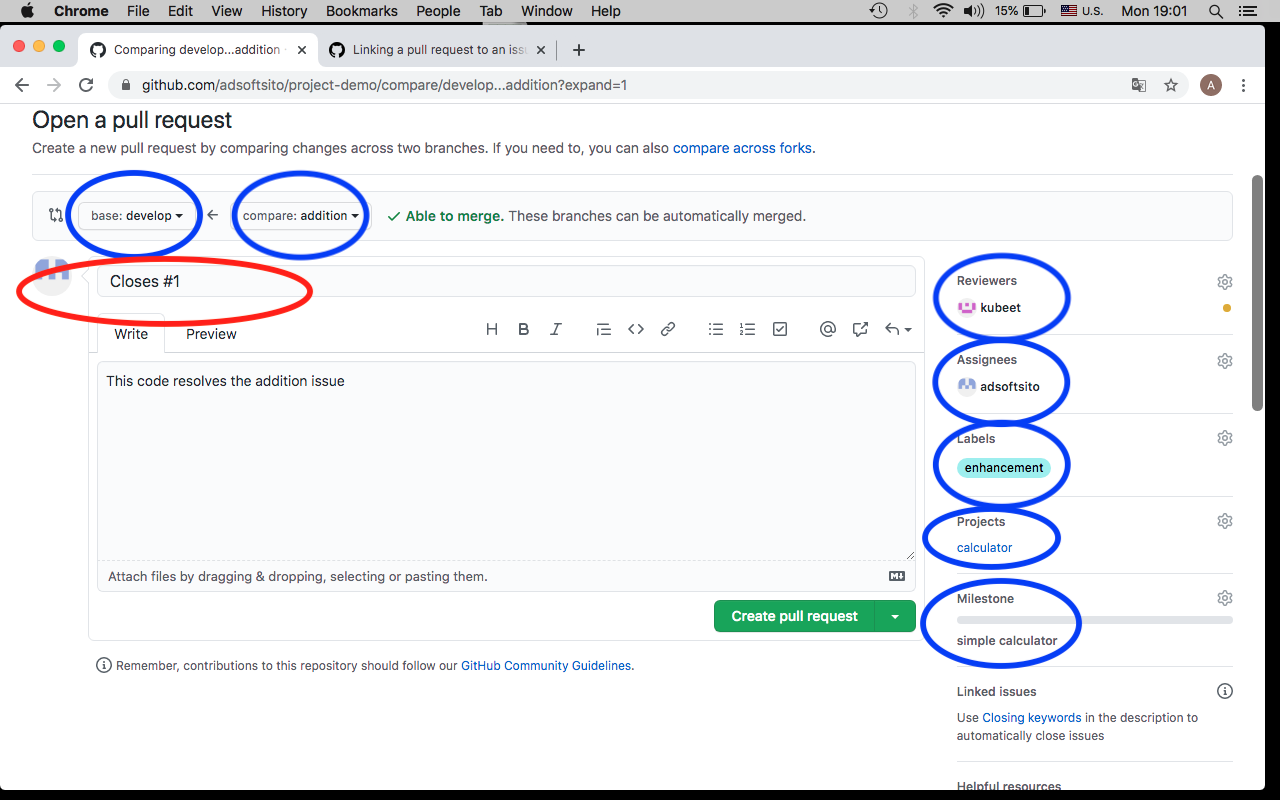
26.- Finally when code is ready, create a Pull Request from this <issue> to branch develop



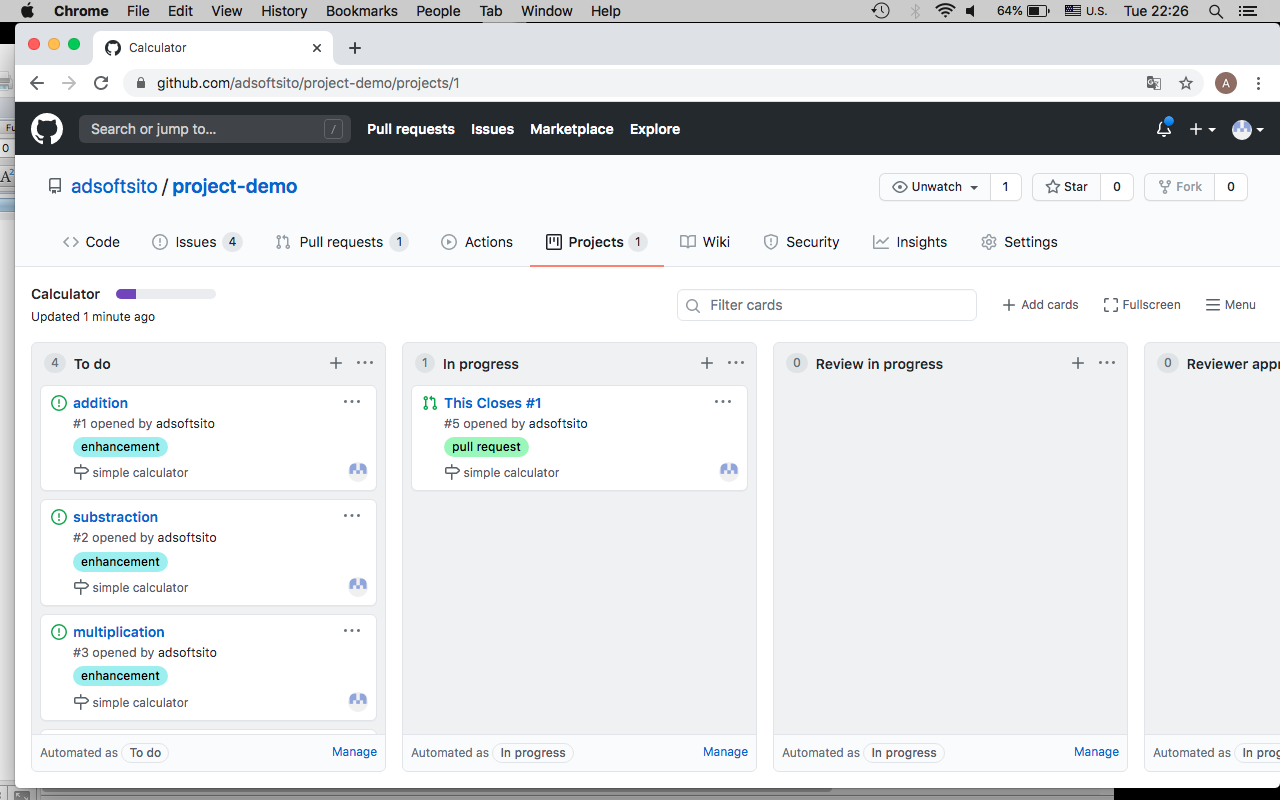
NOTE: close issue with some supported keywords



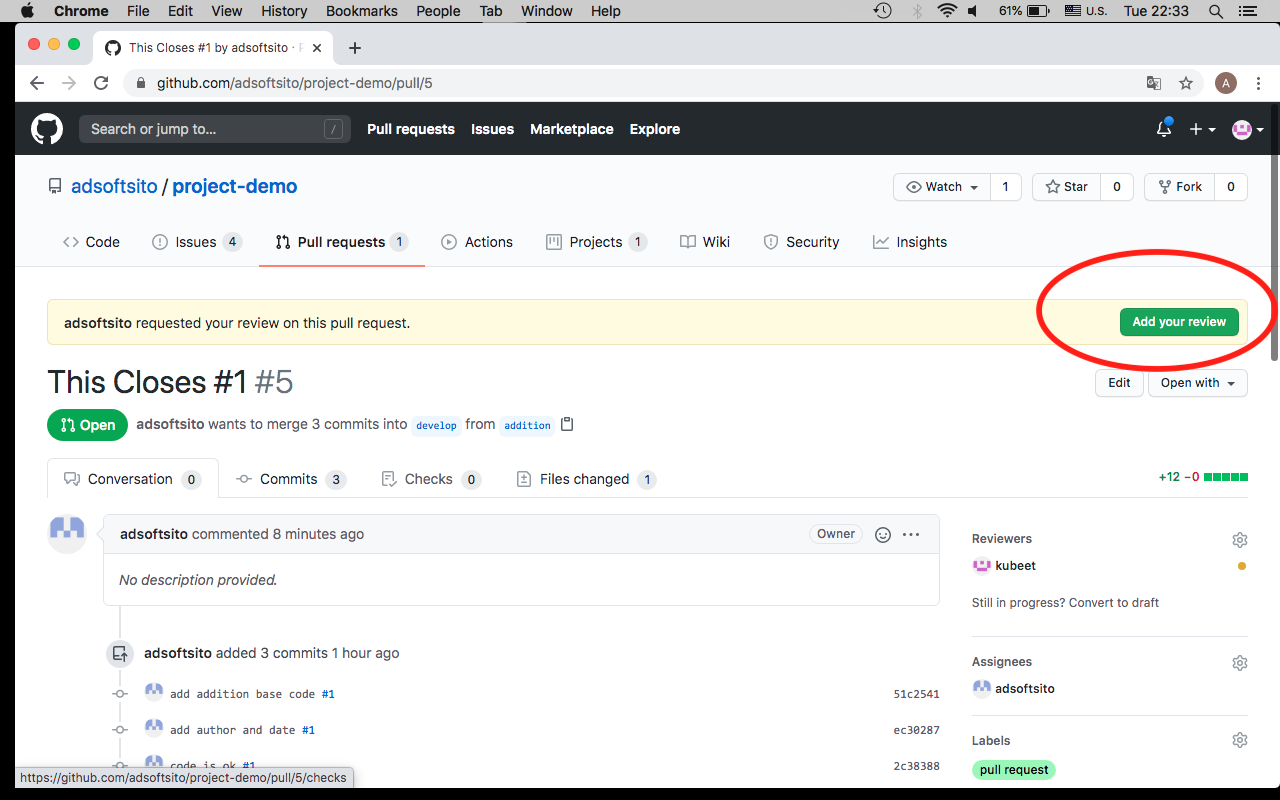
Verify PR contains supported keywords, reviewers, assignees, labels, projects and milestone



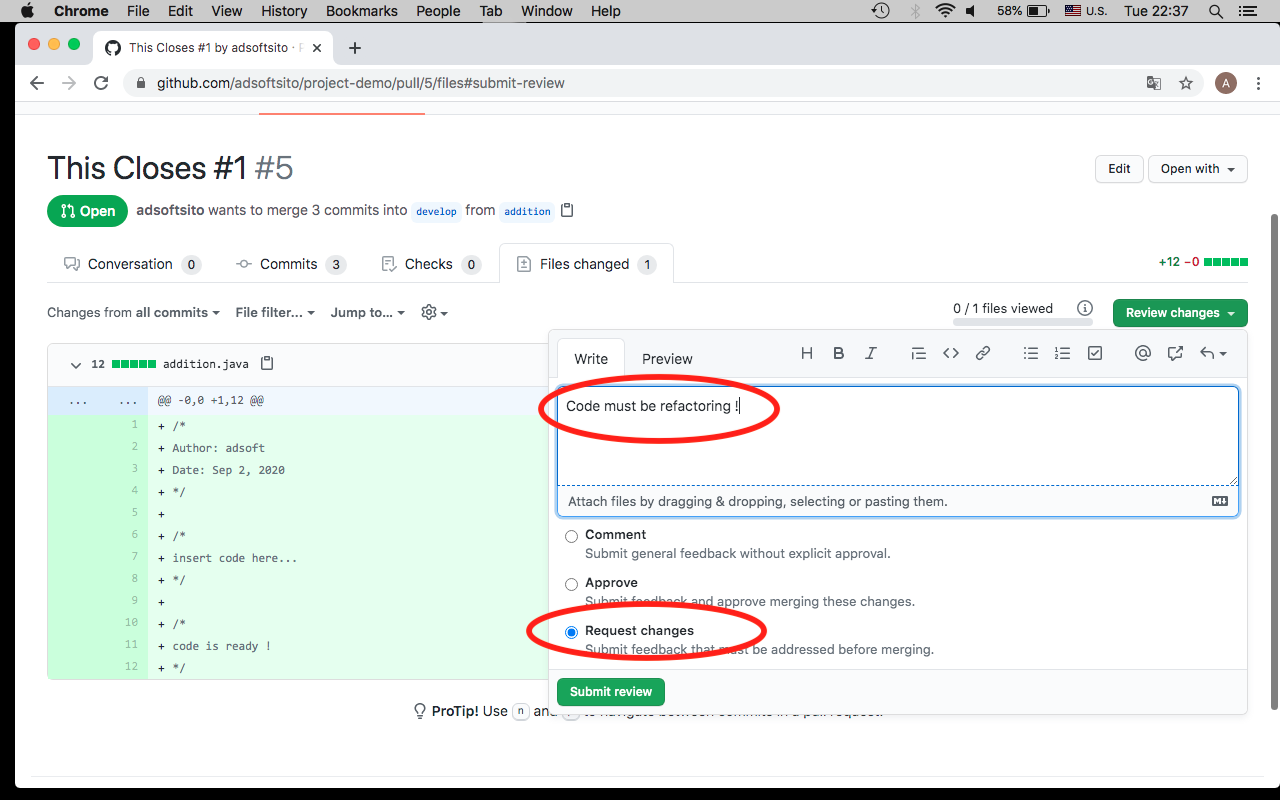
27.- Verify new PR is “In Progress” Column



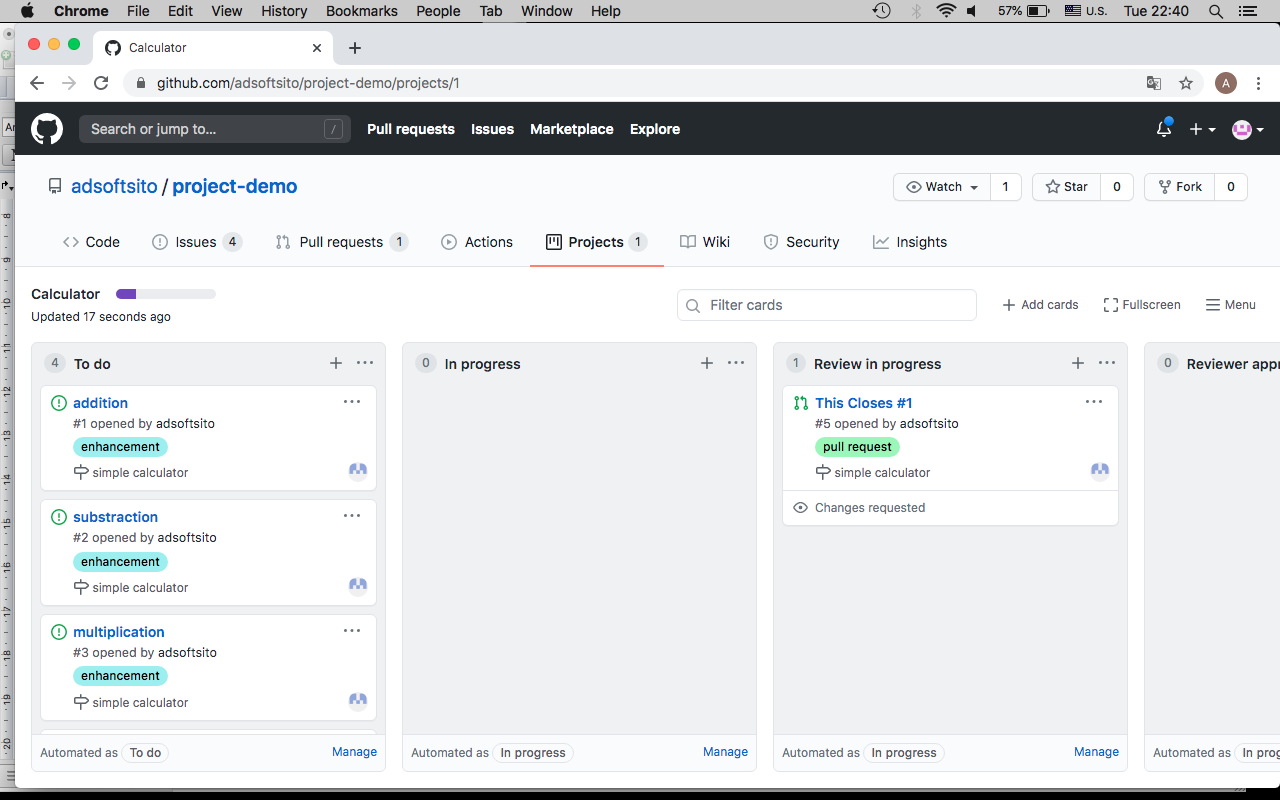
28.- Reviewer should add Review for this issue



29.- Reviewer migth “Request Changes”

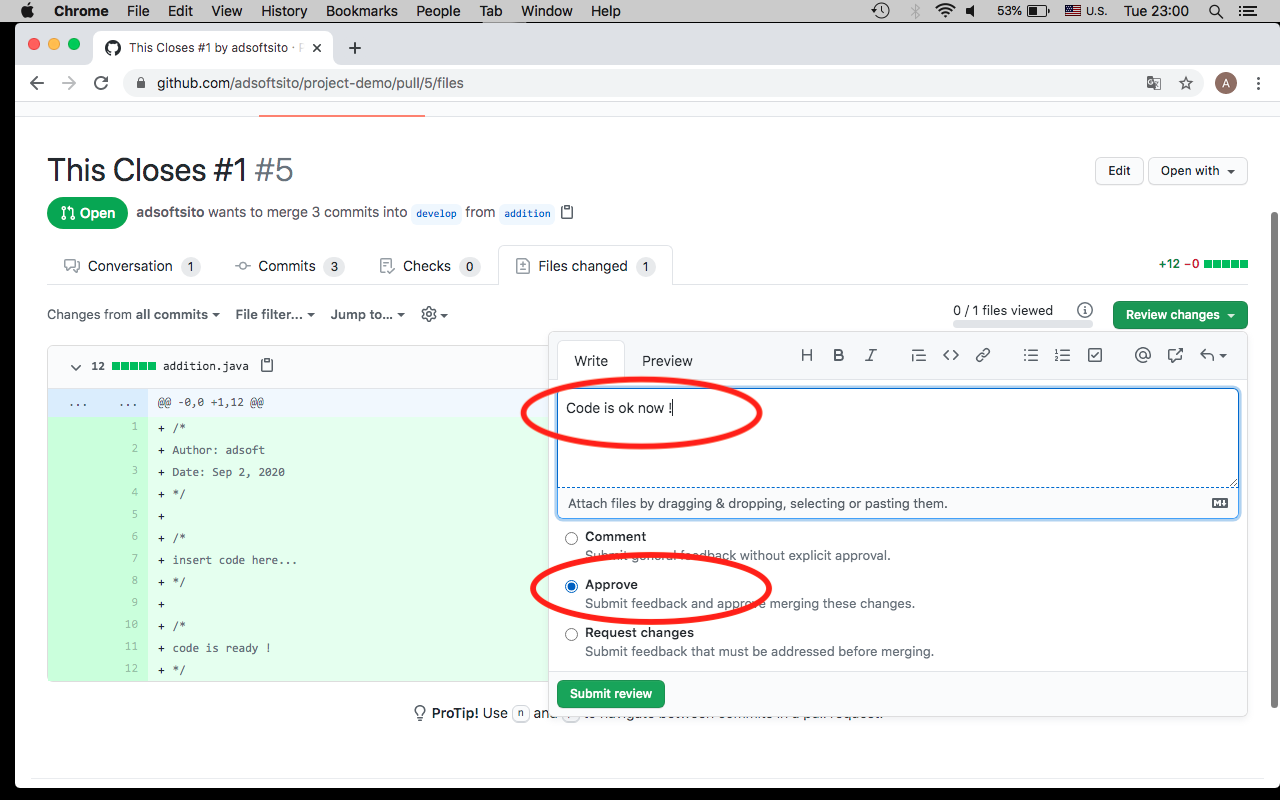


30.- Verify PR moves to “Review in Progress”

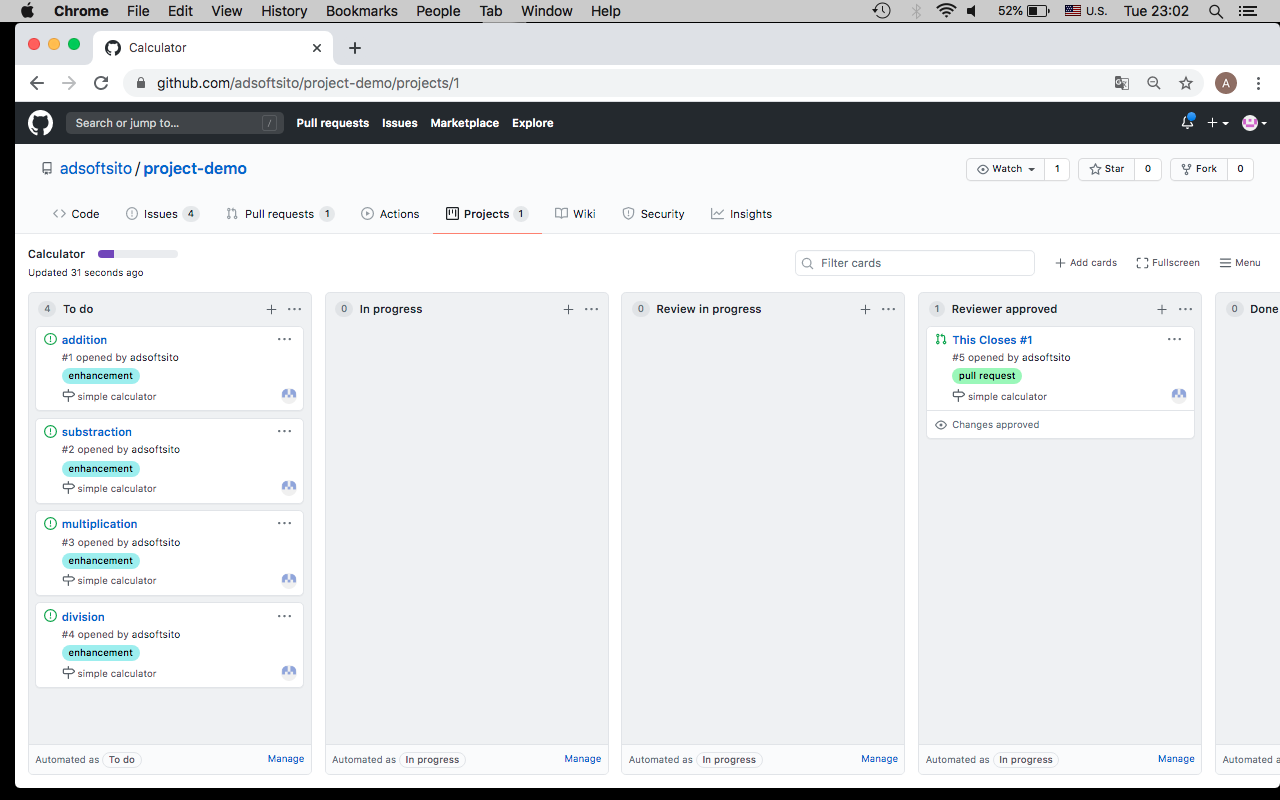


31.- Add more commits to resolve this “Request Changes” (optionally)

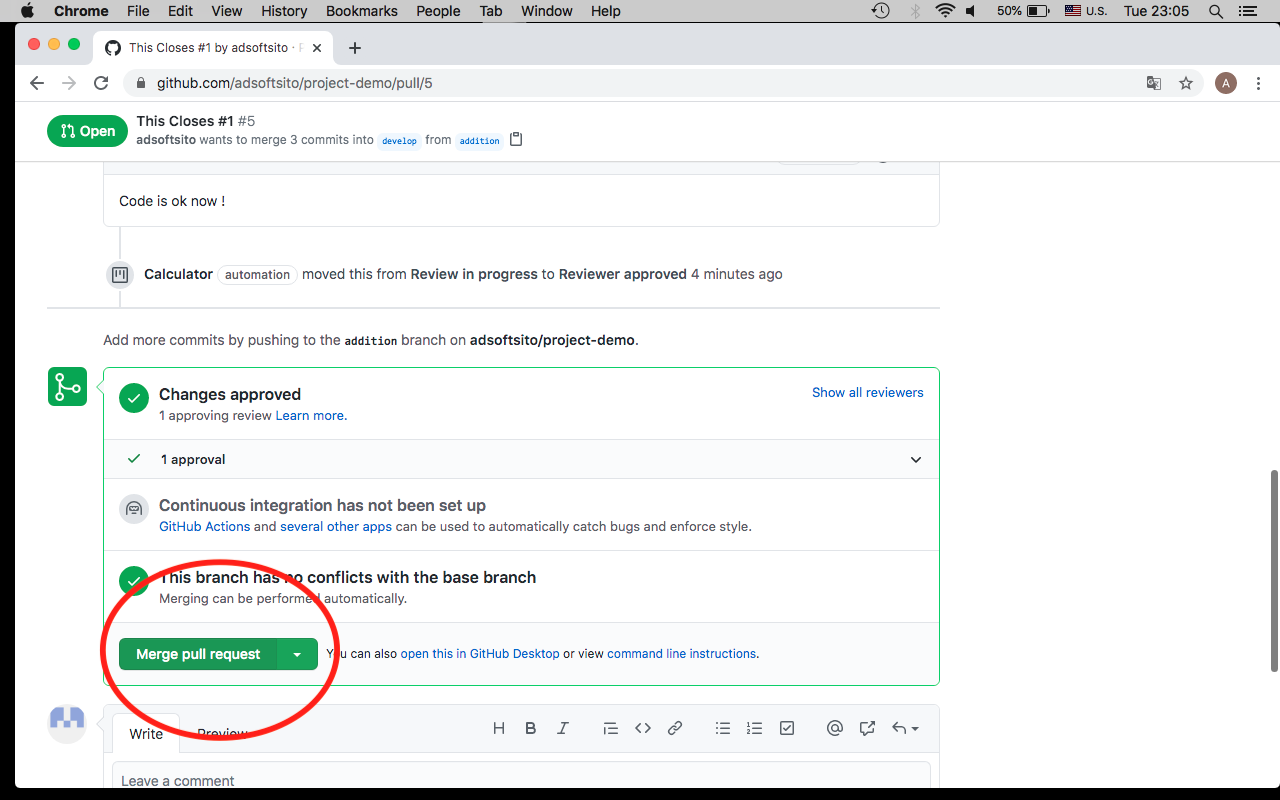
32.- Now, Reviewer should approve issue.



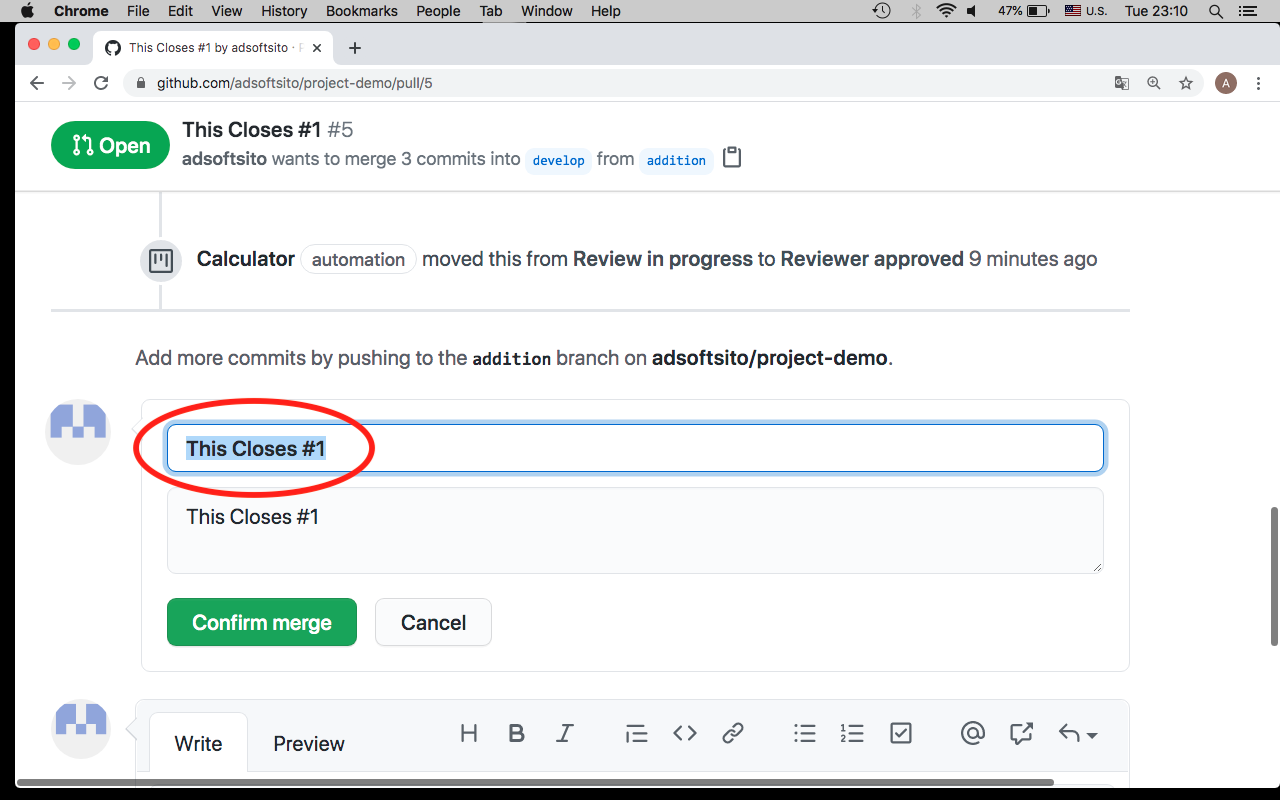
33.- With PR approved, PR moves to “Reviewer Approved”



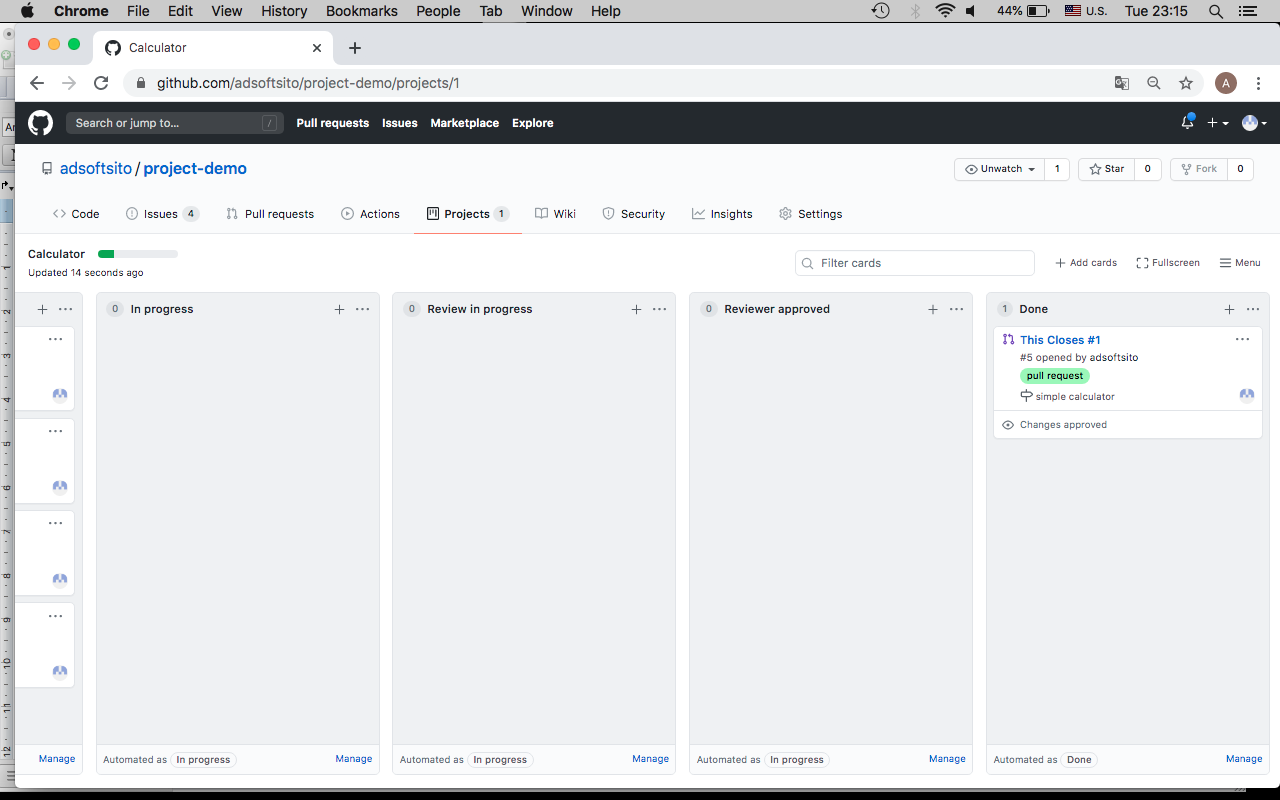
34.- Finally issue owner, must “Merge Pull Request”



35.- Make sure commit message contain keywords : Closes #<n>, Resolves #<n>, Fixes#<n>

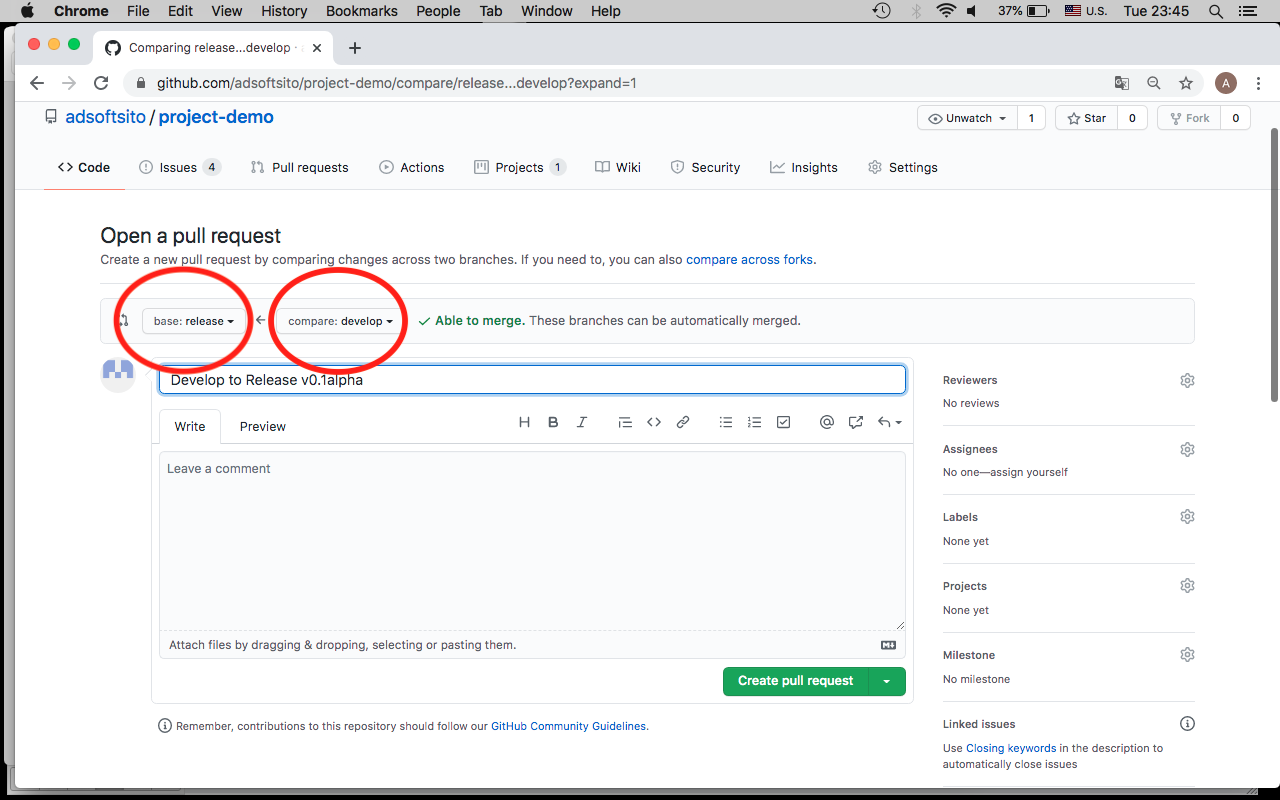


36.- Now, PR moves to “Done” column.

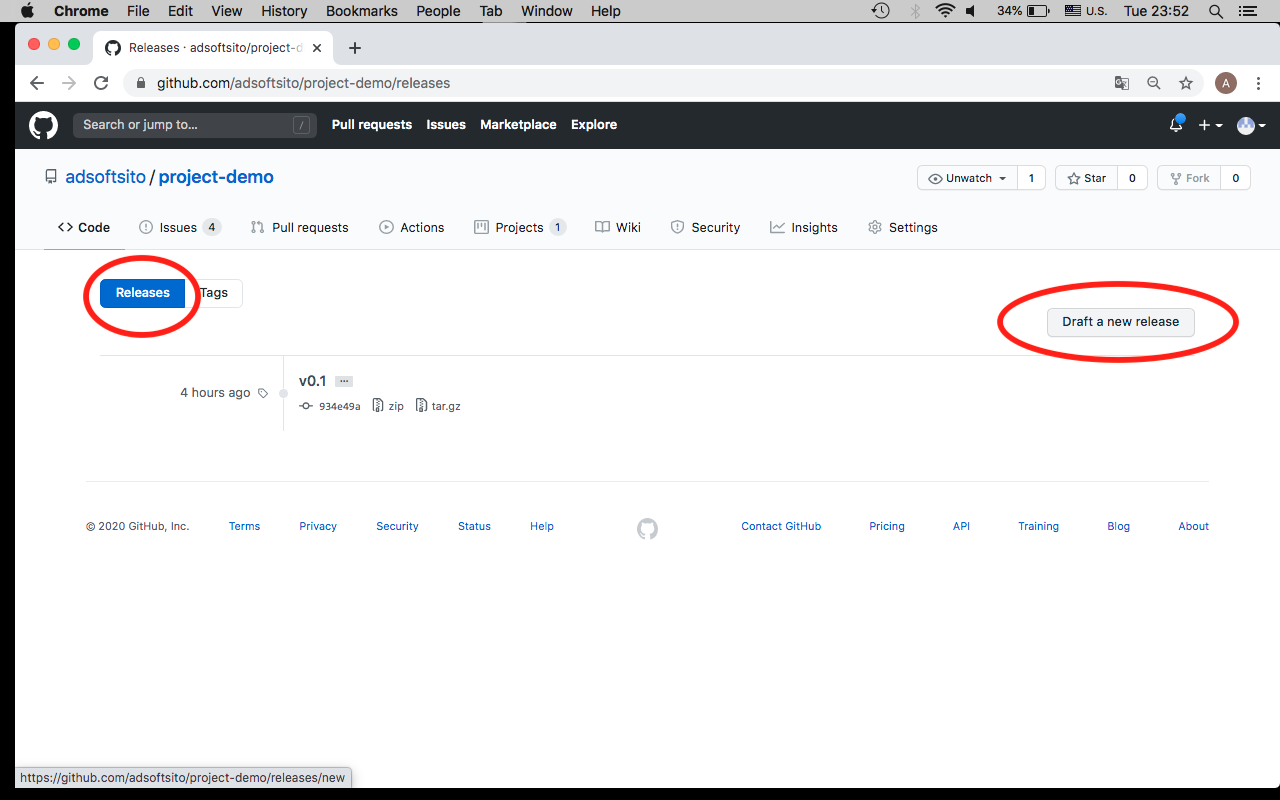


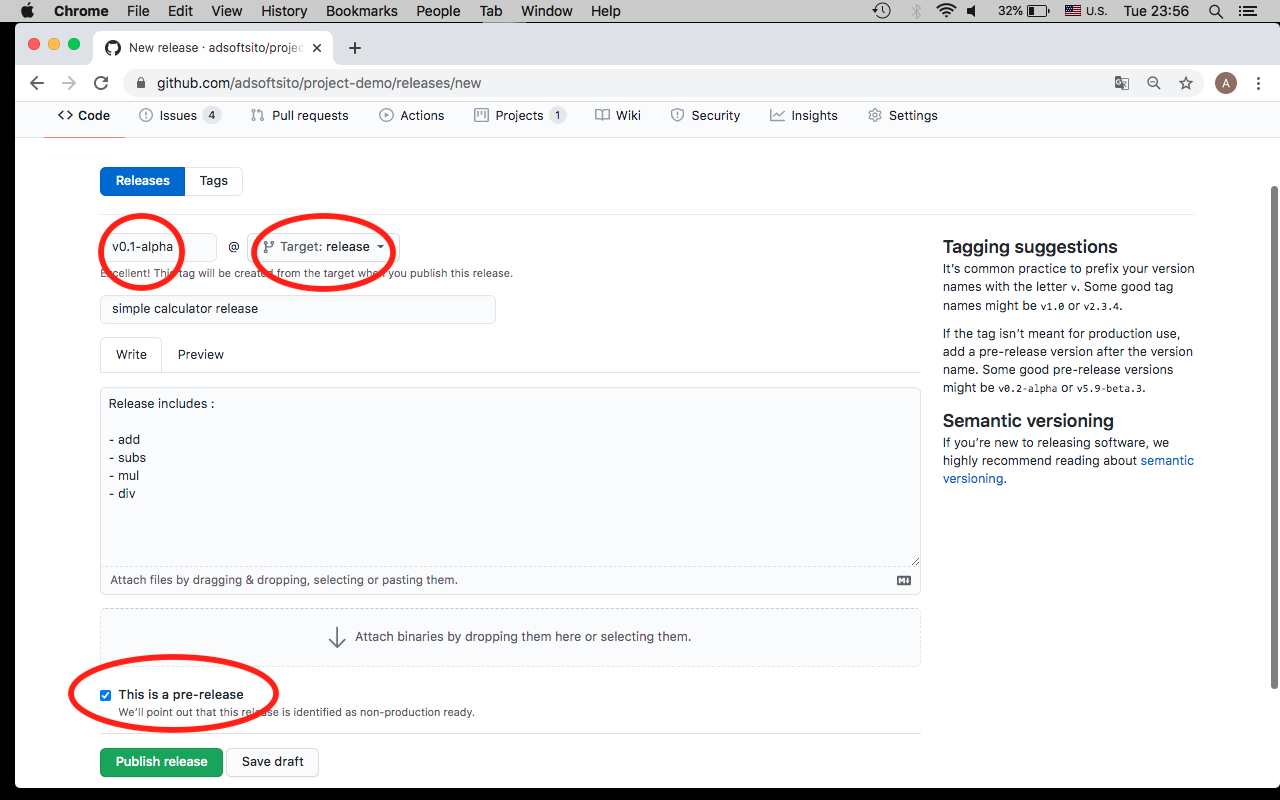
37.- Repeat this process for each issue

38.- Create a new PR from develop to release branch and confirm Merge.

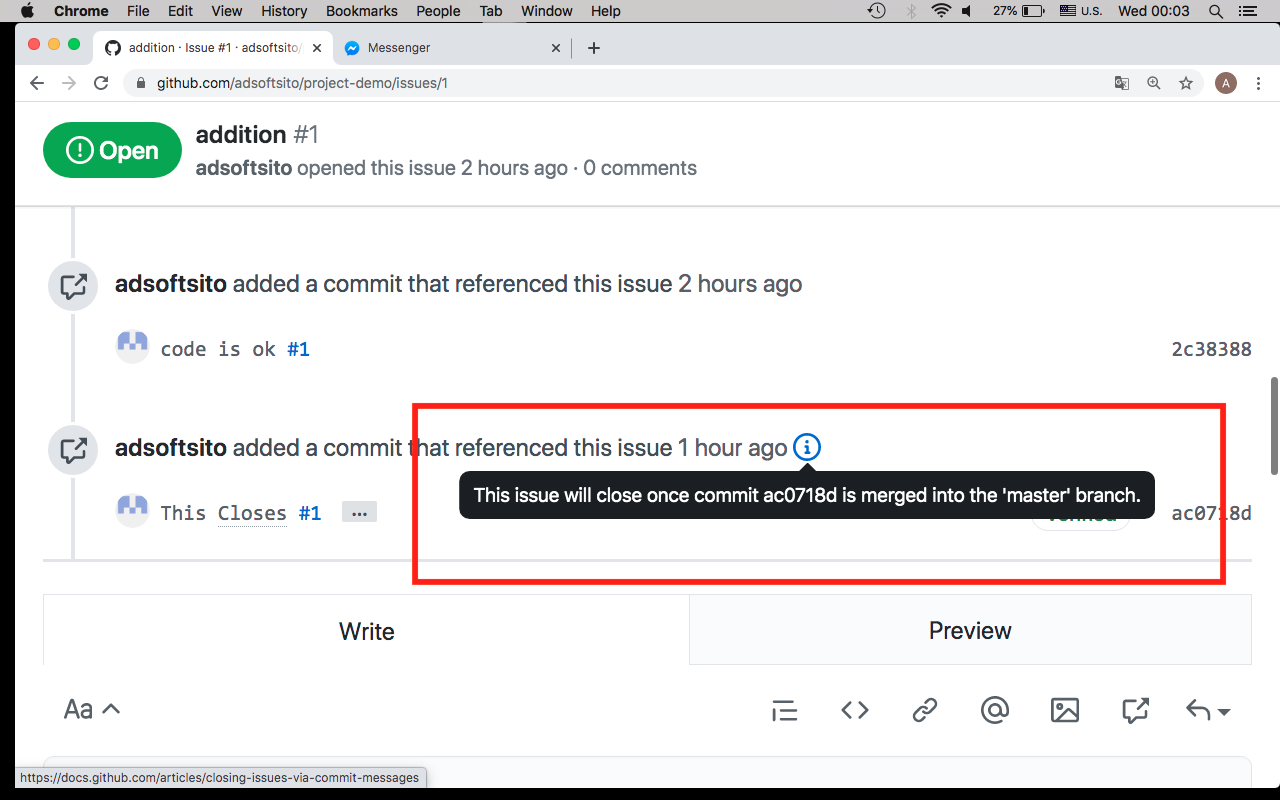


39.- Create a Release

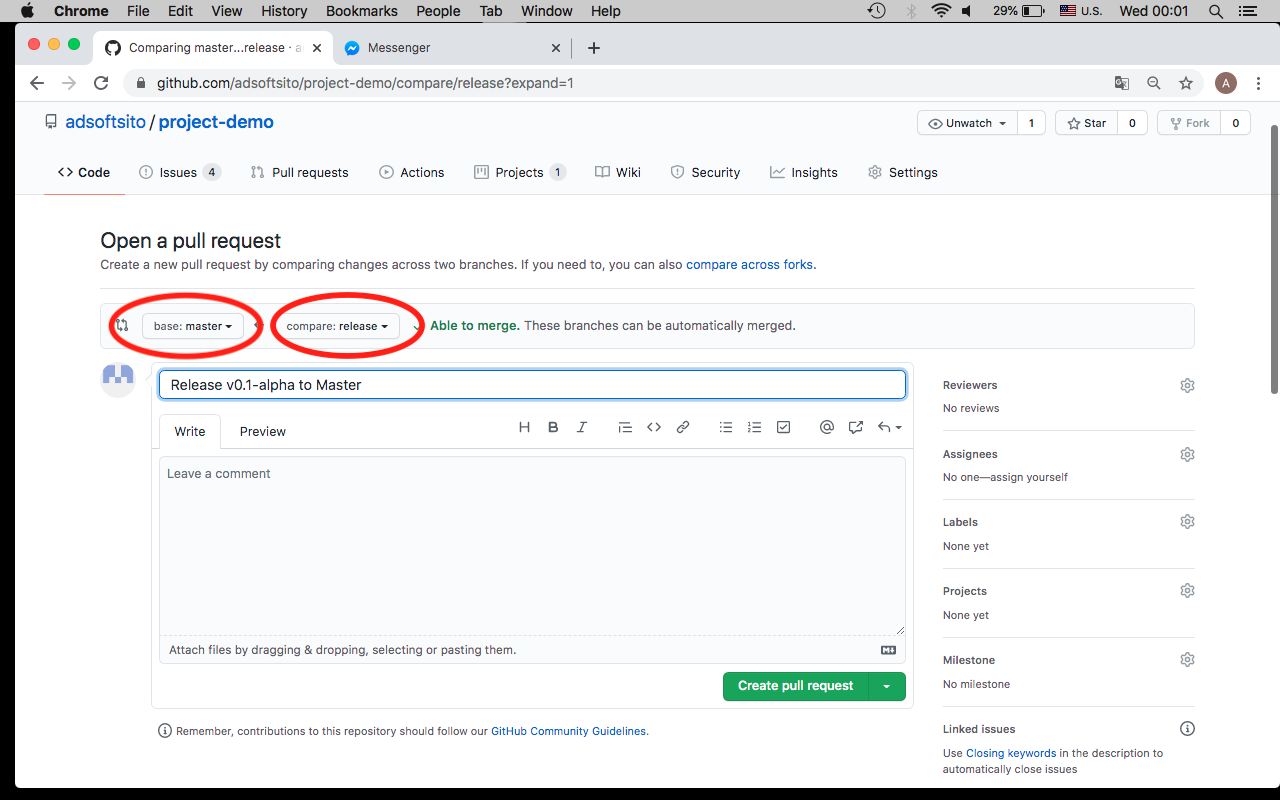




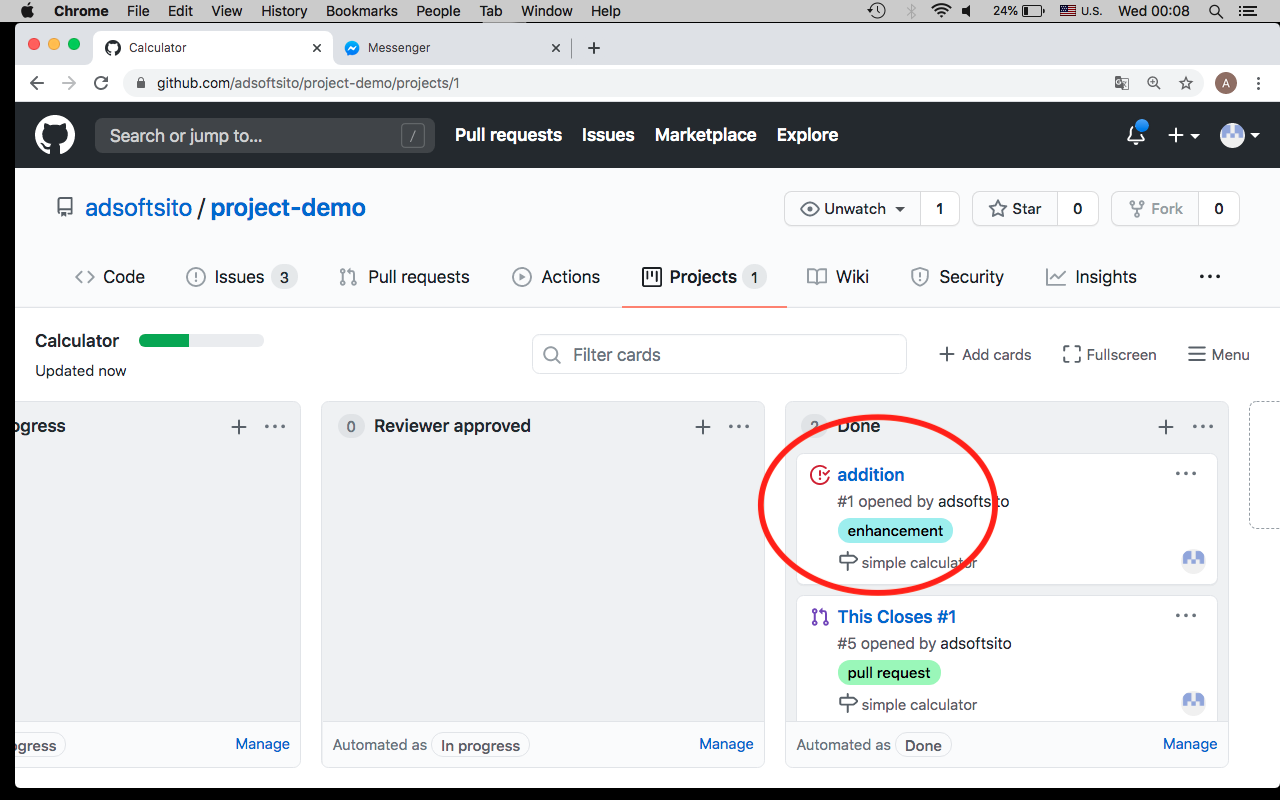
40.- Issues are “closed” but continues in “To do” column, check issue log.



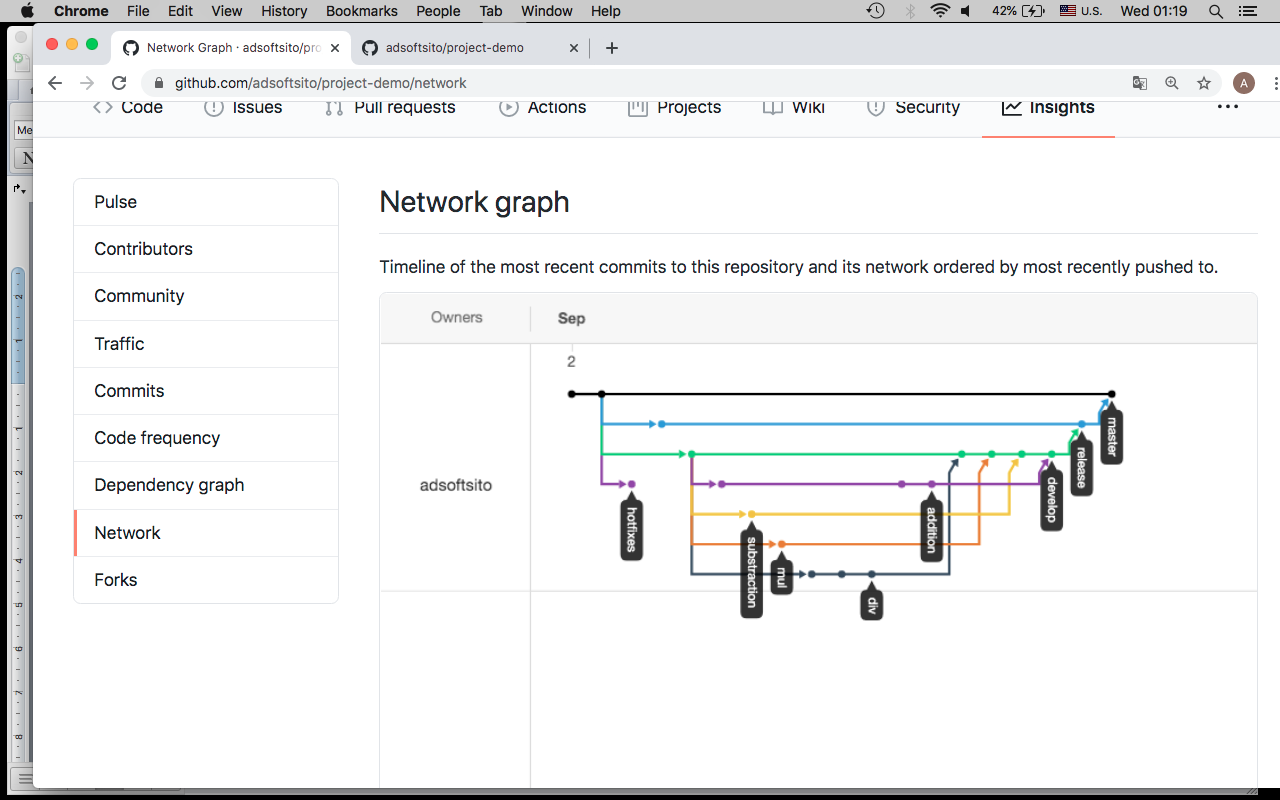
41.- Publish to Production, create PR from release to master branch



42.- Finally, issues are closed when code is merged in master branch



43.- Now, Check network graph

­