

For the Scrum masters

- Add the labels of the repository (High, Medium, Low, Spike, Setup, Size 1, Size 2, Size 3, Size 5, Size 8, Size 13, Size 21)
- add the milestones (sprint 0, sprint 1, sprint 2, etc)
- Also, change the pipelines of zenhub. Delete all existing pipelines and add Backlog, In progress, Under Review and Done.

The role of the reviewer:

Before you start with github, make sure you have been assigned a reviewer.

Every team member will have a peer reviewer. This reviewer will change every sprint. The reviewer does the following (a) test review (b) documentation and (c) design reviewer (d) scenario testing (acceptance)

Guidelines for documentation review:

1. check this documentation source: http://guides.rubyonrails.org/api_documentation_guidelines.html
2. Have the author, description, parameters or return
3. Clean/clear description of what the code does, the author and the parameters..
4. Internal documentation inside the methods is mostly needed when the code is complex or involves several steps. Otherwise, don't pollute the code. So, it is not recommended

Test review: each story should have a unit test, controller test and integration test. Please check these resources: <http://blog.stevensanderson.com/2009/08/24/writing-great-unit-tests-best-and-worst-practises/>
<https://github.com/colszowka/simplecov>

Design review: Compare the code with the class diagram. Is it in the right place? should it be in another class? is the logic redundant somehow? consult the team or the scrum master if you are confused.

Scenario review: Test the feature is working as if you are a user or a customer. Try to use edge cases and write the cases you used in the comment on the issue. Feel free to put extra comments on usability or on improving the feature. These comments can be optional and be scheduled for coming sprints; or they can be crucial and require not closing the issue to do more work.

Adding your issues:

0- Make sure you installed the Zenhub Chrome extension : <https://www.zenhub.io/>

1- you need to add your stories as issues on the github repo. Go to issues navigation page (ex, [https://github.com/tiec-eg/ \[your_repon_ame\]/issues](https://github.com/tiec-eg/[your_repon_ame]/issues)

2- Press on new issue. Then, add the title to be your story title. Then, add the tasks from your your sprint backlog, success and failure scenarios. And any other notes and attached your sketch file (scanned or in any other format). Everything related to this issue should be attached there or added in description clearly.

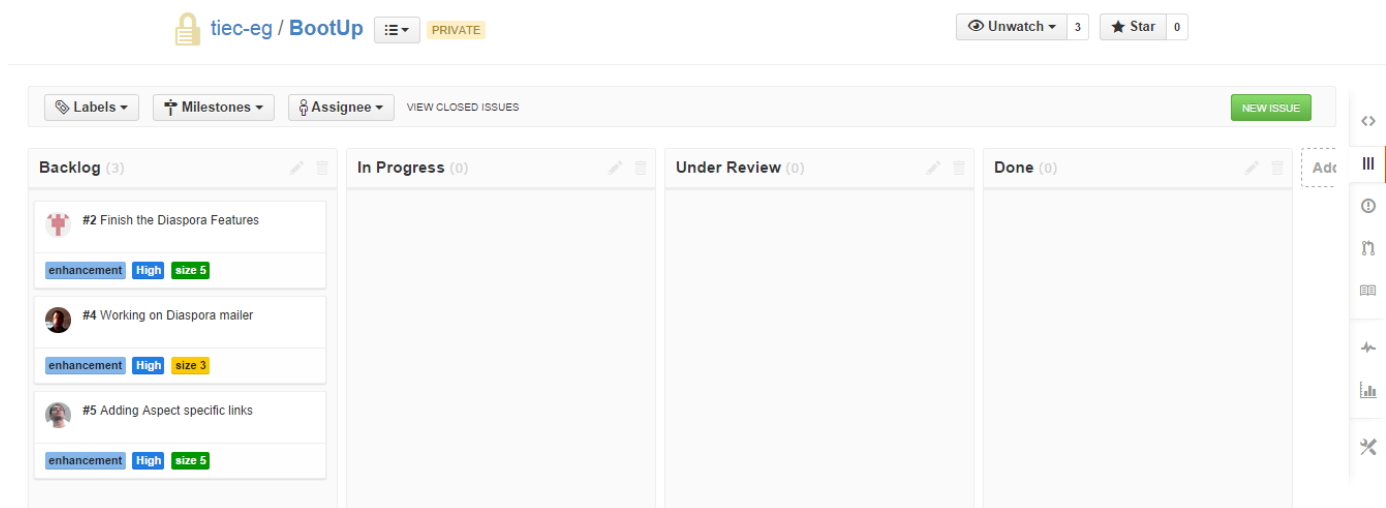
3- If what you are adding is a user story, then choose the label “enhancement” and then choose the label of priority of the story. You might want to add a bug or a task for someone else.

4- Make sure you choose the milestone (sprint). you will find a list created by the scrum master.

5- Choose the one assigned to this issue. We want each member to add his stories, so you will assign yourself. If you want to mention anyone in your comments, you can use the mention notation. However, the assignee is the one who will do the task. Sometimes, you might want to add a task and assign someone else for it (based on a prior discussion).

6- Choose the label reflecting the effort estimation you did for your story.

7- Make sure you you hook your commits with your issues using the ‘Issue #issue_number’ notation (**which is required**). More info about cool issues-related features [here](#). This will relate the right issue with the code and changes done with it.

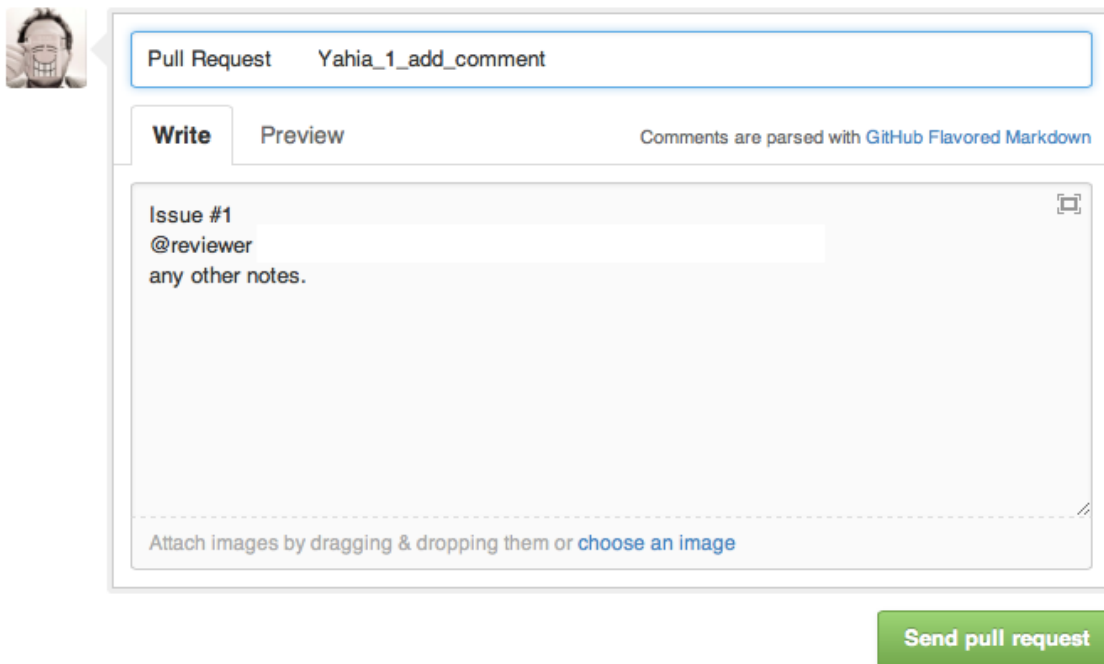


8- By default when you create an issue, it will go into the first pipeline (column) on your board. If you click on your board (on the right list of options), you will see your issues listed as below.

9 - When you start working on a story or any issue, on the board, drag it to the “in Progress” column. This is important so we can see you started working every time we check the board

When you are ready with Pull requests:

1- When you are ready to make a pull request make sure that you mention the issue, reviewers and add any note, for example :



The screenshot shows the GitHub Pull Request creation form. At the top, there's a header with a user profile picture on the left and a text box containing 'Pull Request' and 'Yahia_1_add_comment'. Below this, there are two tabs: 'Write' (active) and 'Preview'. To the right of the tabs, it says 'Comments are parsed with GitHub Flavored Markdown'. The main content area is a large text box with the following text: 'Issue #1', '@reviewer', and 'any other notes.'. At the bottom of the text box, there's a dashed line and the text 'Attach images by dragging & dropping them or [choose an image](#)'. Below the text box, there's a green button labeled 'Send pull request'.

2- After you do the pull request while mentioning the reviewer, make sure you drag the issue on the board to the “under review” column.

Reviewing:

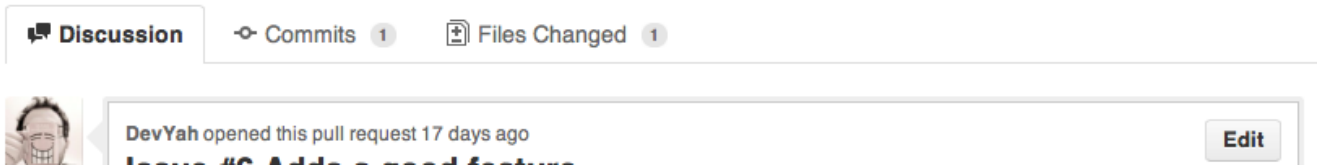
1- You will get a notification of mention for review. You need to give this a priority since your feedback might end up in extra work for the assignee.

2- If things are good and pull request can be automatically merged, merge it and drag the issue to “Done” column on the board and leave a comment.

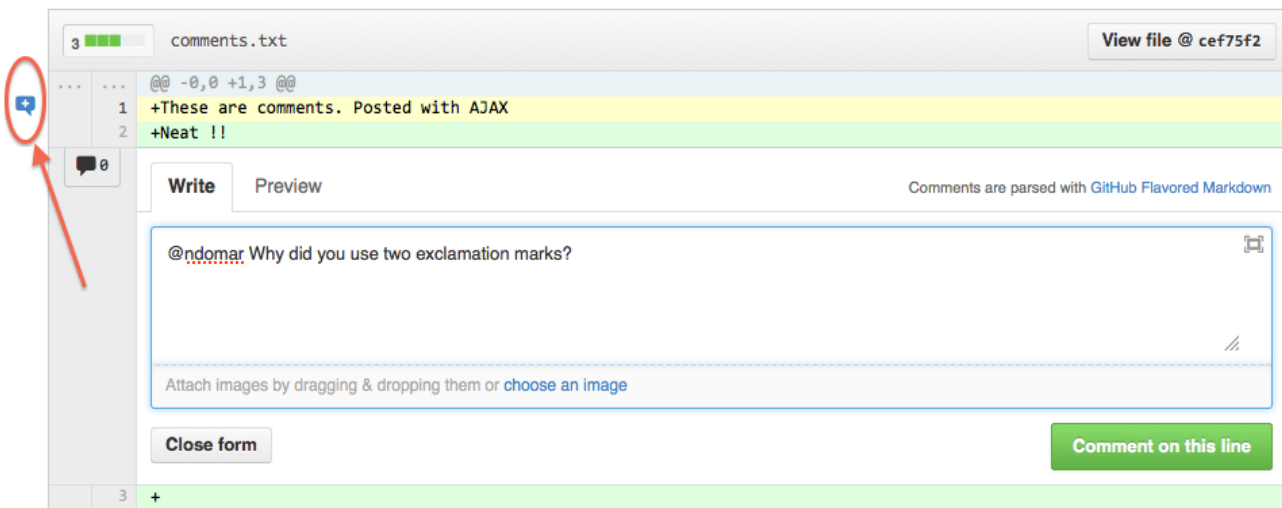
3- If things are not good and need modifications, then leave the proper comments on the issue and drag the issue back to “inprogress” on the board.

Note: In case of code reviewing, github offers inline comments. In the pull request page, there will be a tab

for file changed



press on in, it will redirect you the file and highlighting diffs (additions and deletions). Go to any line and press on the blue '+' sign on the right.



Related helpful Gems

<https://github.com/troessner/reek/wiki> For detecting code smells

<https://github.com/colszowka/simplecov> for code coverage

Definition of Done

- 1- Code written
- 2- Tests done (model and controller)
- 3- Reviews passed and merged by reviewer (including acceptance testing)
- 4- All related issues are closed.
- 5- In Sprint 1 - integration can be bonus ..while in other sprints it should be mandatory
- 6- Deployed bonus

Other types of issues:

- 1- Request change to Class diagram (from class diagram owner) (how will this reflect to DB migrations?)
- 2- Research tasks (spikes)
- 3- Data gathering and database population

During Sprints

- 3 meetings either offline or online. (attendance is a must starting Eid) - recommended to updated progress daily
- Three questions to be asked about the work in progress stories.
- After each meeting, every member should go inside the issue and write next to a task if it is done or not.
- Please put a link in github wiki to the Class diagram and product backlog (map or any other format).
- Please set a Pre-deadline (two days before sprint deadline to make sure all reviews are done).