Historically, we were unable to give direct merge access to non-Facebook employees, because every landed pull request also updates our copy of PyTorch in Facebook (which, understandably, requires you to be a Facebook employee).

Therefore, we are introducing the following alternate mechanism to allow you to merge pull requests: **to merge a pull request, comment on the PR "@pytorchbot merge this please"**. A helpful bookmarklet to make this command for you is: `javascript:(function(){document.getElementById("new_comment_field").value = "@pytorchbot merge this please"; document.getElementsByClassName('js-new-comment-form')[0]`

You can use this comment on your own PRs (if they have been accepted by someone else) or other people's PRs. Adding this label will ping an oncall at Facebook, who will merge this pull request on your behalf, with only cursory review (as needed by Facebook). During business hours, we hope that the turnaround time for merges will be within a few hours (which is similar to the turnaround time Facebook employees have.)

Here's what you need to know about merging to PyTorch master:

1. **Make sure CI is passing before labeling merge-now.** PyTorch's CI is a bit flaky, so if not all jobs are green, you should check the failing jobs and make sure they are actually spurious failures (the on-call will trust your judgment; but if there were real failures, the PR will get reverted later).
2. **You can directly edit pull requests.** The most useful application of this power is to merge someone's branch with master; if CI was broken at the time of the original pull request submission, merging to master may unbreak the patch. You can do this using `@pytorchbot rebase this please`, but you also have ability to push directly to the branch if you want to do a nontrivial merge resolution.
3. **You can label, assign and edit pull requests and issues.** Feel free to exercise this power as you see fit; in particular, don't feel shy about editing the title or description of someone's issue to make it clearer. Labeling is a bit ad hoc at the moment, but here are some labels that do get regular use:
    1. Topic labels (e.g., **jit**, **distributed**, **distributions**, **rocm**, **windows**, etc.) help surface subsystem specific issues to people who are best able to triage them.
    2. **bc-breaking** label can be used to identify an issue or pull request that can only be resolved in a backwards compatibility breaking way. This can be used to get more wider awareness of the change in question.
    3. **high priority** label is used to identify high priority issues that we should investigate ASAP.
    4. We use **[WIP]** in the title of pull requests to indicate that they are not to be reviewed. If you see a PR that is missing this, feel free to add the tag (and let the author of the PR know that they can remove the tag when their PR is ready.)

4. Part of being a maintainer is reviewing code. If you are looking for generalized advice in this aspect, consider taking a look at Code review values