Exploring Additional Slash Commands

In this lesson we will take a look at some additional slash commands and their usage.

We'll cover the following

- Creating a PR for experimentation
- The size label
- The /hold command
- The /hold cancel command
- The commands under lifecycle plugin
- The /meow command

We saw a few of the most commonly employed slash commands used through a typical pull request process. We'll expand on that next by creating a new PR and experimenting with a few other commands.

Creating a PR for experimentation

```
git checkout master

git pull

git checkout -b my-pr
```

We created a new branch called `my-pr`.

Next, we'll make a minor change to the source code and push it to the newly created branch. Otherwise, GitHub would not allow us to make a pull request if nothing changed.

```
echo "My PR" | tee README.md

git add .

git commit \
    --message "My second PR with prow"

git push --set-upstream origin my-pr
```

We are finally ready to create a pull request.

```
jx create pullrequest \
    --title "My PR" \
    --body "What I can say?" \
    --batch-mode
```

Please open the link from the output in your favorite browser.

You will notice that this time your colleague is automatically assigned as a reviewer. Prow took the list of reviewers from the `OWNERS` file and saw that there are only two available. Since you made the pull request, it decided to assign the other user as the reviewer. It wouldn't make sense to assign it to you anyways.

The `size` label

We can also observe that the system automatically added a label *size/XS*. It deduced that the changes we're proposing in this pull request are *extra small*. That is done through the Prow plugin `size`. While some plugins (e.g., `approve`) react to slash commands, others (e.g., `size`) are used as a reaction to other events (e.g., creation of a pull request).

Size labels are applied based on the total number of changed lines. Both new and deleted lines are counted, and the thresholds are as follows.

- size/XS: 0-9
- size/S: 10-29
- size/M: 30-99
- size/L: 100-499
- size/XL: 500-999
- size/XXL: 1000+

Since we rewrote `README.md` with a single line, the number of changed lines is one plus whatever was the number of lines in the file before we overwrote it. We know that up to nine lines were changed in total since we got the label *size/XS*.

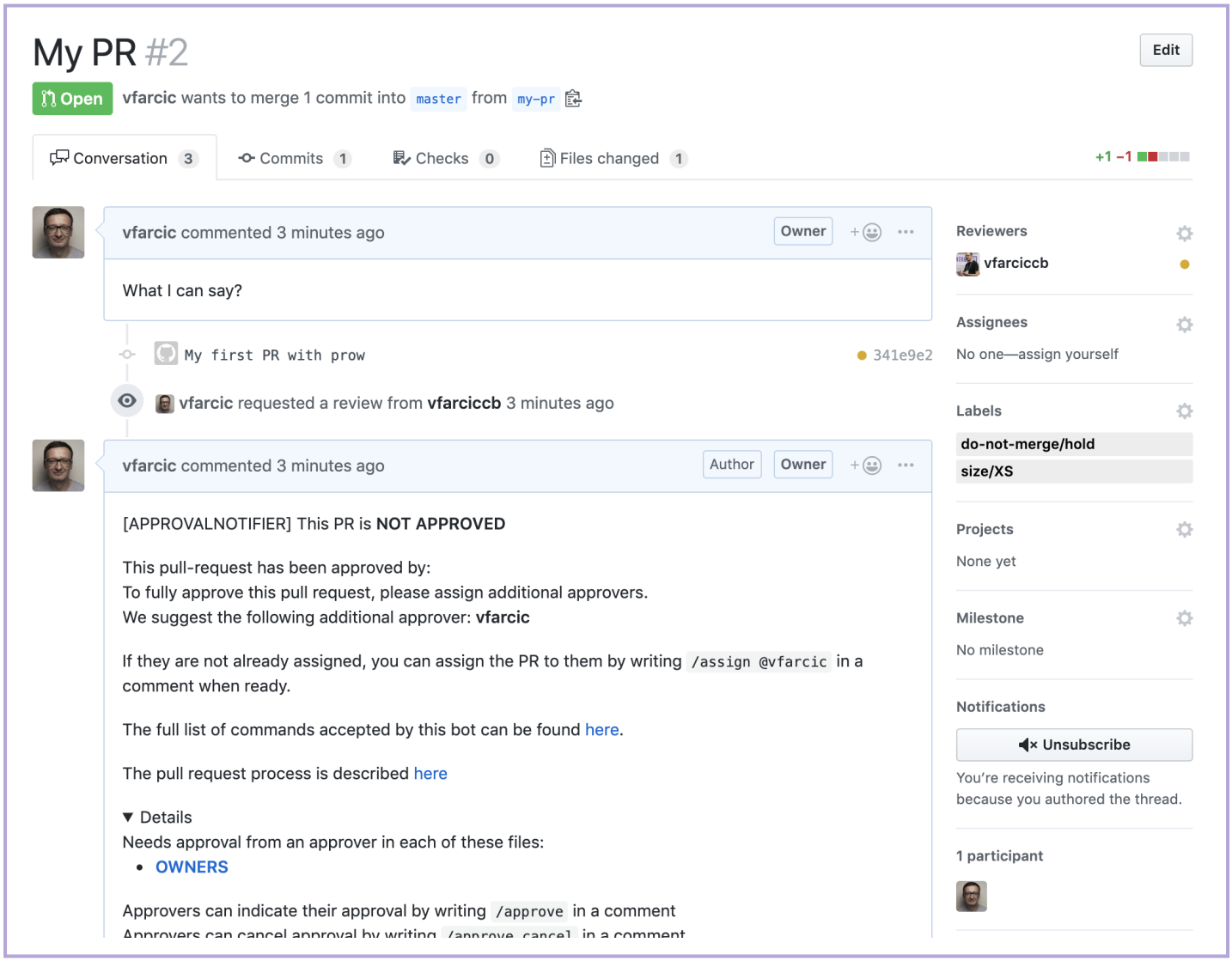
Sometimes we might use a code generator and might want to exclude the files it creates from the calculation. In such a case, all we'd need to do is place `.generated_files` to the project root.

The `/hold` command

The `/hold` command

Let's imagine that we reviewed the pull request and that we decided that we don't want to proceed in the near future. In such a case, we probably don't want to close it, but we still want to make sure that everyone is aware that it is on hold.

Please type `/hold` and click the *Comment* button. You'll see that the *do-not-merge/hold* label was added. Now it should be clear to everyone that the PR should not be merged. Hopefully, we would add a comment that would explain our reasoning. I will let your imagination kick in and let you compose it.



Pull request with Prow-created labels

At this moment, you might wonder if it's more efficient to create comments with slash commands that add labels rather than creating them directly. It is not. We could just as easily create a label using the GitHub console. But the reason we apply ChatOps principles is not only efficiency, instead, but it is also more focused on documenting and executing actions. When we write a comment with the `/hold` command, we not only created a label, but we also recorded who wrote the

comment and when it was done. The comments serve as a ledger and can be used to infer the flow of actions and decisions. We can find out everything that happened to a pull request by reading the comments from top to bottom.

The `/hold cancel` command

Now, let's say that the situation changed and that the pull request should not be on hold anymore. Many of the slash commands can be written with `cancel`. So, to remove the label, all we have to do is to write a new comment with `/hold cancel`. Try it out and confirm that the *do-not-merge/hold* label is removed.

The commands under *lifecycle* plugin

All commonly performed actions with pull requests are supported. We can, for example, `/close` and `/reopen` a pull request. Both belong to the *lifecycle* plugin which also supports `/lifecycle frozen`, `/lifecycle stale`, and `/lifecycle rotten` commands that add appropriate labels. To remove those labels, all we have to do is add `remove` as the prefix (e.g., `/remove-lifecycle stale`).

> ### The `/meow` command 
>
> Finally, if you want to bring a bit of life to your comments, try the `/meow` command.

Once you're done experimenting with the commands, please tell the approver to write a comment with the `/lgtm` command and the pull request will be merged to the master branch which, in turn, will initiate yet another Jenkins X build that will deploy a new release to the staging environment.

By now, you hopefully see the benefits of ChatOps, but you are probably wondering how to know which commands are available. We'll explore that next.