

---

**Buffy**

**Arfon Smith, Juanjo Bazán & the Open Journals community**

**Apr 12, 2023**



# GETTING STARTED

<b>1</b>	<b>Installation</b>	<b>3</b>
1.1	Create the bot GitHub user . . . . .	3
1.2	Deploy Buffy . . . . .	6
1.2.1	Server requirements . . . . .	6
1.2.2	Deployment . . . . .	6
1.3	Configure a webhook to send events from GitHub to Buffy . . . . .	7
<b>2</b>	<b>Configuration</b>	<b>9</b>
2.1	File Structure . . . . .	10
2.2	Env: General configuration settings . . . . .	10
2.3	Teams . . . . .	10
2.4	Responders . . . . .	11
2.4.1	Common options . . . . .	11
2.4.2	Multiple instances of the same responder . . . . .	13
<b>3</b>	<b>Available Responders</b>	<b>15</b>
3.1	Help . . . . .	15
3.1.1	Listens to . . . . .	15
3.1.2	Settings key . . . . .	15
3.1.3	Params . . . . .	15
3.1.4	Examples . . . . .	15
3.1.5	In action . . . . .	16
3.2	Hello . . . . .	17
3.2.1	Listens to . . . . .	17
3.2.2	Settings key . . . . .	17
3.2.3	Examples . . . . .	17
3.2.4	In action . . . . .	17
3.3	Basic command . . . . .	18
3.3.1	Listens to . . . . .	18
3.3.2	Settings key . . . . .	18
3.3.3	Params . . . . .	18
3.3.4	Examples . . . . .	19
3.3.5	In action . . . . .	19
3.4	Assign editor . . . . .	22
3.4.1	Listens to . . . . .	23
3.4.2	Requirements . . . . .	23
3.4.3	Settings key . . . . .	23
3.4.4	Params . . . . .	23
3.4.5	Examples . . . . .	23
3.4.6	In action . . . . .	24

3.5	Remove editor . . . . .	25
3.5.1	Listens to . . . . .	25
3.5.2	Requirements . . . . .	25
3.5.3	Settings key . . . . .	26
3.5.4	Params . . . . .	26
3.5.5	Examples . . . . .	26
3.5.6	In action . . . . .	26
3.6	Reviewers list . . . . .	28
3.6.1	Listens to . . . . .	28
3.6.2	Requirements . . . . .	28
3.6.3	Settings key . . . . .	28
3.6.4	Params . . . . .	28
3.6.5	Examples . . . . .	29
3.6.6	In action . . . . .	29
3.7	Invite . . . . .	30
3.7.1	Listens to . . . . .	30
3.7.2	Settings key . . . . .	30
3.7.3	Examples . . . . .	30
3.7.4	In action . . . . .	31
3.8	Set value . . . . .	31
3.8.1	Listens to . . . . .	31
3.8.2	Requirements . . . . .	31
3.8.3	Settings key . . . . .	32
3.8.4	Params . . . . .	32
3.8.5	Examples . . . . .	32
3.8.6	In action . . . . .	33
3.9	List of values . . . . .	34
3.9.1	Listens to . . . . .	34
3.9.2	Requirements . . . . .	34
3.9.3	Settings key . . . . .	34
3.9.4	Params . . . . .	35
3.9.5	Examples . . . . .	35
3.9.6	In action . . . . .	35
3.10	List team members . . . . .	37
3.10.1	Listens to . . . . .	37
3.10.2	Settings key . . . . .	37
3.10.3	Params . . . . .	37
3.10.4	Examples . . . . .	37
3.10.5	In action . . . . .	38
3.11	Add/Remove assignee . . . . .	38
3.11.1	Listens to . . . . .	38
3.11.2	Requirements . . . . .	38
3.11.3	Settings key . . . . .	39
3.11.4	Examples . . . . .	39
3.11.5	In action . . . . .	39
3.12	Reviewer checklist comment . . . . .	39
3.12.1	Listens to . . . . .	40
3.12.2	Requirements . . . . .	40
3.12.3	Settings key . . . . .	40
3.12.4	Params . . . . .	40
3.12.5	Examples . . . . .	40
3.12.6	In action . . . . .	41
3.13	Add/Remove checklist . . . . .	43
3.13.1	Listens to . . . . .	43

3.13.2	Requirements	43
3.13.3	Settings key	43
3.13.4	Params	43
3.13.5	Examples	43
3.13.6	In action	44
3.14	Label command	47
3.14.1	Listens to	47
3.14.2	Settings key	47
3.14.3	Params	47
3.14.4	Examples	47
3.14.5	In action	48
3.15	Check references	48
3.15.1	Listens to	48
3.15.2	Requirements	49
3.15.3	Settings key	49
3.15.4	Params	49
3.15.5	Examples	49
3.15.6	In action	50
3.16	Repository checks	52
3.16.1	Listens to	52
3.16.2	Requirements	53
3.16.3	Settings key	53
3.16.4	Params	53
3.16.5	Available checks	53
3.16.6	Examples	54
3.16.7	In action	55
3.17	Thanks	56
3.17.1	Listens to	56
3.17.2	Settings key	56
3.17.3	Params	56
3.17.4	Examples	56
3.17.5	In action	57
3.18	Reminders	57
3.18.1	Listens to	57
3.18.2	Settings key	57
3.18.3	Params	57
3.18.4	Examples	58
3.18.5	In action	58
3.19	Initial values	59
3.19.1	Listens to	59
3.19.2	Settings key	59
3.19.3	Params	59
3.19.4	Examples	59
3.19.5	In action	60
3.20	Welcome	62
3.20.1	Listens to	62
3.20.2	Settings key	62
3.20.3	Requirements	63
3.20.4	Params	64
3.20.5	Examples	65
3.20.6	In action	66
3.21	Goodbye	69
3.21.1	Listens to	69
3.21.2	Settings key	69

3.21.3	Requirements	69
3.21.4	Params	69
3.21.5	Examples	70
3.21.6	In action	71
3.22	Close issue command	72
3.22.1	Listens to	72
3.22.2	Settings key	73
3.22.3	Params	73
3.22.4	Examples	73
3.22.5	In action	73
3.23	Update comment	74
3.23.1	Listens to	74
3.23.2	Requirements	74
3.23.3	Settings key	74
3.23.4	Params	74
3.23.5	Examples	74
3.23.6	In action	75
3.24	External start review	76
3.24.1	Listens to	76
3.24.2	Requirements	76
3.24.3	Settings key	76
3.24.4	Params	76
3.24.5	Examples	76
3.24.6	In action	77
3.25	External service	77
3.25.1	Listens to	77
3.25.2	Requirements	77
3.25.3	Settings key	78
3.25.4	Params	78
3.25.5	Examples	79
3.25.6	In action	80
3.26	GitHub Action	82
3.26.1	Listens to	82
3.26.2	Requirements	82
3.26.3	Settings key	82
3.26.4	Params	83
3.26.5	Examples	84
3.27	Wrong command	84
3.27.1	Listens to	84
3.27.2	Settings key	84
3.27.3	Params	85
3.27.4	Examples	85
3.27.5	In action	86
3.28	ROpenSci :: Reviewers & due date	86
3.28.1	Listens to	86
3.28.2	Requirements	86
3.28.3	Settings key	87
3.28.4	Params	87
3.28.5	Examples	88
3.28.6	In action	88
3.29	ROpenSci :: Set due date	90
3.29.1	Listens to	90
3.29.2	Requirements	90
3.29.3	Settings key	91

3.29.4	Examples	91
3.29.5	In action	91
3.30	ROpenSci :: Seeking reviewers	92
3.30.1	Listens to	92
3.30.2	Settings key	92
3.30.3	Params	93
3.30.4	Examples	93
3.30.5	In action	94
3.31	ROpenSci :: Approve	94
3.31.1	Listens to	95
3.31.2	Requirements	95
3.31.3	Settings key	95
3.31.4	Params	95
3.31.5	Examples	95
3.32	ROpenSci :: Finalize transfer	96
3.32.1	Listens to	96
3.32.2	Requirements	96
3.32.3	Settings key	96
3.32.4	Example:	97
3.33	ROpenSci :: Invite author	97
3.33.1	Listens to	97
3.33.2	Requirements	97
3.33.3	Settings key	97
3.33.4	Example:	97
3.33.5	In action	98
3.34	ROpenSci :: Mint	98
3.34.1	Listens to	98
3.34.2	Requirements	98
3.34.3	Settings key	98
3.34.4	Examples	99
3.34.5	In action	99
3.35	ROpenSci :: Submit review	100
3.35.1	Listens to	100
3.35.2	Requirements	100
3.35.3	Settings key	100
3.35.4	Params	101
3.35.5	Examples	101
3.35.6	In action	102
3.36	ROpenSci :: Submit author response	102
3.36.1	Listens to	102
3.36.2	Requirements	103
3.36.3	Settings key	103
3.36.4	Params	103
3.36.5	Examples	103
3.36.6	In action	103
3.37	ROpenSci :: On hold	104
3.37.1	Listens to	104
3.37.2	Settings key	104
3.37.3	Params	104
3.37.4	Example:	105
3.37.5	In action	105
<b>4</b>	<b>Labeling</b>	<b>107</b>
4.1	Settings	107

4.2	Responders listening to Add/Remove actions . . . . .	107
<b>5</b>	<b>Using templates</b>	<b>109</b>
5.1	Template files . . . . .	109
5.1.1	Location . . . . .	109
5.1.2	Name . . . . .	109
5.1.3	Example . . . . .	109
5.2	Populating templates . . . . .	110
<b>6</b>	<b>Creating a custom responder</b>	<b>111</b>
6.1	Responder structure . . . . .	111
6.1.1	The Responder Ruby class . . . . .	111
6.1.2	Keyname . . . . .	112
6.1.3	Define listening . . . . .	112
6.1.4	Process message . . . . .	114
6.1.5	Description . . . . .	115
6.1.6	Example invocation . . . . .	115
6.2	Sample custom responder . . . . .	115
6.3	Tests . . . . .	117



Buffy is an editorial bots generator, a service to provide a bot helping scientific journals manage submission reviews.

Buffy is a configurable Ruby application that –once deployed as a web service listening to incoming GitHub webhooks– provides a bot that interacts during the peer-review process with editors, reviewers and authors to help them perform actions on the review, the software being reviewed and its corresponding paper, automating common editorial tasks like those needed by the [Journal of Open Source Software](#), [rOpenSci](#) or [Scipy](#).

Buffy is an Open Source project, [the code](#) is hosted at GitHub and released under a MIT license.



## **INSTALLATION**

Buffy works listening to events received from GitHub and deciding if/how to reply by passing the received payload to different Responders. You can fork Buffy and configure the responders you want to use for any specific repository. There is no need for the Buffy fork to be located in the same GitHub user/organization as the repo where it will be used. To have Buffy ready and listening to events you can install it locally or deploy it to a server or service platform. You'll need the following components:

- A GitHub user to use as the bot with admin permissions on the target repo (usually a member of the organization owning the repo).
- An instance of Buffy running
- A webhook configured in the GitHub repo's settings to send events to Buffy

### **1.1 Create the bot GitHub user**

This will be the user responding to commands in the reviews repo.

1. [Sign up at GitHub](#) and create the bot user:



Join GitHub

# Create your account

Username \*

botsci



botsci is available.

Password \*

.....


Make sure it's **at least 15 characters** OR at least 8 characters including a number and a lowercase letter. [Learn more.](#)

Email preferences




GitHub's

signup page

2. Go to Settings >> Developer settings >> Personal access tokens and create a new token with these scopes: `public_repo`, `repo:invite`, `read:org`, `read:user` and, if using the approval responder, `admin:org` (as the app will create teams and invite members to the ropensci organization). Save that token, it will be your `BUFFY_GH_ACCESS_TOKEN`.



[Pull requests](#)
[Issues](#)
[Marketplace](#)
[Explore](#)

[Settings](#) / [Developer settings](#)

[GitHub Apps](#)
[OAuth Apps](#)
[Personal access tokens](#)

## New personal access token

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

**Note**

token for buffy testing

What's this token for?

**Select scopes**

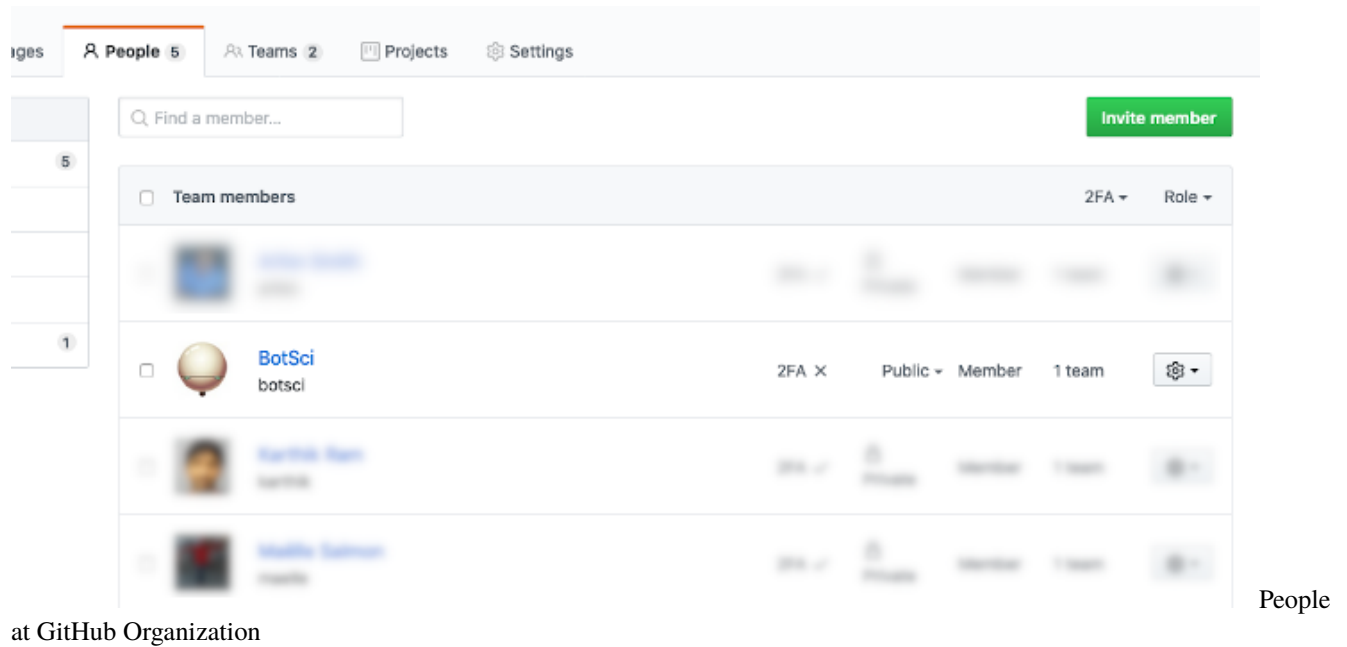
Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

<input type="checkbox"/> <b>repo</b>	Full control of private repositories
<input type="checkbox"/> repo:status	Access commit status
<input type="checkbox"/> repo_deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input checked="" type="checkbox"/> repo:invite	Access repository invitations
<input type="checkbox"/> security_events	Read and write security events
<input type="checkbox"/> write:packages	Upload packages to github package registry
<input type="checkbox"/> read:packages	Download packages from github package registry
<input type="checkbox"/> delete:packages	Delete packages from github package registry
<input type="checkbox"/> <b>admin:org</b>	Full control of orgs and teams, read and write org projects
<input type="checkbox"/> write:org	Read and write org and team membership, read and write org projects
<input checked="" type="checkbox"/> read:org	Read org and team membership, read org projects
<input type="checkbox"/> <b>admin:public_key</b>	Full control of user public keys
<input type="checkbox"/> write:public_key	Write user public keys
<input type="checkbox"/> read:public_key	Read user public keys
<input type="checkbox"/> <b>admin:repo_hook</b>	Full control of repository hooks
<input type="checkbox"/> write:repo_hook	Write repository hooks
<input type="checkbox"/> read:repo_hook	Read repository hooks
<input type="checkbox"/> <b>admin:org_hook</b>	Full control of organization hooks
<input type="checkbox"/> gist	Create gists
<input type="checkbox"/> notifications	Access notifications
<input type="checkbox"/> <b>user</b>	Update all user data
<input checked="" type="checkbox"/> read:user	Read all user profile data

Settings

>> Developer settings >> Personal access tokens

3. Give the bot admin permissions: add it as member of the organization owning the repo where the reviews will take place:



at GitHub Organization

## 1.2 Deploy Buffy

### 1.2.1 Server requirements

Some applications and services must be available to use by Buffy:

- **Redis:** To process background jobs Buffy needs `redis` installed.
- **Gitinspector:** The *Respository Checks Responder* performs a statistical analysis using it.
- **cloc:** The *Respository Checks Responder* can analyze source code, to run this check `cloc` is used.

### 1.2.2 Deployment

We will use here [Heroku](#) as an example service to deploy Buffy but you can use any other server or platform.

1. Create a new app in heroku linked to the url of your fork of Buffy. Automatically Heroku will use the `heroku/ruby` buildpack.

- To process background jobs Buffy needs `redis` installed, several add-ons providing it are available: Heroku Redis, RedisGreen, Redis To Go, etc.
- To install the `cloc` dependency there's a buildpack for Heroku available [here](#).
- Gitinspector can be installed [using npm](#). To do so in Heroku, the `heroku/nodejs` buildpack can be added.

2. In the app settings add the following Config Vars:

```

BUFFY_BOT_GH_USER: <the_github_username_of_the_bot>
BUFFY_GH_ACCESS_TOKEN: <the_access_token_for_the_bot_created_in_the_previous_step>
BUFFY_GH_SECRET_TOKEN: <a_random_string>
RACK_ENV: production
    
```

**2b.** You can set the Ruby version to install using the `CUSTOM_RUBY_VERSION` env var. Unless you need any other specific version, please add also a Config Var named `CUSTOM_RUBY_VERSION` with the value of the latest version listed in the [Buffy tested Ruby versions](#).

**3.** You can set Heroku to automatically redeploy when new commits are added. You can also add heroku as a git remote and deploy manually using

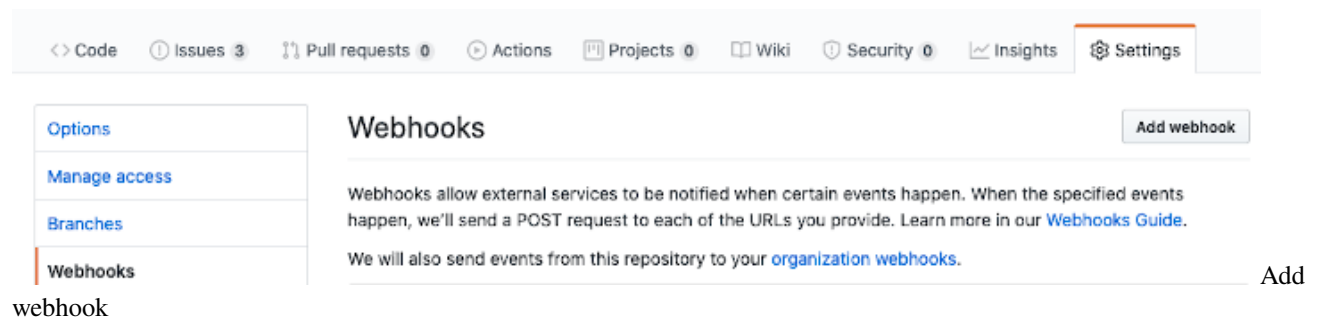
```
$ git push heroku main
```

There are detailed instructions in the Deploy section of your Heroku app.

**4.** You should now have a public URL with Buffy running. You can test that pointing your browser to `/status`, like for example: `https://your-new-buffy-deploy.herokuapp.com/status` It should return a simple *up and running* message.

## 1.3 Configure a webhook to send events from GitHub to Buffy

**1.** Go to the settings page of the repository where you want to use buffy. Add a new webhook.



The screenshot shows the GitHub repository settings page for 'Webhooks'. The 'Settings' tab is selected in the top navigation bar. On the left sidebar, 'Webhooks' is highlighted under the 'Options' section. The main content area is titled 'Webhooks' and includes an 'Add webhook' button. Below the title, there is explanatory text about webhooks and a link to the 'Webhooks Guide'. At the bottom right of the main content area, there is an 'Add' button.

**2.** Configure the new webhook with:

```
Payload URL: /dispatch path at your public buffy url
Content type: application/json
Secret: The BUFFY_GH_SECRET_TOKEN you configured in the previous step
```

<> Code 1 Issues 3 Pull requests 0 Actions Projects 0 Wiki

Options

Manage access

Branches

Webhooks

Notifications

Integrations

Deploy keys

Secrets

Actions

Dependabot alerts

Moderation

Interaction limits

Reported content

Webhooks / Manage webhook

We'll send a POST request to the URL below with details in the data format you'd like to receive (JSON, x-www-form-urlencoded, etc.). See [developer documentation](#).

Payload URL \*  
https://MY\_URL\_WHERE\_BUFFY\_IS\_RUNNING/dispatch

Content type  
application/json

Secret  
\*\*\*\*\* — Edit

SSL verification  
By default, we verify SSL certificates when delivering payloads.  
☒ Enable SSL verification ☐ Disable (not recommended)

Which events would you like to trigger this webhook?  
☐ Just the push event.  
☐ Send me everything.  
☒ Let me select individual events.  

☐ Check runs  
Check run is created, requested, rerequested, or completed.

☐ Commit comments  
Commit or diff commented on.

☐ Branch or tag deletion  
Branch or tag deleted.

☐ Deployments  
Repository was deployed or a deployment was deleted.

☐ Forks  
Repository forked.

☒ Issue comments  
Issue comment created, edited, or deleted.

Select individual events to trigger: **issue comments** and **issues** webhook

If everything went well you should have now your bot responding on the reviews issues. Try `@botname help` for example.



## CONFIGURATION

Buffy is configured using a simple YAML file containing all the settings needed. The settings file is located in the `/config` dir and is named `settings-<environment>.yaml`, where `<environment>` is the name of the environment Buffy is running in, usually set via the `RACK_ENV` env var. So for a Buffy instance running in production mode, the configuration file will be `/config/settings-production.yaml`

A sample settings file will look similar to this:

```
buffy:
  env:
    bot_github_user: <%= ENV['BUFFY_BOT_GH_USER'] %>
    gh_access_token: <%= ENV['BUFFY_GH_ACCESS_TOKEN'] %>
    gh_secret_token: <%= ENV['BUFFY_GH_SECRET_TOKEN'] %>
  teams:
    editors: 3824115
    eics: myorg/editor-in-chief-team
  responders:
    help:
    hello:
      hidden: true
    assign_editor:
      only: editors
    remove_editor:
      only: editors
      no_editor_text: "TBD"
    list_of_values:
      - reviewers:
          only: editors
          if:
            role_assigned: editor
            reject_msg: "Can't assign reviewer because there is no editor assigned for_
↪this submission yet"
            sample_value: "@username"
            add_as_assignee: true
    invite:
      only: eics
    set_value:
      - version:
          only: editors
          sample_value: "v1.0.0"
      - archive:
          only: editors
```

(continues on next page)

(continued from previous page)

```
sample_value: "10.21105/joss.12345"
welcome:
```

## 2.1 File Structure

The structure of the settings file starts with a single root node called `buffy`. It contains three main parts:

- The `env` node
- The `teams` node
- The `responders` node

A detailed description of all of them:

## 2.2 Env: General configuration settings

```
env:
  bot_github_user: <%= ENV['BUFFY_BOT_GH_USER'] %>
  gh_access_token: <%= ENV['BUFFY_GH_ACCESS_TOKEN'] %>
  gh_secret_token: <%= ENV['BUFFY_GH_SECRET_TOKEN'] %>
  templates_path: ".templates"
```

The `env` section is used to declare general key/value settings. For security reasons is a good practice to load the secret values from your environment instead of hardcoding them in the code.

## 2.3 Teams

```
teams:
  editors: 3824117
  eics: myorg/editor-in-chief-team
  reviewers: 45363564
  collaborators:
    - user33
    - user42
```

The optional `teams` node includes entries to reference GitHub teams, used later to grant access to responders only to users belonging to specific teams. The teams referred here must be **visible** teams of the organization owner of the repositories where the reviews will take place. Multiple entries can be added to the `teams` node. All entries follow this simple format:

## 2.4 Responders

```
responders:
  help:
  hello:
    hidden: true
  assign_reviewers:
    only: editors
```

The responders node lists all the responders that will be available. The key for each entry is the name of the responder and nested under it the configuration options for that responder are declared.

### 2.4.1 Common options

All the responders share some options available to all of them. They can also have their own particular configurable parameters (see *docs for each responder*). The common parameters are:

Usage:

```
...
secret_responder:
  hidden: true
...
```

Usage:

```
public_responder:
available_for_one_team_responder:
  only: editors
available_for_two_teams_responder:
  only:
    - editors
    - reviewers
```

Usage:

```
public_responder:
restricted_responder:
  only: editors
  authorized_roles_in_issue:
    - author-handle
    - reviewers-list
```

(restricted\_responder can only be called by members of the editors team and by users listed in the issue in the author-handle and reviewers-list HTML-marked fields)

#### title

<String> or <Regular Expression> Responder will run only if issue' title matches this.

#### body

<String> or <Regular Expression> Responder will run only if the body of the issue matches this.

#### value\_exists

<String> Responder will run only if there is a not empty value for this in the issue (marked with HTML comments).

**value\_matches**

<Hash> Responder will run only if the param values (marked with HTML comments) in the body of the issue matches the ones specified here.

**role\_assigned**

<String> Responder will be run only if there is a username assigned for the specified value.

**labels**

<Array> Responder will be run only if the issue is labeled with all the labels listed here.

**reject\_msg**

<String> Optional. The response to send as comment if the conditions are not met

Usage:

```
# This responder should be invoked only if there's an editor assigned
# otherwise will reply with a custom "no editor assigned yet" message
assign_reviewer:
  if:
    role_assigned: editor
    reject_msg: I can not do that because there is no editor assigned yet

# This responder will run only if issue title includes '[PRE-REVIEW]' and if
# there is a value for repo-url, ie: <!--repo-url-->whatever<!--end-repo-url-->
start_review:
  if:
    title: "^\\[PRE-REVIEW\\]"
    value_exists: repo-url

# This responder will run only if the value for submission_type in the body of
# the issue matches 'astro', ie: <!--submission_type-->astro<!--end-submission_type-->
start_review:
  if:
    value_matches:
      submission_type: astro

# This responder will run only if issue title includes '[REVIEW]' and
# the issue is labeled as 'accepted'
start_review:
  if:
    title: "^\\[REVIEW\\]"
    labels:
      - accepted
```

Usage:

```
...
custom_responder:
  description: "This responder do something"
...
```

Usage:

```
...
custom_responder:
  example_invocation: "@botname run performance checks (please run this only on_
```

(continues on next page)

(continued from previous page)

```
↪mondays)"
...

```

A complete example:

```
# Two responders are configured here:
#
# The assign_reviewers responder will respond only when triggered by a user that is
# member of any of the editors or editors-in-chief teams. It will also respond only
# in issues with the text "[REVIEW]" in its title and that have a not empty value
# in its body marked with HTML comments: <!--editor-->@EDITOR_HANDLE<!--end-editor-->
# Once invoked, it will label the issue with the 'reviewers-assigned' label.
#
# The hello responder is configured as hidden, so when calling the help responder the
# description and usage example of the hello responder won't be listed in the response.
...
responders:
  assign_reviewers:
    only:
      - editors
      - editors-in-chief
    if:
      title: "^\\[REVIEW\\]"
      role_assigned: editor
    add_labels:
      - reviewers-assigned
    description: "Use this command to assign a reviewers once the editor is assigned"
    hello:
      hidden: true
    ...

```

Several responders also allow *adding or removing labels*.

## 2.4.2 Multiple instances of the same responder

Sometimes you want to use a responder more than once, with different parameters. In that case under the name of the responder you can declare an array of instances, and the key for each instance will be passed to the responder as the name parameter.

Example:

The `set_value` responder uses a name param to change the value to a variable. If declared in the settings file like this:

```
responders:
  set_value:
    name: version

```

It could be invoked with `@botname set 1.0 as version`.

If you want to use the same responder to change version but also to allow editors to change url you would declare multiple instances in the settings file like this:

```
responders:  
  set_value:  
    - version:  
    - url:  
      only: editors
```

Now `@botname set 1.0 as version` is a public command and `@botname set abcd.efg as url` is a command available to editors.

## AVAILABLE RESPONDERS

Buffy listens to events in the target repo using responders. Every responder is a subclass of the `Responder` class. Each responder have a `define_listening` method where the action and/or regex the responder is listening to are defined. The actions a responder takes if called are defined in the `process_message` method.

Buffy includes a list of Responders that can be used by configuring them in the YAML settings file.

### 3.1 Help

The help responder provides a customizable command to list all the available actions. It only lists options available to the user triggering the responder and only responders not marked as `hidden`.

#### 3.1.1 Listens to

```
@botname help
```

`help` is the default command, but it is customizable via params.

#### 3.1.2 Settings key

`help`

#### 3.1.3 Params

**help\_command**

*Optional.* The command triggering this responder. Default value is **help**.

#### 3.1.4 Examples

Simplest use case:

```
...
responders:
  help:
...


```

Custom command:

```
...  
responders:  
  help:  
    help_command: commands  
...
```


Now it will reply to `@botname commands`.

### 3.1.5 In action



karthik commented on May 15

@botsci help



botsci commented on May 15

Hello @karthik, here are the things you can ask me to do:  
  

```
# List all available commands  
@botsci help  
  
# Say hi!  
Hello @botsci  
  
# You are welcome  
Thanks @botsci!  
  
# Assign a user as the reviewer N of this submission (where N=1,2...)  
@botsci assign @username as reviewer 2  
  
# Remove the user assigned as reviewer N of this submission (where N=1,2...)  
@botsci remove reviewer 2  
  
# Assign a user as the editor of this submission  
@botsci assign @username as editor  
  
# Remove the editor assigned to this submission  
@botsci remove editor
```



## 3.2 Hello

A simple responder to reply to user greetings.

### 3.2.1 Listens to

```
Hi @botname
```

```
Hello @botname
```

### 3.2.2 Settings key

```
hello
```

### 3.2.3 Examples

Simplest use case:

```
...  
  responders:  
    hello:  
...
```

Hidden from public commands list

```
...  
  responders:  
    hello:  
      hidden: true  
...
```

### 3.2.4 In action



xuanxu commented 2 minutes ago

Hello @botsci



botsci commented 2 minutes ago

Hi!

## 3.3 Basic command

This responder defines a custom command and replies with text messages, optionally *using a template*. Allows *labeling*.

### 3.3.1 Listens to

```
@botname <command>
```

For example, if you configure the command to be *list editors*, it would respond to:

```
@botname list editors
```

### 3.3.2 Settings key

basic\_command

### 3.3.3 Params

**command**

The command this responder will listen to.

**description**

*Optional* String to show when the help command is invoked.

**example\_invocation**

*Optional* String to show as an example of the command being used when the help command is invoked.

**message**

*Optional* A text message to use as reply.

**messages**

*Optional* <Array> A list of text messages to respond with.

**template\_file**

*Optional* A template file to use to build the response message.

**data\_from\_issue**

<Array> An optional list of values that will be extracted from the issue's body and used to fill the template.

**external\_call**

*Optional* Configuration for a external service call. All available subparams are described in the [external\\_service docs](#).

### 3.3.4 Examples

#### Simplest use case:

Reply with a preconfigured text

```
...
  responders:
    basic_command:
      command: issue complaint
      message: "Please send an email to reports@open.journal"
...
```

#### Multiple instances of the responder, multiple replies, using a template to respond:

```
...
  responders:
    basic_command:
      - code_of_conduct:
          command: code of conduct
          description: Show our community Code of Conduct and Guidelines
          messages:
            - "Our CoC: https://github.com/openjournals/joss/blob/master/CODE\_OF\_CONDUCT.md"
            - "It's adapted from the Contributor Covenant: http://contributor-covenant.org"
            - "Reports of abusive or harassing behavior may be reported to reports@theoj.org"
      - editor_list:
          command: list editors
          description: List all current topic editors
          template_file: editors.md
...
```

### 3.3.5 In action

- Multiple responses:



**submitting\_author** commented

@botsci code of conduct



**botsci** commented

Our CoC: [https://github.com/openjournals/joss/blob/master/CODE\\_OF\\_CONDUCT.md](https://github.com/openjournals/joss/blob/master/CODE_OF_CONDUCT.md)



**botsci** commented

It's adapted from the Contributor Covenant: <http://contributor-covenant.org>



**botsci** commented

Reports of abusive or harassing behavior may be reported to [reports@theoj.org](mailto:reports@theoj.org)

- Replying with a template - The template file:

master ▾

[reviews-repository](#) / [.buffy](#) / [templates](#) / [editors.md](#)

jenny\_calendar Create editors.md

1 contributor

30 lines (22 sloc) | 305 Bytes

**Here is the list of current topic editors:**

**Astronomy**

- astroeditor\_33
- astro\_editor\_2

**Social science**

- sociology\_editor\_1
- sociology\_editor\_2

**Machine Learning**

- Alice
- Bob

Thanks for asking, {{sender}}.

- Replying with a template - In use:



submitting\_author commented

@botsci list editors



botsci commented

**Here is the list of current topic editors:**

**Astronomy**

- astroeditor\_33
- astro\_editor\_2

**Social science**

- sociology\_editor\_1
- sociology\_editor\_2

**Machine Learning**

- Alice
- Bob

Thanks for asking, submitting\_author.

## 3.4 Assign editor

Use this responder to update the value of the editor in the body of the issue. Allows *labeling*.

### 3.4.1 Listens to

```
@botname assign @username as editor
```

### 3.4.2 Requirements

The body of the issue should have the editor placeholder marked with HTML comments.

```
<!--editor--> <!--end-editor-->
```

### 3.4.3 Settings key

assign\_editor

### 3.4.4 Params

#### add\_as\_assignee

*<Boolean>* If true, the editor user will be added as assignee to the issue. Default value is **true**.

#### add\_as\_collaborator

*<Boolean>* If true, the editor user will be added as collaborator to the repo. Default value is **false**.

#### external\_call

*Optional* Configuration for a external service call. All available subparams are described in the [external\\_service docs](#).

### 3.4.5 Examples

Simplest use case:

```
...
  responders:
    assign_editor:
...

```

Restricted to editors:

```
...
  teams:
    editors: 1111111
...
  responders:
    assign_editor:
      only: editors
...

```

Restrict access to editors and add user as assignee and collaborator:

```
...
responders:
  assign_editor:
    only: editors
    add_as_collaborator: true
...
```

### 3.4.6 In action

- Initial state:

## Software submission #3



sample\_user opened this issue 29 days ago



sample\_user commented 29 days ago

Submitting Author: J (@submission\_author)

Repository:

Version submitted:

Editor: <!--editor-->TBD<!--end-editor-->

Editor: TBD

Reviewer 1: TBD

Reviewer 2: TBD

Archive: TBD

Version accepted: TBD

Description

- Invocation:



@botsci assign @arfon as editor




botsci commented on May 20

Assigned! @arfon is now the editor

- Final state:



# Software submission #3

 Open sample\_user opened this issue 29 days ago



sample\_user commented 29 days ago • edited by botsci ▾

Submitting Author: J (@submission\_author)

Repository:

Version submitted:

Editor: @arfon

Reviewer 1: TBD

Reviewer 2: TBD

Archive: TBD

Version accepted: TBD

Description

## 3.5 Remove editor

This responder removes the assigned editor from the body of the issue (the one that can be assigned using the [Assign Editor](#) responder). The user will also be removed from the issue's assignees. Allows *labeling*.

### 3.5.1 Listens to

```
@botname remove editor
```

### 3.5.2 Requirements

In the body of the issue the editor should be enclosed in HTML comments.

```
...
<!--editor--> @sarah_m_g <!--end-editor-->
...
```

### 3.5.3 Settings key

`remove_editor`

### 3.5.4 Params

`no_editor_text`

The text that will go in the editor place to state there's no one assigned. The default value is **Pending**.

### 3.5.5 Examples

Simplest use case:

```
...
  responders:
    remove_editor:
...

```

Action restricted to editors:

```
...
  teams:
    editors: 1111111
...
  responders:
    remove_editor:
      only: editors
...

```

Restrict access to editors, use custom text when there's not editor:

```
...
  responders:
    remove_editor:
      only: editors
      no_editor_text: To be decided
...

```

### 3.5.6 In action

- Initial state:

## Software submission #2

🔔 Open

submitting\_author opened this issue · 53 comments



submitting user opened this issue ·

Submitting Author:

Repository:

Version submitted:

Editor: @journal\_editor

Reviewer 1: TBD

Reviewer 2: TBD

Archive: TBD

Version accepted: v1.0.2

Editor: <!--editor-->@journal\_editor<!--end-editor-->

### Description

#### • Invocation:



arfon commented

@botsci remove editor



botsci commented

Editor removed!

#### • Final state:

## Software submission #2

🔔 Open

submitting\_author opened this issue · 53 comments



submitting user opened this issue · edited by botsci

Submitting Author:

Repository:

Version submitted:

Editor: To be decided

Reviewer 1: TBD

Reviewer 2: TBD

Archive: TBD

Version accepted: v1.0.2

## 3.6 Reviewers list

This responder adds/removes usernames to/from the list of reviewers in the body of the issue. Allows *labeling*.

### 3.6.1 Listens to

```
@botname add <username> as reviewer
```

```
@botname add <username> to reviewers
```

```
@botname remove <username> from reviewers
```

### 3.6.2 Requirements

The body of the issue should have the target field placeholder marked with HTML comments.

```
<!--reviewers-list--> <!--end-reviewers-list-->
```

### 3.6.3 Settings key

reviewers\_list

### 3.6.4 Params

**sample\_value**

<String> An optional sample value string for the target field. It is used for documentation purposes when the *Help responder* lists all available responders. Default value is **@username**.

**no\_reviewers\_text**

The text that will go in the reviewers list place to state there are no reviewers assigned yet. The default value is **Pending**.

**add\_as\_assignee**

<Boolean> Optional. If true, when adding a new reviewer will be added as assignee to the issue. Default value is **false**.

**add\_as\_collaborator**

<Boolean> Optional. If true, when adding a new reviewer will be added as collaborator to the repo. Default value is **false**.

### 3.6.5 Examples

Simplest case:

```
...
  responders:
    reviewers_list:
...

```

With different options:

```
...
  responders:
    reviewers_list:
      only: editors
      sample_value: "@reviewer-login"
      add_as_assignee: true
...

```

### 3.6.6 In action

- Initial state:

Reviewers: @reviewer33

Archive: TRD

- Adding a reviewer:



editor commented

@botsci add @reviewer42 to reviewers



botsci commented

@reviewer42 added to the reviewers list!

- Reviewer added:

Reviewers: @reviewer33, @reviewer42

Archive: TRD

- Removing a reviewer:



**editor** commented

@botsci remove @reviewer42 from reviewers



**botsci** commented

@reviewer42 removed from the reviewers list!

- **Reviewer removed:**

Reviewers: @reviewer33

## 3.7 Invite

This responder creates a repo invitation for a user to be added as collaborator so they have the needed permissions to edit comments. Use this responder to send an invitation to a user to collaborate in the review.

### 3.7.1 Listens to

```
@botname invite @username
```

### 3.7.2 Settings key

invite

### 3.7.3 Examples

**Simplest use case:**

```
...
  responders:
    invite:
...

```

**Action restricted to users in the editors team:**

```
...
  teams:
    editors: 1111111
...
  responders:

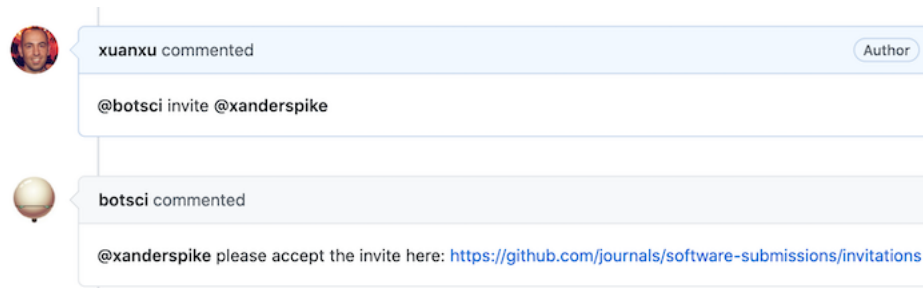
```

(continues on next page)

(continued from previous page)

```
invite:
  only: editors
...
```

### 3.7.4 In action



## 3.8 Set value

This responder can be used to update the value of any field in the body of the issue. Allows *labeling*.

### 3.8.1 Listens to

```
@botname set <value> as <name>
```

For example, if you configure this responder to change the value of the *version*, it would respond to:

```
@botname set v1.0.3 as version
```

### 3.8.2 Requirements

The body of the issue should have the target field placeholder marked with HTML comments.

```
<!--<name>--> <!--end-<name>-->
```

Following the previous example if the name of the field is *version*:

```
<!--version--> <!--end-version-->
```

### 3.8.3 Settings key

set\_value

### 3.8.4 Params

**name**

*Required.* The name of the target field in the body of the issue. It can be set using the **name:** keyword, or via the name of each instance if there are several instances of this responder specified in the settings file.

**if\_missing**

*Optional* Strategy when value placeholders are not defined in the body of the issue. Valid options: *append* (will add the value at the end of the issue body), *prepend* (will add the value at the beginning of the issue body), *error* (will reply a not-found message). If this param is not present nothing will be done if value placeholder is not found.

**aliased\_as**

*Optional.* The name of the value to be used in the command, in case it is different from the target field placeholder marked with HTML comments.

**heading**

if the value placeholder is missing and the *if\_missing* strategy is set to *append* or *prepend*, when adding the value it will include this text as heading instead of just the value name.

**sample\_value**

A sample value string for the target field. It is used for documentation purposes when the [Help responder](#) lists all available responders. Default value is **xxxxx**.

**template\_file**

*Optional* A template file to use to build the response message (name and value are passed to it).

**external\_call**

*Optional* Configuration for a external service call. All available subparams are described in the [external\\_service docs](#).

### 3.8.5 Examples

Simplest use case:

```
...
responders:
  set_value:
    name: version
    sample_value: v1.0.1
...
```

Multiple instances of the responder, some of them restricted to editors:

```
...
responders:
  set_value:
    - version:
        only: editors
        sample_value: "v1.0.0"
```

(continues on next page)



(continued from previous page)


```

- archive:
  only: editors
  sample_value: "10.21105/joss.12345"
  if_missing: prepend
  heading: "Archive DOI"
- repository:
  sample_value: "github.com/openjournals/buffy"
...

```

### 3.8.6 In action

- Initial state:

 Open submitting\_author opened this issue · 53 comments



submitting user opened this issue

Submitting Author:  
Repository:  
Version submitted:

Editor: To be decided

Reviewer 1: TBD

Reviewer 2: TBD

Archive: TBD

Version accepted: TBD

Version accepted: <!--version--> TBD <!--end-version-->

- Invocation:



xuanxu commented


@botsci set v1.0.2 as version



botsci commented

Done! version is now v1.0.2

- Final state:

 submitting\_author opened this issue · 53 comments



submitting user opened this issue · edited by botsci

Submitting Author:  
Repository:  
Version submitted:  
  
Editor: To be decided  
Reviewer 1: TBD  
Reviewer 2: TBD  
Archive: TBD  
Version accepted: v1.0.2

## 3.9 List of values

This responder adds values to/removes values from a list in the body of the issue. Allows *labeling*.

### 3.9.1 Listens to

```
@botname add <value> to <list-name>
```

```
@botname remove <value> from <list-name>
```

For example, if you configure this responder to add/remove values for the *authors* list, it would respond to:

```
@botname add @username to authors
```

### 3.9.2 Requirements

The body of the issue should have the target field placeholder marked with HTML comments.

```
<!--<listname>-list--> <!--end-<listname>-list-->
```

Following the previous example if the name of the field is *authors*:

```
<!--authors-list--> <!--end-authors-list-->
```

### 3.9.3 Settings key

list\_of\_values

### 3.9.4 Params

**name**

*Required.* The name of the list. It can be set using the `name:` keyword, or via the name of each instance if there are several instances of this responder specified in the settings file.

**sample\_value**

An optional sample value string for the target field. It is used for documentation purposes when the *Help responder* lists all available responders. Default value is **xxxxx**.

**add\_as\_assignee**

*<Boolean>* Optional. If true and the value is a user name, it will be added as assignee to the issue. Default value is **false**.

**add\_as\_collaborator**

*<Boolean>* Optional. If true and the value is a user name, it will be added as collaborator to the repo. Default value is **false**.

### 3.9.5 Examples

**Simple case: A single list**

```
...
responders:
  list_of_values:
    name: authors
...
```

**Several lists with different options:**

```
...
responders:
  list_of_values:
    - versions:
      sample_value: "v1.0.2"
    - authors
      only: editors
      sample_value: "@username"
      add_as_collaborator: true
...
```

### 3.9.6 In action

- Initial state:

```
Authors: John, Alice
```

- Adding to the list:



**editor** commented

@botsci add Kate to authors



**botsci** commented

Kate added to the authors list!

- **Intermediate state:**

Authors: John, Alice, Kate

- **Removing from the list:**



**editor** commented

@botsci remove John from authors



**botsci** commented

John removed from the authors list!

- **Final state:**

Authors: Alice, Kate

## 3.10 List team members

This responder replies with a list of members from a GitHub team

### 3.10.1 Listens to

```
@botname <command>
```

For example, if you configure the command to be *list editors*, it would respond to:

```
@botname list editors
```

### 3.10.2 Settings key

list\_team\_members

### 3.10.3 Params

**command**

The command this responder will listen to.

**team\_id**

The id of the GitHub team to be listed.

**heading**

*Optional* Heading for the replied list.

**description**

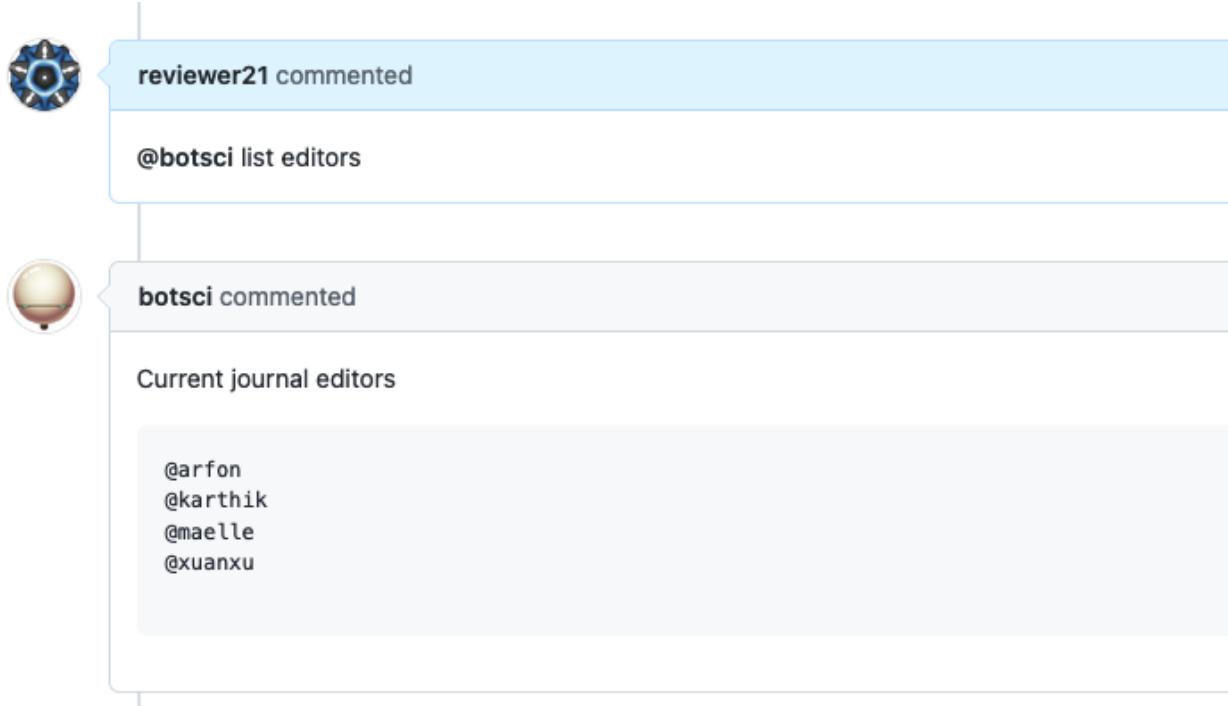
*Optional* String to show when the help command is invoked.

### 3.10.4 Examples

List editors team members with custom heading

```
...
responders:
  list_team_members:
    command: list editors
    team_id: 3824115
    heading: Current journal editors
...
```

### 3.10.5 In action



## 3.11 Add/Remove assignee

This responder adds and removes users to the assignees list of the issue. Allows *labeling*.

### 3.11.1 Listens to

```
@botname add assignee: @username
```

```
@botname remove assignee: @username
```

### 3.11.2 Requirements

Only users that are collaborators in the target issue can be added as assignees. Otherwise the responder will reply with a *not enough permissions* message.

### 3.11.3 Settings key

add\_remove\_assignee

### 3.11.4 Examples

Simplest use case:

```
...
  responders:
    add_remove_assignee:
...

```

Hidden from commands list and restricted to editors:

```
...
  responders:
    add_remove_assignee:
      only: editors
      hidden: true
...

```

### 3.11.5 In action



journal\_editor commented

@botsci add assignee: @arfon



botsci commented

@arfon added as assignee.

## 3.12 Reviewer checklist comment

This responder adds a reviewer checklist editing the comment triggering the responder if the author of the comment is a reviewer. This way of adding checklists (instead of adding them to the body of the issue) does not require the reviewers to be collaborator of the repository, as they will be able to edit their own comments to update the progress of the checklist.

### 3.12.1 Listens to

```
@botname generate my checklist
```

### 3.12.2 Requirements

The checklist is read from a template file that should be available in the repository.

### 3.12.3 Settings key

reviewer\_checklist\_comment

### 3.12.4 Params

**template\_file**

*Required.* The name of the template file to edit the comment with.

**data\_from\_issue**

*<Array>* An optional list of values that will be extracted from the issue's body and used to fill the template.

**command**

Optional. The command triggering this responder. Default is *generate my checklist*

### 3.12.5 Examples

Simplest use case:

```
...
responders:
  add_remove_checklist:
    template_file: reviewer_checklist.md
...
```

Using info from the body to fill in the template. Custom command:

```
...
responders:
  add_remove_checklist:
    command: create reviewer checklist
    template_file: reviewer_checklist.md
    data_from_issue:
      - target-repository
      - author-handle
...
```



### 3.12.6 In action

- The template:

master
.buffy / templates / reviewer\_checklist.md

29 lines (20 sloc) | 2.01 KB

```

## Review checklist for @{{sender}}

### Conflict of interest

- [ ] I confirm that I have read the [JOSS conflict of interest (COI) policy](https://github.com/openjournals/joss/blob/master/COI.md) and that: I have no COIs with reviewing this work or that any perceived COIs have been waived by JOSS for the purpose of this review.

### Code of Conduct

- [ ] I confirm that I read and will adhere to the [JOSS code of conduct](https://joss.theoj.org/about#code_of_conduct).

### General checks

- [ ] **Repository:** Is the source code for this software available at the [{{target-repository}}]({{target-repository}})?
- [ ] **License:** Does the repository contain a plain-text LICENSE file with the contents of an [OSI approved](https://opensource.org/licenses/alphabetical) software license?
- [ ] **Contribution and authorship:** Has the submitting author ({{author-handle}}) made major contributions to the software? Does the full list of paper authors seem appropriate and complete?
- [ ] **Substantial scholarly effort:** Does this submission meet the scope eligibility described in the [JOSS guidelines](https://joss.readthedocs.io/en/latest/submitting.html#substantial-scholarly-effort)?

### Functionality

- [ ] **Installation:** Does installation proceed as outlined in the documentation?
- [ ] **Functionality:** Have the functional claims of the software been confirmed?
- [ ] **Performance:** If there are any performance claims of the software, have they been confirmed? (If there are no claims, please check off this item.)

### Documentation

- [ ] **A statement of need:** Do the authors clearly state what problems the software is designed to solve and who the target audience is?
- [ ] **Installation instructions:** Is there a clearly-stated list of dependencies? Ideally these should be handled with an automated package management solution.
- [ ] **Example usage:** Do the authors include examples of how to use the software (ideally to solve real-world analysis problems).
- [ ] **Functionality documentation:** Is the core functionality of the software documented to a satisfactory level (e.g., API method documentation)?
- [ ] **Automated tests:** Are there automated tests or manual steps described so that the functionality of the software can be verified?
- [ ] **Community guidelines:** Are there clear guidelines for third parties wishing to 1) Contribute to the software 2) Report issues or problems with the software 3) Seek support

### Software paper

```

- Invocation:



reviewer33 commented

@botsci generate my checklist

- Comment edited by the bot:



reviewer33 commented • edited by botsci ▾



## Review checklist for @reviewer33

### Conflict of interest

- ☐ I confirm that I have read the [JOSS conflict of interest \(COI\) policy](#) and that: I have no COIs with reviewing this work or that any perceived COIs have been waived by JOSS for the purpose of this review.

### Code of Conduct

- ☐ I confirm that I read and will adhere to the [JOSS code of conduct](#).

### General checks

- ☐ **Repository:** Is the source code for this software available at the <https://github.com/andr1976/HydDown>?
- ☐ **License:** Does the repository contain a plain-text LICENSE file with the contents of an [OSI approved](#) software license?
- ☐ **Contribution and authorship:** Has the submitting author ([@package-author](#)) made major contributions to the software? Does the full list of paper authors seem appropriate and complete?
- ☐ **Substantial scholarly effort:** Does this submission meet the scope eligibility described in the [JOSS guidelines](#)

### Functionality

- ☐ **Installation:** Does installation proceed as outlined in the documentation?
- ☐ **Functionality:** Have the functional claims of the software been confirmed?
- ☐ **Performance:** If there are any performance claims of the software, have they been confirmed? (If there are no claims, please check off this item.)

## 3.13 Add/Remove checklist

This responder adds and removes checklists for reviewers at the end of the body of the issue. Allows *labeling*.

### 3.13.1 Listens to

```
@botname add checklist for @username
```

```
@botname remove checklist for @username
```

### 3.13.2 Requirements

The checklist is read from a template file that should be available in the repository.

### 3.13.3 Settings key

add\_remove\_checklist

### 3.13.4 Params

**template\_file**

*Required.* The name of the template file to append to the body.

**data\_from\_issue**

<Array> An optional list of values that will be extracted from the issue's body and used to fill the template.

### 3.13.5 Examples

Simplest use case:

```
...
responders:
  add_remove_checklist:
    template_file: reviewer_checklist.md
...
```

Using info from the body to fill in the template. Action restricted to editors:

```
...
responders:
  add_remove_checklist:
    only: editors
    template_file: reviewer_checklist.md
    data_from_issue:
      - target-repository
      - author-handle
...
```

### 3.13.6 In action

- The template:

master
.bufy / templates / reviewer\_checklist.md

29 lines (20 sloc) | 2.01 KB

```

### Conflict of interest

- [ ] I confirm that I have no COIs with reviewing this work or that any perceived COIs have been waived by an editor for the purpose of this review.

### Code of Conduct

- [ ] I confirm that I read and will adhere to the code of conduct.

### General checks

- [ ] **Repository:** Is the source code for this software available at the [repo url]({{target-repository}})?
- [ ] **License:** Does the repository contain a plain-text LICENSE file with the contents of an [OSI approved](https://opensource.org/licenses/alphabetical) software license?
- [ ] **Contribution and authorship:** Has the submitting author ({{author-handle}}) made major contributions to the software?
- [ ] **Substantial scholarly effort:** Does this submission meet the scope eligibility?

### Functionality

- [ ] **Installation:** Does installation proceed as outlined in the documentation?
- [ ] **Functionality:** Have the functional claims of the software been confirmed?
- [ ] **Performance:** If there are any performance claims of the software, have they been confirmed? (If there are no claims, please check off this item.)

### Documentation

- [ ] **A statement of need:** Do the authors clearly state what problems the software is designed to solve and who the target audience is?
- [ ] **Installation instructions:** Is there a clearly-stated list of dependencies? Ideally these should be handled with an automated package management solution.
- [ ] **Example usage:** Do the authors include examples of how to use the software (ideally to solve real-world analysis problems).
- [ ] **Functionality documentation:** Is the core functionality of the software documented to a satisfactory level (e.g., API method documentation)?
- [ ] **Automated tests:** Are there automated tests or manual steps described so that the functionality of the software can be verified?
- [ ] **Community guidelines:** Are there clear guidelines for third parties wishing to 1) Contribute to the software 2) Report issues or problems with the software 3) Seek support

```

- Initial state:

## Software review #7

 Open submitter opened this issue yesterday



submitter commented yesterday • edited ▾

Member 😊 ...

Submitting Author: JJ (@submitter)  
 Repository: <https://github.com/matrix-q/generator>  
 Version submitted: 1.3  
 Editor: TBD  
 Reviewer 1: TBD  
 Reviewer 2: TBD  
 Archive: TBD  
 Version accepted: TBD

### Description

This software is a Q-Matrices generator.

Based on explicit values for solar abundances,  $z$  and IMF, it calculates matrices  $Q(i,j)$  of masses of elements  $i$  ejected to the galactic medium as element  $j$ , for a complete range of stellar masses, accounting for supernovas of types Ia and II.  
 You can read more about the Matrices  $Q$  formalism in Ferrini et al. 1992.

Intergalactic computes the contribution matrix of 15 elements:

H D He3 He4 C C13 N O n.r. Ne Mg Si S Ca Fe

#### • Invocation:



editor commented

Member 😊 ...

@botsci add checklist for @awesomereviewer



botsci commented

Member 😊 ...

Checklist added for @awesomereviewer

#### • Final state:

## Software review #7



submitter opened this issue yesterday



submitter commented yesterday • edited ▾

Member



Submitting Author: JJ (@submitter)

Repository: <https://github.com/matrix-q/generator>

Version submitted: 1.3

Editor: TBD

Reviewer 1: TBD

Reviewer 2: TBD

Archive: TBD

Version accepted: TBD

## Description

This software is a Q-Matrices generator.

Based on explicit values for solar abundances,  $z$  and IMF, it calculates matrices  $Q(i,j)$  of masses of elements  $i$  ejected to the galactic medium as element  $j$ , for a complete range of stellar masses, accounting for supernovas of types Ia and II.  
You can read more about the Matrices  $Q$  formalism in Ferrini et al. 1992.

Intergalactic computes the contribution matrix of 15 elements:

H D He3 He4 C C13 N O n.r. Ne Mg Si S Ca Fe

## Review checklist for @awesomereviewer

## Conflict of interest

- ☐ I confirm that I have no COIs with reviewing this work or that any perceived COIs have been waived by an editor for the purpose of this review.

## Code of Conduct

- ☐ I confirm that I read and will adhere to the code of conduct.

## General checks

- ☐ **Repository:** Is the source code for this software available at the [repo url](#)?
- ☐ **License:** Does the repository contain a plain-text LICENSE file with the contents of an [OSI approved](#) software license?
- ☐ **Contribution and authorship:** Has the submitting author (@submitter) made major contributions to the software?
- ☐ **Substantial scholarly effort:** Does this submission meet the scope eligibility?

## Functionality

- ☐ **Installation:** Does installation proceed as outlined in the documentation?
- ☐ **Functionality:** Have the functional claims of the software been confirmed?
- ☐ **Performance:** If there are any performance claims of the software, have they been confirmed? (If there are no claims, please check off this item.)

## Documentation

- ☐ **A statement of need:** Do the authors clearly state what problems the software is designed to solve and who the target audience is?
- ☐ **Installation instructions:** Is there a clearly-stated list of dependencies? Ideally these should be handled with an automated package management solution.
- ☐ **Example usage:** Do the authors include examples of how to use the software (ideally to solve real-world analysis problems).
- ☐ **Functionality documentation:** Is the core functionality of the software documented to a satisfactory level (e.g., API method documentation)?
- ☐ **Automated tests:** Are there automated tests or manual steps described so that the functionality of the software can be verified?
- ☐ **Community guidelines:** Are there clear guidelines for third parties wishing to 1) Contribute to the software 2) Report issues or problems with the software 3) Seek support

## 3.14 Label command

This responder defines a custom command to add and/or remove labels to the issue when invoked.

### 3.14.1 Listens to

```
@botname <command>
```

For example, if you configure the command to be *review successful*, it would respond to:

```
@botname review successful
```

### 3.14.2 Settings key

label\_command

### 3.14.3 Params

#### command

The command this responder will listen to.

#### add\_labels

<Array> A list of text labels to add to the issue.

#### remove\_labels

<Array> A list of text labels to remove from the labels of the issue.

### 3.14.4 Examples

#### Simplest use case:

Just add a label.

```
...
responders:
  label_command:
    command: review successful
    add_labels:
      - recommend publication
...
```

#### Multiple instances of the responder, restricted to editors, adding and removing labels:

```
...
responders:
  label_command:
    - review_ok:
        only: editors
        command: review successful
        add_labels:
```

(continues on next page)

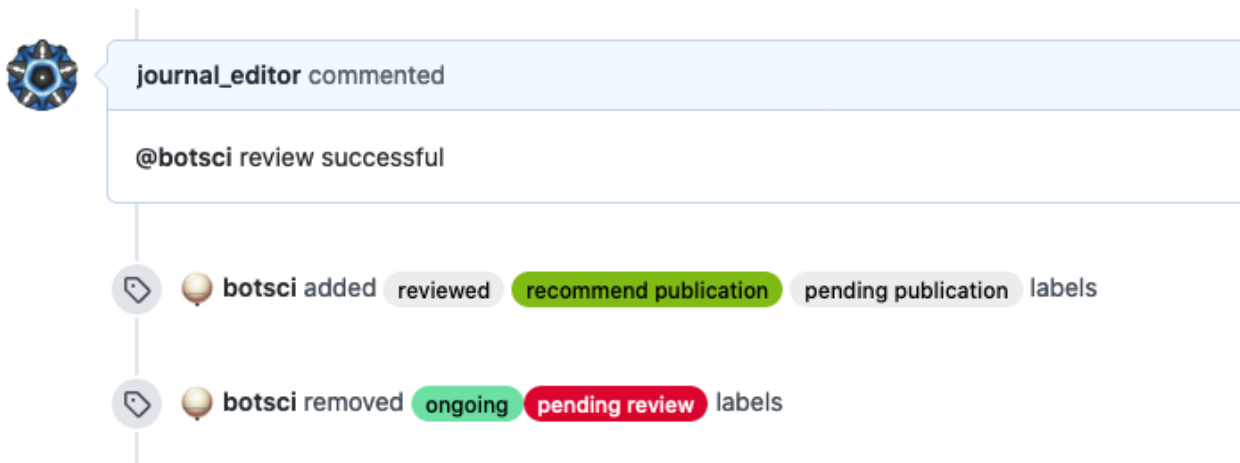
(continued from previous page)

```

- reviewed
- recommend publication
- pending publication
remove_labels:
- ongoing
- pending review
- review_nok:
  only: editors
  command: review failed
  add_labels:
    - recommend rejection
...

```

### 3.14.5 In action



## 3.15 Check references

This responder checks (asynchronously) the validity of the DOIs from a list of BibTeX entries (a paper's references file).

### 3.15.1 Listens to

```
@botname check references
```

A non-default branch can be specified to look for the paper's files in it:

```
@botname check references from branch <custom-branch-name>
```



### 3.15.2 Requirements

The target repository should include a paper.md or paper.tex file and its corresponding references file (paper.bib or paper.yml) with the BibTeX entries.

The body of the issue should have the url of the repository with the paper's files marked with HTML comments.

```
<!--target-repository--> URL HERE <!--end-target-repository-->
```

### 3.15.3 Settings key

check\_references

### 3.15.4 Params

#### url\_field

The optional name of the field marked with HTML comments where the URL of the repository with the paper is located. By default if this setting is not present, the value will be **target-repository**. Meaning Buffy will look for a string in the body of the issue between **<!--target-repository-->** and **<!--end-target-repository-->** HTML comments.

#### branch\_field

The optional name of the field marked with HTML comments where the name of the branch is located. Defaults to **branch** (so Buffy will look for **<!--branch-->** and **<!--end-branch-->** in the body of the issue). If the setting is not present or the branch field is not found in the body of the issue, the default branch of the git repo will be used.

### 3.15.5 Examples

Simplest case:

```
...
check_references:
...
```

Buffy will clone the git repository specified between **<!--target-repository-->** and **<!--end-target-repository-->** marks and check the DOIs for all entries in the paper.bib file.

Example customizing fields:

```
...
check_references:
  url_field: software-location
  branch_field: branch-to-review
...
```

Buffy will clone the git repository specified between **<!--software-location-->** and **<!--end-software-location-->** marks, then checkout into the branch specified between **<!--branch-to-review-->** and **<!--end-branch-to-review-->** and then check the DOIs for all entries in the paper.bib file.

### 3.15.6 In action

- Issue body with the repository's URL:

## Checking references #8



Open

Journal opened this issue 2 days ago · 16 comments



Write

Preview

Submitting Author: GG

Repository: <https://github.com/alan-turing-institute/MLJ.jl>

Version submitted: 2.1

```
<!--target-repository-->https://github.com/alan-turing-institute/MLJ.jl<!--end-target-repository-->
```

- In use:



editor commented

@botsci check references



botsci commented

Reference check summary (note 'MISSING' DOIs are suggestions that need verification):

OK DOIs

- 10.1109/ANZIIS.1994.396988 is OK
- 10.1137/141000671 is OK
- 10.21105/joss.00602 is OK
- 10.18637/jss.v028.i05 is OK
- 10.5281/zenodo.3730565 is OK
- 10.15200/winn.153459.98975 is OK
- 10.5334/jors.151 is OK

MISSING DOIs

- None

INVALID DOIs

- None

- With non-default branch:



editor commented

@botsci check references from branch adding-latin-docs



botsci commented

Reference check summary (note 'MISSING' DOIs are suggestions that need verification):

OK DOIs

- 10.1109/ANZIIS.1994.396988 is OK
- 10.1137/141000671 is OK
- 10.21105/joss.00602 is OK
- 10.18637/jss.v028.i05 is OK
- 10.5281/zenodo.3730565 is OK
- 10.15200/winn.153459.98975 is OK
- 10.5334/jors.151 is OK

MISSING DOIs

- None

INVALID DOIs

- None

## 3.16 Repository checks

This responder performs (asynchronously) several checks on the target repository.

### 3.16.1 Listens to

```
@botname check repository
```

A non-default branch can be specified to run the checks on it:

```
@botname check repository from branch <custom-branch-name>
```

### 3.16.2 Requirements

The body of the issue should have the url of the repository marked with HTML comments.

```
<!--target-repository--> URL HERE <!--end-target-repository-->
```

### 3.16.3 Settings key

repo\_checks

### 3.16.4 Params

#### checks

An optional list (Array) of checks to perform. If non present or empty all available checks will be run (see [available checks](#) for the values to use in the config file).

#### url\_field

The optional name of the field marked with HTML comments where the URL of the repository with the paper is located. By default if this setting is not present, the value will be **target-repository**. Meaning Buffy will look for a string in the body of the issue between **<!--target-repository-->** and **<!--end-target-repository-->** HTML comments.

#### branch\_field

The optional name of the field marked with HTML comments where the name of the branch is located. Defaults to **branch** (so Buffy will look for **<!--branch-->** and **<!--end-branch-->** in the body of the issue). If the setting is not present or the branch field is not found in the body of the issue, the default branch of the git repo will be used.

### 3.16.5 Available checks

The following values are valid for the `:checks` list:

- **repo summary**: This check performs an analysis of the source code and list authorship, contributions and file types information.
- **languages**: This will detect the languages used in the repository and tagged the issue with the top three used languages.
- **wordcount**: This will count the number of words in the paper file.
- **license**: This will look for an Open Source License in the target repo and reply an error message if no license is found.
- **statement of need**: This check will look for an *Statement of need* section in the paper content.

### 3.16.6 Examples

**Simplest case:**

```
...  
  repo_checks:  
...
```

Buffy will clone the git repository specified between `<!--target-repository-->` and `<!--end-target-repository-->` HTML comments and run all available checks.

**Run selected checks:**

```
...  
  repo_checks:  
    checks:  
      - repo summary  
      - languages  
...
```

Buffy will only run the `repo summary` and the `languages` checks.

## 3.16.7 In action



journal\_reviewer commented

@botsci check repository



botsci commented

Software report:

github.com/AlDanial/cloc v 1.88 T=0.40 s (1114.3 files/s, 49181.4 lines/s)

Language	files	blank	comment	code
Ruby	117	1063	431	4752
JSON	17	14	0	2165
ERB	46	275	1	1675
Markdown	20	720	0	1582
YAML	14	49	101	226
LESS	1	3	2	219
CSS	2	4	4	218
HTML	4	15	3	183
Python	1	43	90	45
SUM:	232	2056	632	10770

Statistical information for the repository '674cb9cbc5609aa1a154J64':

Author	Commits	Insertions	Deletions	% of changes
Arfon Smith	810	21729	4154	68.78
Daniel S. Katz	39	110	53	0.43
Elizabeth DuPre	12	329	80	1.09
Jed Brown	7	51	27	0.21
Juanjo Bazán	57	1975	1715	9.81
Karthik Ram	6	21	18	0.10
Kyle Niemeyer	17	158	12	0.45
Lorena A. Barba	12	127	45	0.46
dependabot[bot]	8	13	13	0.07



botsci added Ruby JavaScript CoffeeScript labels 11 minutes ago

## 3.17 Thanks

This responder replies when a user thanks the bot.

### 3.17.1 Listens to

Thanks @botname

Thank you @botname

@botname thanks

@botname thank you

### 3.17.2 Settings key

thanks

### 3.17.3 Params

**reply**

The message the bot will send back. Default value is “**You are welcome**”.

### 3.17.4 Examples

Simplest use case:

```
...  
  responders:  
    thanks:  
...  

```

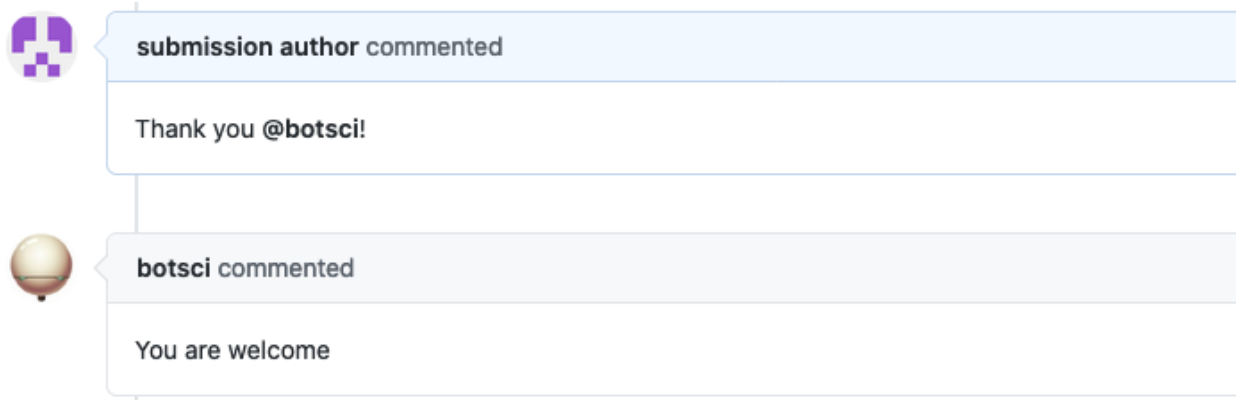
Custom message and hidden from public commands list:

```
...  
  responders:  
    thanks:  
      reply: "No problem, I'm here to help!"  
      hidden: true  
...  

```



### 3.17.5 In action



## 3.18 Reminders

This responder allows to schedule a reminder for the author or a reviewer to return to a review after a certain period of time (supported units: days and weeks). The command will only work if the mentioned user is an author, a reviewer for the submission or the sender of the message (so editors can set reminders for themselves).

### 3.18.1 Listens to

```
@botname remind @username in 2 weeks
```

```
@botname remind @reviewer in 10 days
```

### 3.18.2 Settings key

reminders

### 3.18.3 Params

#### reviewers

*Optional.* The HTML-comment value name in the body of the issue to look for reviewers. Default value is **reviewers-list**.

#### authors

*Optional.* The HTML-comment value name in the body of the issue to look for authors. Default value is **author-handle**.

### 3.18.4 Examples

Simple use case:

```
...
responders:
  reminders:
    only: editors
...
```

Custom html fields:

```
...
responders:
  reminders:
    only: editors
    authors:
      - author1
      - author2
...
```

Now it will allow to set a reminder for all the users listed in the body of the issue in `reviewers-list`, `author1` and `author2` HTML fields.

### 3.18.5 In action

- Scheduling a reminder:



editor commented 15 days ago

@botsci remind @reviewer33 in 2 weeks



botsci commented 15 days ago

Reminder set for @reviewer33 in 2 weeks

- The reminder:



botsci commented now

👋@reviewer33, please update us on how your review is going (this is an automated reminder).

## 3.19 Initial values

This responder acts when a new issue is opened. It checks for the presence of placeholders in the body of the issue for all the configured values.

### 3.19.1 Listens to

New issue opened event.

### 3.19.2 Settings key

`initial_values`

### 3.19.3 Params

The **values** parameter is mandatory.

#### **values**

An array of values. Optionally each value can be individually customized.

For each value listed under **values** this options can be provided:

#### **heading**

*Optional.* If the value placeholder is missing when adding the value it will include this text as heading instead of just the value name. Default value is the value name capitalized bold.

#### **value**

*Optional* Value to add inside the HTML comments. Default value is empty string.

#### **action**

*Optional* Strategy when value placeholders are not defined in the body of the issue. Valid options: *append* (will add the value at the end of the issue body), *prepend* (will add the value at the beginning of the issue body). Default value is *prepend*

#### **warn\_if\_empty**

*Optional* If set to *true* if the placeholder for this value is not present or present but empty a new comment will be replied to the issue warning of the missing value. Default is *false*.

### 3.19.4 Examples

#### **Simplest use case:**

Verify presence of `<!--version--><!--end-version-->` and `<!--target-repository--><!--end-target-repository-->` in the body of the issue:

```
...
  responders:
    initial_values:
      values:
        - version
        - target-repository
...
```

#### **Multiple values with custom properties:**

```

...
responders:
  initial_values:
    values:
      - version:
        - value: "vX.X.X"
        - action: append
      - author1:
        - heading: "Author Github handle:"
        - warn_if_empty: true
      - target-repository
      - archive
      - package-name:
        - warn_if_empty: true
...

```

### 3.19.5 In action

Using the *Multiple values with custom properties* example config:

- Initial state:

## Software submission #16

 Open author opened this issue



author commented

Package name:  
Submitting Author: J.J.  
Repository: <https://github.com/openjournals/science.py>

## This is the review issue

Editor: TBD  
Reviewers: No reviewers yet  
Archive: TBD

Actual text in the initial body of the issue:

```

**Package name:** <!--package-name--><!--end-package-name-->
**Submitting Author:** <!--author-name-->J.J.<!--end-author-name-->
**Repository:** <!--target-repository-->https://github.com/openjournals/science.py<!--end-target-repos

# This is the review issue

**Editor:** <!--editor-->TBD<!--end-editor-->
**Reviewers:** <!--reviewers-list-->No reviewers yet<!--end-reviewers-list-->
**Archive:** <!--archive-->TBD<!--end-archive-->

```

- Final state:

## Software submission #16



author opened this issue



author commented • edited ▾

Author Github handle

**Package name:**

**Submitting Author:** J.J.

**Repository:** <https://github.com/openjournals/science.py>

### This is the review issue

**Editor:** TBD

**Reviewers:** No reviewers yet

**Archive:** TBD

**Version:** vX.X.X



editorial-bot commented

Missing values: author1, package-name

Actual text in the final body of the issue:



```

Author Github handle <!--author1--><!--end-author1-->
**Package name:** <!--package-name--><!--end-package-name-->
**Submitting Author:** <!--author-name-->J.J.<!--end-author-name-->
**Repository:** <!--target-repository-->https://github.com/openjournals/science.py<!--end-target-repository-->

# This is the review issue

**Editor:** <!--editor-->TBD<!--end-editor-->
**Reviewers:** <!--reviewers-list-->No reviewers yet<!--end-reviewers-list-->
**Archive:** <!--archive-->TBD<!--end-archive-->
**Version:** <!--version-->vX.X.X<!--end-version-->

```

**Result:**

- *version* has been appended at the end of the body with a value of *vX.X.X*
- *author1* has been added with the custom heading
- *target-repository* was already present, so nothing has been done with it
- *archive* was already present, so nothing has been done with it
- *package-name* was not present so it has been prepended.
- New comment was created with a warning of missing values for *package-name* and *author1*

## 3.20 Welcome

This responder acts when a new issue is opened. It can reply with text messages, using a template, create a background job to asynchronously call an external service's API and/or triggering another responder.

Allows *labeling*.

### 3.20.1 Listens to

New issue opened event.

### 3.20.2 Settings key

welcome

### 3.20.3 Requirements

#### When using a template to respond:

When rendering a template a map of values will be passed to it:

- **issue\_id**: The id of the issue
- **repo**: the name of the repository
- **sender**: the handle of the user creating the issue
- **bot\_name**: the name of the bot user responding

If the template needs some other value included in the body of the issue, they can be declared using the `data_from_issue` param and those values will be passed to the template too, if can be extracted from the body. They can be used in the template using the syntax:

```
{{variable_name}}
```

In order to use a template, Buffy will look for the file declared in the `template_file` param in the target repo, in the location specified with the `template_path` setting (by default `.buffy/templates`). In short: the *template\_file* should be located in the *template\_path*.

The values needed by the template that are listed in the `data_from_issue` param must be extractable: they have to be enclosed in HTML comments:

```
<!--<name>--> Info to extract <!--end-<name>-->
```

So, for example, if you want to use the value of *version* in the template, the body of the issue must include it inside HTML comments:

```
<!--version--> v2.1 <!--end-version-->
```

Then it should be declared in the settings file, listed in the `data_from_issue` param:

```
responders:
  welcome:
    template_file: welcome.md
    data_from_issue:
      - version
```

And can then be used in the template:

```
Thank you for your submission, we will review the {{version}} release of your software.
```

#### When invoking an external service:

Some parameters are required for the external call to work: the `name` of the service and the `url` of the call, both configured in the settings YAML file nested under the `external_service` param.

Similarly to the [External Service responder](#) if the call is successful the response is posted as a comment in the issue (optionally using a template).

You can configure a template file as a response after the external API call, this template is configured separately from the previous general response template. The response from the external service should be in JSON format. It will be parsed and the resulting hash values will be passed to the template.

### 3.20.4 Params

For replying with plain text message(s):

**message**

A text message to use as reply.

**messages**

*<Array>* A list of text messages to respond with.

To reply with a template file:

**template\_file**

The name of the template file to use to build the response message.

**data\_from\_issue**

*<Array>* An optional list of values that will be extracted from the issue's body and used to fill the template.

Calling an external service:

**external\_service**

All the configuration for the service is nested under this param. Possible options are:

**name**

*Required.* The name for this service.

**url**

*Required.* The url to call.

**method**

The HTTP method to use. Valid values: [get, post]. Default is **post**.

**template\_file**

The optional template file to use to build the response message after the external call.

**headers**

*<Array>* An optional list of *key: value* pairs to be passed as headers in the external service request.

**data\_from\_issue**

*<Array>* An optional list of values that will be extracted from the issue's body and used to fill the template.

**query\_params**

*<Array>* An optional list of params to add to the query of the external call. Common place to add API\_KEYS or other authentication info.

**mapping**

*<Array>* An optional mapping of variable names in the query of the external service call.

Running other responder(s):

**run\_responder**

Allows to call a different responder. Subparams are:

**responder\_key**

*Required.* The key to find the responder in the config file.

**responder\_name**

*Optional.* The name of the responder in the config file if there are several instances under the same responder key.



**message**

*Optional.* The message to trigger the responder with.

If you want to run multiple responders, use an array of these subparams.

General:

**close**

*<Boolean>* Optional parameter, if **true** the responder will close the issue. Default is **false**.

**check\_references**

Optional. If present the validity of the DOIs from the paper's references file will be checked.

**repo\_checks**

Optional. If present the responder will perform (asynchronously) several checks on the target repository. You can configure which checks to perform using nested params. Available options are those of the [repository\\_checks](#) responder

**hidden**

Is **true** by default.

### 3.20.5 Examples

Simplest use case:

```
...
responders:
  welcome:
    message: "Thanks for your submission!"
...
```

Multiple messages and a template:

```
...
responders:
  welcome:
    messages:
      - "You can list all the available commands typing `@botsci help`"
      - "The review will start once two reviewers are assigned, please stay tuned."
    template_file: welcome.md
    data_from_issue:
      - repository
      - version
...
```

Calling an external service:

```
...
responders:
  welcome:
    external_service:
      url: https://dummy-external-service.herokuapp.com/code-analysis
      method: post
      query_params:
        secret: A1234567890Z
      data_from_issue:
```

(continues on next page)

(continued from previous page)

```
- target-repo
mapping:
  id: issue_id
...
```

When a new issue is created the responder will send a POST request to `https://dummy-external-service.herokuapp.com/code-analysis` with a JSON body:

```
{
  "secret": "A1234567890Z", # declared in the query_params setting
  "target-repo": "...",     # the value is extracted from the body of the issue
  "id": "...",              # the value corresponds to issue_id, it has been mapped to id
  "repo": "...",           # the origin repo where the invocation happened
  "sender": "...",         # the user invoking the command
  "bot_name": "...",       # the bot user name that will be responding
}
```

And the response from the external service will be posted as a comment in the original issue.

### 3.20.6 In action

#### Text messages and template file:

- The template file:

Branch: master ▾

[sunnydale](#) / [.buffy](#) / [templates](#) / [welcome.md](#)

3 lines (2 sloc) | 118 Bytes

Welcome {{sender}}, thanks for opening this issue 🎉

Thanks for sending your software {{repository}} to {{repo}}

- In use (template + 2 messages):

## Software submission #5



karthik opened this issue · 1 comment



karthik commented

Submitting Author: karthik (@karthik)

Repository: ropensci/testing

Version submitted: VERSION

Repository: <!--repository-->ropensci/testing<!--end-repository-->

### Description

- A testing package to test code

### Scope

- Please indicate which category or categories this package falls under:
  - ☒ data retrieval
  - ☒ scientific software wrappers
  - ☐ field and lab reproducibility tools
  - ☐ geospatial data
  - ☐ text analysis



botsci commented

Welcome karthik, thanks for opening this issue 🎉

Thanks for sending your software ropensci/testing to aliadalabs/sunnydale



botsci commented

You can list all the available commands typing `@botsci help`



botsci commented

The review will start once two reviewers are assigned, please stay tuned.

Calling an external service:

## Py-Astro #12



package\_author opened this issue 1 hour ago



package\_author commented 1 hour ago

Submitting Author: JJ (@package\_author)  
Repository: [github.com/scientific-software/pyastro](https://github.com/scientific-software/pyastro)  
Version submitted: 5.0

Editor: TBD  
Reviewer 1: TBD  
Reviewer 2: TBD  
Archive: TBD  
Version accepted: TBD

### Description

Lorem ipsum fugit nulla sint ea aspernatur. Repellat dolor consequatur optio nihil o  
Quia aut maxime molestiae numquam sit nobis reprehenderit exercitationem.  
Sed animi modi quia distinctio. Voluptatem laborum ipsum et earum deleniti vel harum  
Voluptate delectus nihil culpa laborum in porro. Velit ut dolor atque dolorem ration  
Ut odit inventore ut consequatur id. Nam nihil reprehenderit ratione dolor autem.



botsci commented 1 hour ago

The quality of the code in [github.com/scientific-software/pyastro](https://github.com/scientific-software/pyastro) is not bad

## 3.21 Goodbye

This responder acts when an issue is closed. It can reply with text messages, *using a template* or creating a background job to asynchronously call an external service's API.

Allows *labeling*.

### 3.21.1 Listens to

Issue closed event.

### 3.21.2 Settings key

goodbye

### 3.21.3 Requirements

#### When invoking an external service:

Some parameters are required for the external call to work: the **name** of the service and the **url** of the call, both configured in the settings YAML file nested under the **external\_service** param.

Similarly to the [External Service responder](#) if the call is successful the response is posted as a comment in the issue (optionally using a template).

You can configure a template file as a response after the external API call, this template is configured separately from the previous general response template. The response from the external service should be in JSON format. It will be parsed and the resulting hash values will be passed to the template.

### 3.21.4 Params

For replying with plain text message(s):

**message**

A text message to use as reply.

**messages**

<Array> A list of text messages to respond with.

To reply with a template file:

**template\_file**

The name of the template file to use to build the response message.

**data\_from\_issue**

<Array> An optional list of values that will be extracted from the issue's body and used to fill the template.

Calling an external service:

**external\_service**

All the configuration for the service is nested under this param. Possible options are:

**name**

*Required.* The name for this service.

**url**

*Required.* The url to call.

**method**

The HTTP method to use. Valid values: [get, post]. Default is **post**.

**template\_file**

The optional template file to use to build the response message after the external call.

**headers**

*<Array>* An optional list of *key: value* pairs to be passed as headers in the external service request.

**data\_from\_issue**

*<Array>* An optional list of values that will be extracted from the issue's body and used to fill the template.

**query\_params**

*<Array>* An optional list of params to add to the query of the external call. Common place to add API\_KEYS or other authentication info.

**mapping**

*<Array>* An optional mapping of variable names in the query of the external service call.

### 3.21.5 Examples

Simplest use case:

```
...
responders:
  goodbye:
    message: "Congratulations on the acceptance of your paper!"
...
```

Multiple messages and a template only if label present:

```
...
responders:
  goodbye:
    if:
      labels:
        - accepted
      messages:
        - "Congratulations on the acceptance of your paper!"
        - "Review process finished. Closing the issue."
      template_file: goodbye.md
      data_from_issue:
        - repository
        - doi
...
```

Calling an external service:

```
...
responders:
```

(continues on next page)

(continued from previous page)

```
goodbye:
  external_service:
    url: https://dummy-external-service.herokuapp.com/code-analysis
    method: post
    query_params:
      secret: A1234567890Z
    data_from_issue:
      - target-repo
    mapping:
      id: issue_id
...

```

When a new issue is closed the responder will send a POST request to `https://dummy-external-service.herokuapp.com/code-analysis` with a JSON body:


```
{
  "secret": "A1234567890Z", # declared in the query_params setting
  "target-repo": "...",    # the value is extracted from the body of the issue
  "id": "...",             # the value corresponds to issue_id, it has been mapped to id
  "repo": "...",          # the origin repo where the invocation happend
  "sender": "...",        # the user invoking the command
  "bot_name": "...",      # the bot user name that will be responding
}
```

And the response from the external service will be posted as a comment in the original issue.


### 3.21.6 In action

#### Text messages and template file:

- Replying with a template once the issue is closed:

 **editor** closed this

---

 **editorialbot** commented

🎉🎉🎉 Congratulations on your paper acceptance! 🎉🎉🎉

If you would like to include a link to your paper from your README use the following code:

```

Markdown:
[![DOI](https://joss.theoj.org/papers/10.21105/joss.04060/status.svg)](https://doi.org/10.21105/joss.04060)

HTML:
<a style="border-width:0" href="https://doi.org/10.21105/joss.04060">
  
</a>

reStructuredText:
.. image:: https://joss.theoj.org/papers/10.21105/joss.04060/status.svg
   :target: https://doi.org/10.21105/joss.04060
  
```

This is how it will look in your documentation:

**JOSS** [10.21105/joss.04060](https://doi.org/10.21105/joss.04060)

## 3.22 Close issue command

This responder replies to a specific command closing the issue and possibly adding some labels. Allows *labeling*.

### 3.22.1 Listens to

`@botname <command>`

For example, if you configure the command to be *reject*, it would respond to:

`@botname reject`



### 3.22.2 Settings key

close\_issue\_command

### 3.22.3 Params

**command**

The command this responder will listen to.

**description**

*Optional* String to show when the help command is invoked (if the responder is not hidden).

### 3.22.4 Examples

**Simplest use case:**

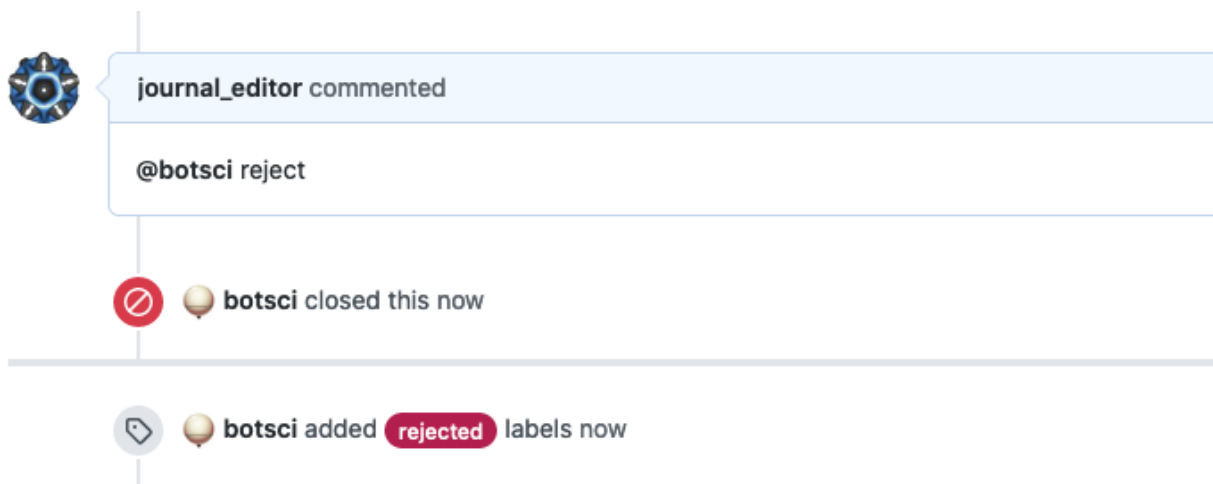
Just close the issue.

```
...
responders:
  close_issue_command:
    command: reject
...
```

**Close issue, add labels, restrict access to editors:**

```
...
responders:
  close_issue_command:
    only: editors
    command: reject
    add_labels:
      - rejected
...
```

### 3.22.5 In action



## 3.23 Update comment

This responder edits the comment triggering the responder updating it with the content of a customizable *template file*.

These updates of the original comment are useful to add content (for instance: checklists) to be modified/updated by the original author of the comment, as they are always allowed to edit their own comments, not requiring to add them as collaborator of the repository/organization.

### 3.23.1 Listens to

```
@botname <command>
```

For example, if you configure the command to be *list pre-acceptance tasks*, it would respond to:

```
@botname list pre-acceptance tasks
```

### 3.23.2 Requirements

The response is generated using a template file that should be available in the repository.

### 3.23.3 Settings key

update\_comment

### 3.23.4 Params

**command**

*Required.* The command this responder will listen to.

**template\_file**

*Required.* The name of the template file to edit the comment with.

**description**

*Optional* String to show when the help command is invoked.

### 3.23.5 Examples

Simplest use case:

```
...
responders:
  update_comment:
    command: list tasks
    template_file: tasks.md
...
```

Limiting use to editors team and adding info to use in the template:

```
...
responders:
  add_remove_checklist:
    only: editors
    command: create pre-acceptance steps checklist
    template_file: editor_final_checklist.md
    data_from_issue:
      - target-repository
      - author-handle
...
```

### 3.23.6 In action

- Invocation:



editor33 commented

@botsci create pre-acceptance steps checklist

- Comment edited by the bot:



editor33 commented • edited ▾

#### Editor Tasks Prior to Acceptance

- ☐ Read the text of the paper and offer comments/corrections (as either a list or a PR)
- ☐ Check the references in the paper for corrections (e.g. capitalization)
- ☐ Set article DOI with @botsci set <DOI here> as archive
- ☐ Set version with @botsci set <version here> as version
- ☐ Double check rendering of paper with @botsci generate pdf
- ☐ Specifically check the references with @botsci check references and ask author(s) to upd
- ☐ Do a 'dry run' of acceptance with @botsci recommend-accept

## 3.24 External start review

This responder checks for the presence of editor and reviewers in an issue and then delegates the creation of a new review issue to an external API call.

### 3.24.1 Listens to

```
@botname start review
```

### 3.24.2 Requirements

The parameters required for the responder to work are the ones configuring the external API call, nested under the `external_call` parameter.

### 3.24.3 Settings key

`external_start_review`

### 3.24.4 Params

#### **external\_call**

*Required.* Nested under this parameter is the configuration for the external call that will start the review. All available subparams are described in the [external\\_service docs](#).

#### **review\_title\_regex**

*Optional.* By default the responder will check that this command has not been triggered from a review issue by checking the title. If it starts with `[REVIEW]`: the command will be rejected. This parameter allows to specify a different string/regex to identify a review issue matching the title.

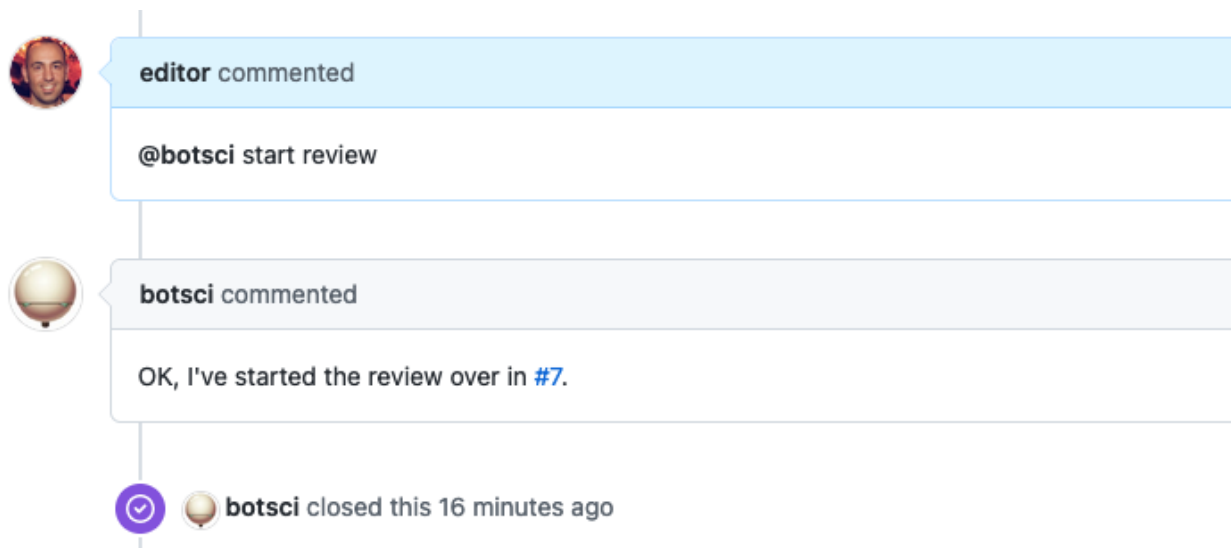
### 3.24.5 Examples

Restricted to editors, respond with a template and close the issue:

```
...
external_start_review:
  only: editors
  external_call:
    url: "https://test.joss.theoj.org/papers/api_start_review"
    query_params:
      secret: <%= ENV['TEST_SECRET'] %>
    mapping:
      id: issue_id
      editor: editor_login
      reviewers: reviewers_logins
  silent: true
  template_file: "review_started.md"
  close: true
...
```

The responder will call `https://test.joss.theoj.org/papers/api_start_review` and the response will be passed to the `review_started.md` template.

### 3.24.6 In action



## 3.25 External service

This responder creates a background job to asynchronously call an external service's API. If the call is successful the response is posted as a comment in the issue (optionally using a template).

### 3.25.1 Listens to

```
@botname <command>
```

For example, if you configure the command to be *analyze code*, it would respond to:

```
@botname analyze code
```

### 3.25.2 Requirements

Some parameters are required for the responder to work: the `name` of the service, the `command` to invoke it, and the `url` of the call. All can be set using the settings YAML file.

### If using a template

If you want to use a template to respond, Buffy will look for the file declared in the `template_file` param in the target repo, in the location specified with the `template_path` setting (by default `.buffy/templates`). In short: the `template_file` should be located in the `template_path`.

The response from the external service should be in JSON format. It will be parsed and the resulting hash values will be passed to the template, where they can be used with the syntax:

```
{{variable_name}}
```

### 3.25.3 Settings key

`external_service`

### 3.25.4 Params

#### General

**name**

*Required.* The name for this service.

**command**

*Required.* The command this responder will listen to.

**description**

The description of the service. It will show in the help command if the responder is not hidden.

**example\_invocation**

Optional string to show as an example of the command being used when the help command is invoked.

**message**

An optional message to reply when the command is received, before the external service is called.

#### Configuring the request

**url**

*Required.* The url to call.

**method**

The HTTP method to use. Valid values: [get, post]. Default is **post**.

**headers**

<Array> An optional list of *key: value* pairs to be passed as headers in the external service request.

**query\_params**

<Array> An optional list of params to add to the query of the external call. Common place to add API\_KEYS or other authentication info.

**data\_from\_issue**

<Array> An optional list of values that will be extracted from the issue's info or issue's body and sent as query params to the service call. Available info includes: *issue\_id*, *issue\_author*, *repo*, *sender*, *bot\_name*, and any variable included in the body of the issue. Also if the command matches any data it will be available as *match\_data\_1*, *match\_data\_2*, etc.

**mapping**

<Array> An optional mapping of variable names in the query of the external service call.

#### Configuring the response

**template\_file**

The optional template file to use to build the response message if the response from the external service is successful.

**success\_msg**

Optional message to respond with if the service call is successful.

**error\_msg**

Optional message to respond with if the service call fails with a 400/500 response.

**silent**

<Boolean> Optional parameter, if **true** the responder won't reply after the external service is called (*template\_file*, *success\_msg* and *error\_msg* will overwrite this if present). Default is **false**.

**add\_labels**

<Array> Optional parameter. Labels to add to the issue if the external service call is successful.

**remove\_labels**

<Array> Optional parameter. Labels to remove from the issue if the external service call is successful.

**close**

<Boolean> Optional parameter, if **true** the responder will close the issue if the external service call is successful. Default is **false**.

### 3.25.5 Examples

A simple case using a template:

```
...
external_service:
  name: cat_facts
  command: tell me something about cats
  description: Random facts about cats
  url: "https://cat-fact.herokuapp.com/facts/random"
  method: get
  query_params:
    animal_type: cat
    amount: 1
  template_file: cats.md
...
```

The request will be `https://cat-fact.herokuapp.com/facts/random?animal_type=cat&amount=1` and the response will be passed to the *cats.md* template.

A complete example:

```
...
external_service:
  - code_quality:
      only: editors
      command: analyze code
      description: Reports on the quality of the code
      message: Inspecting code...
      url: https://dummy-external-service.herokuapp.com/code-analysis
      method: post
```

(continues on next page)

(continued from previous page)

```
query_params:
  secret: A1234567890Z
data_from_issue:
  - target-repo
mapping:
  id: issue_id
...
```

Once the responder is invoked it will reply with “*Inspecting code...*” as a comment in the issue. Later, a POST request will be sent to <https://dummy-external-service.herokuapp.com/code-analysis> with a JSON body:

```
{
  "secret": "A1234567890Z", # declared in the query_params setting
  "target-repo": "...",    # the value is extracted from the body of the issue
  "id": "...",             # the value corresponds to issue_id, it has been mapped to id
  "repo": "...",          # the origin repo where the invocation happend
  "sender": "...",        # the user invoking the command
  "bot_name": "...",      # the bot user name that will be responding
}
```

And the response will posted as a comment in the original issue.

### 3.25.6 In action

- In use:



xuanxu commented 2 days ago

@botsci analyze code



botsci commented 2 days ago

Inspecting code...



botsci commented 2 days ago

The quality of the code in <http://github.com/aliadalabs/research-software> is quite good



### With a template as response

- The template file:

3 lines (2 sloc) | 62 Bytes

Here is a random fact about cats:

🐱🐱 {{text}} 🐱🐱

- The JSON response:

The screenshot shows a web browser with the address bar displaying `https://cat-fact.herokuapp.com/facts/random?animal_type=cat&amou`. The browser's developer tools are open to the 'JSON' tab, showing the following JSON response:

```
{
  "used": false,
  "source": "api",
  "type": "cat",
  "deleted": false,
  "_id": "591f98703b9x27150a19c14c",
  "__v": 0,
  "text": "The life expectancy of cats has nearly doubled over the last fifty years.",
  "updatedAt": "2020-08-23T20:20:01.611Z",
  "createdAt": "2018-01-04T01:10:54.673Z",
  "status": {
    "verified": true,
    "sentCount": 1,
    "user": "5a9ac17x3c478810ea6c06381"
  }
}
```

- In use:



xuanxu commented 2 hours ago

@botsci tell me something about cats



botsci commented 2 hours ago

Here is a random fact about cats:

🐱🐱 The life expectancy of cats has nearly doubled over the last fifty years. 🐱🐱

## 3.26 GitHub Action

This responder triggers workflow run on a GitHub Action using the GitHub API. Optionally if the call is successful (not the result of the workflow run but the call to trigger it) a reply message can be posted as a comment in the issue. Allows *labeling*.

### 3.26.1 Listens to

```
@botname <command>
```

For example, if you configure the command to be *compile pdf*, it will respond to:

```
@botname compile pdf
```

### 3.26.2 Requirements

Some parameters are required for the responder to work: the `command` to invoke it, and the `workflow_repo` and `workflow_name` values to identify the action to run. All can be set using the settings YAML file.

### 3.26.3 Settings key

github\_action

### 3.26.4 Params

**command**

*Required.* The command this responder will listen to.

**description**

The description of the action this command runs. It will show in the help command if the responder is not hidden.

**example\_invocation**

*Optional* String to show as an example of the command being used when the help command is invoked.

**workflow\_repo**

*Required.* The repo to run the action on, in *org/name* format.

**workflow\_name**

*Required.* Name of the workflow to run.

**workflow\_ref**

*Optional.* The git ref for the GitHub action to use. Defaults to *main*.

**message**

An optional message to reply with once the workflow is triggered.

**inputs**

*<Map>* An optional list of params/values to pass as inputs to the GitHub Action.

**data\_from\_issue**

*<Array>* An optional list of fields from the body of the issue to pass as inputs to the GitHub Action.

**mapping**

*<Map>* An optional mapping of variable names to add to the inputs.

You can use this action to run other responder(s) after after the GitHub action is triggered:

**run\_responder**

Allows to call a different responder. Subparams are:

**responder\_key**

*Required.* The key to find the responder in the config file.

**responder\_name**

*Optional.* The name of the responder in the config file if there are several instances under the same responder key.

**message**

*Optional.* The message to trigger the responder with.

If you want to run multiple responders, use an array of these subparams.

### 3.26.5 Examples

A complete example:

```
...
github_action:
  only: editors
  command: compile pdf
  description: Generates a PDF based on the paper.md file in the repository
  workflow_repo: openjournals/reviews
  workflow_name: compile-pdf.yml
  inputs:
    file: paper.md
  data-from-issue:
    - branch
    - target_repository
  mapping:
    repository: target_repository
    number: issue_id
...
```

Once the responder is invoked it triggers the *compile-pdf.yml* workflow on the *openjournals/reviews* repository passing to it the *file*, *repository*, *branch* and *number* inputs.

## 3.27 Wrong command

This is a special responder that replies when Buffy receives a command directed to the bot that no responder understand. By default it replies with:

```
I'm sorry human, I don't understand that. You can see what commands I support by typing:
@botname help
```

But the reply can be configured to be a custom message or to use a template.

### 3.27.1 Listens to

```
@botname whatever is not a command to other responder
```

### 3.27.2 Settings key

If using default reply this responder doesn't need to be added to the config file. Otherwise:

```
wrong_command
```

### 3.27.3 Params

**ignore**

*Optional.* If *true* this responder won't act. Default value: *false*.

**template\_file**

*Optional.* A template file to use to build the response message.

**message**

*Optional.* A text message to use as reply.

### 3.27.4 Examples

**Simplest use case:**

Nothing added to the config file, it will reply the default response

```
...  
  responders:  
...  
...  
...
```

**Deactivate responder:**

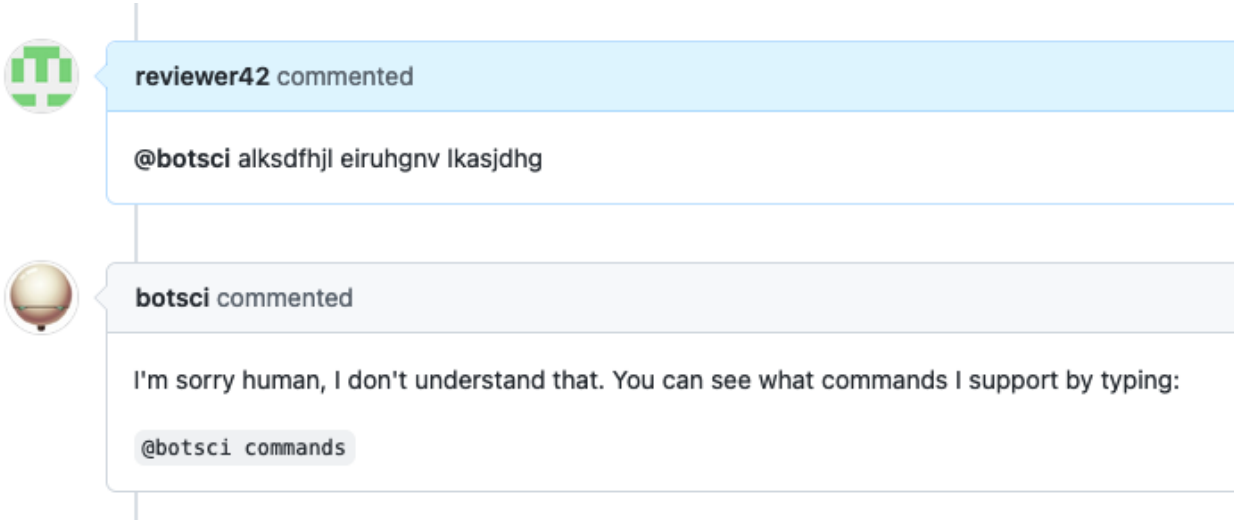
```
...  
  responders:  
    wrong_command:  
      ignore: true  
...  
...  
...
```

**Use custom message:**

```
...  
  responders:  
    wrong_command:  
      message: "Say what?"  
...  
...  
...
```

### 3.27.5 In action

- Unknown command:



## 3.28 ROpenSci :: Reviewers & due date

This responder can be used to add/remove a user to/from the reviewers list in the body of the issue. It also sets a due date for the review and updates that info in the body of the issue and in the reply comment. This responder will also update Airtable adding entries to the reviewers and reviews tables, and creating if still not present entries in the packages and authors tables. Allows *labeling*, that will take effect when the second reviewer is assigned.

### 3.28.1 Listens to

```
@botname add @username to reviewers
```

```
@botname remove @username from reviewers
```

### 3.28.2 Requirements

The body of the issue should have a couple of placeholders marked with HTML comments: the *reviewers-list* and the *due-dates-list*

```
<!--reviewers-list--> <!--end-reviewers-list-->
<!--due-dates-list--> <!--end-due-dates-list-->
```

### 3.28.3 Settings key

ropensci\_reviewers

### 3.28.4 Params

#### due\_date\_days

<Integer> Optional. The number of days from the moment a reviewer is assigned to the due date for the review. Default value is **21** (three weeks).

#### sample\_value

Optional. A sample value string for the username field. It is used for documentation purposes when the [Help responder](#) lists all available responders. Default value is **xxxxx**.

#### no\_reviewer\_text

Optional. The text that will go in the removed reviewer place to state there's no one assigned. Default value is **TBD**.

#### add\_as\_assignee

<Boolean> Optional. If true, the new reviewer will be added as assignee to the issue. Default value is **false**.

#### add\_as\_collaborator

<Boolean> Optional. If true, the new reviewer it will be added as collaborator to the repo. Default value is **false**.

#### reminder

Used to configure automatic reminders. See next.

Automatic reminders: To configure an automatic reminder for the reviewers the **reminder** param can be used with two nested options under it:

#### days\_before\_deadline

<Integer> Optional. Configure when the reminder will be posted (how many days before the deadline for the review). Default value: **4**

#### template\_file

The template file to use for the reminder (will receive variables: *reviewer*, *days\_before\_deadline* and *due\_date*).

For the **Airtable** connection to work two parameters must be present in the **env** section of the settings file, configured using environment variable:

```
...
env:
  airtable_api_key: <%= ENV['AIRTABLE_API_KEY'] %>
  airtable_base_id: <%= ENV['AIRTABLE_BASE_ID'] %>
...
```

### 3.28.5 Examples

Simplest case:

```
...
  responders:
    ropensci_reviewers:
...

```

With labeling, changing no\_reviewer\_text, setting a reminder, limiting access and only if there's an editor already assigned:

```
...
  responders:
    ropensci_reviewers:
      only:
        - editors
      if:
        role_assigned: editor
      no_reviewer_text: "Pending"
      add_labels:
        - 3/reviewer(s)-assigned
      remove_labels:
        - 2/seeking-reviewer(s)
      reminder:
        days_before_deadline: 4
        template_file: reminder.md
...

```

### 3.28.6 In action

- Initial state:



# My cool package #10



Open

maelle opened this issue 3 hours ago · 15 comments



maelle commented 3 hours ago •

Submitting Author: Name (@github\_handle)  
 Other Authors: (delete if none) Name (@github\_handle)  
 Repository:  
 Version submitted:  
 Editor: @maelle  
 Reviewers: TBD

Archive: TBD  
 Version accepted: TBD

Issue's body with placeholders

- **Invocation:**



maelle commented 3 hours ago

Member Author 😊 ...

@ropensci-review-bot add @awesome\_reviewer to reviewers



ropensci-review-bot commented 3 hours ago

Collaborator 😊 ...

@awesome\_reviewer added to the reviewers list. Review due date is 2021-03-17. Thanks @awesome\_reviewer for accepting to review! Please refer to our [reviewer guide](#)

Assigns first reviewer



maelle commented

@ropensci-review-bot add @mpadge to reviewers



ropensci-review-bot commented

@mpadge added to the reviewers list. Review due date is 2021-03-17. Thanks @ to [our reviewer guide](#).



ropensci-review-bot added the **3/reviewer(s)-assigned** label and removed

- Assigning second reviewer applies labeling:
- Final state:

# My cool package #10



maelle opened this issue 3 hours ago · 15 comments



maelle commented 3 hours ago • edited by ropensci-review-bot ▾

Submitting Author: Name (@github\_handle)  
 Other Authors: (delete if none) Name (@github\_handle)  
 Repository:  
 Version submitted:  
 Editor: @maelle  
 Reviewers: @awesome\_reviewer, @mpadge  
  
 Due date for @awesome\_reviewer: 2021-03-17  
 Due date for @mpadge: 2021-03-17  
 Archive: TBD  
 Version accepted: TBD

Issue's body with reviewers and due dates info

## 3.29 ROpenSci :: Set due date

This responder can be used to add or change the review due date for a current reviewer.

### 3.29.1 Listens to

```
@botname set due date for @reviewer to YYYY-MM-DD
```

### 3.29.2 Requirements

The body of the issue should have the *reviewers-list* and the *due-dates-list* placeholders marked with HTML comments:

```
<!--reviewers-list--> <!--end-reviewers-list-->
<!--due-dates-list--> <!--end-due-dates-list-->
```

The reviewer should be already listed in the reviewers list

The format for the due date must be YYYY-MM-DD

The new due date can not be in the past

### 3.29.3 Settings key

ropensci\_set\_due\_date

### 3.29.4 Examples

Restricted to editors:

```
...
responders:
  ropensci_set_due_date:
    only:
      - editors
...
```

### 3.29.5 In action

- Initial state:



**Author1** commented

Submitting Author Name: AuthorName  
 Submitting Author Github Handle: @author1  
 Repository: <https://github.com/author1/epair>  
 Version submitted: 0.1.0  
 Editor: @maelle  
 Reviewers: @reviewer1, @reviewer33  
  
 Due date for @reviewer1: 2022-03-13  
 Due date for @reviewer33: 2022-06-21  
 Archive: TBD  
 Version accepted: TBD

Initial issue's body

- Invocation:



maelle commented

@ropensci-review-bot set due date for @reviewer33 to 2022-07-30



ropensci-review-bot commented

Review due date for @reviewer33 is now 30-july-2022

Set new due date for a reviewer

- **Final state:**

Issue's body with new due date info

repository: <https://github.com/author1/epan>

Version submitted: 0.1.0

Editor: @maelle

Reviewers: @reviewer1, @reviewer33

Due date for @reviewer1: 2022-03-13

Due date for @reviewer33: 2022-07-30

Archive: TRD

## 3.30 ROpenSci :: Seeking reviewers

This responder changes the review to a `seeking reviewers` mode, adding and removing appropriate labels and responding with a message with further instructions for authors. This responder will also call Airtable to create an entry in the packages table and entries for all authors (author1 and author-others) in the authors tables.

### 3.30.1 Listens to

```
@botname seeking reviewers
```

### 3.30.2 Settings key

```
ropensci_seeking_reviewers
```

### 3.30.3 Params

**template\_file**

The optional template file to use to build the response message.

**add\_labels**

<Array> Optional parameter. Labels to add to the issue.

**remove\_labels**

<Array> Optional parameter. Labels to remove from the issue.

As with any responder interacting with Airtable, two parameters must be present in the `env` section of the settings file, configured using environment variables:

```
...
env:
  airtable_api_key: <%= ENV['AIRTABLE_API_KEY'] %>
  airtable_base_id: <%= ENV['AIRTABLE_BASE_ID'] %>
...
```

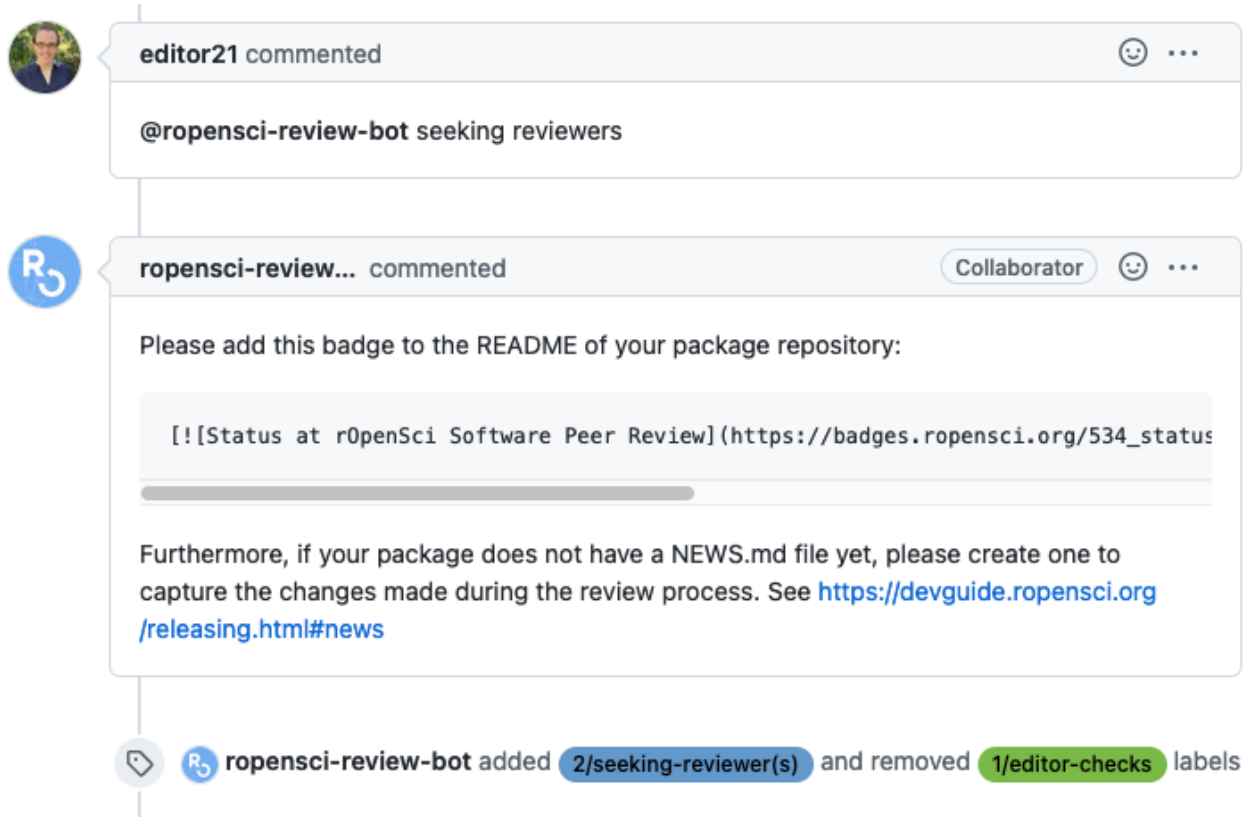
### 3.30.4 Examples

Restricted to editors:

```
...
responders:
  ropensci_seeking_reviewers:
    only:
      - editors
    template_file: badge.md
    remove_labels:
      - 1/editor-checks
    add_labels:
      - 2/seeking-reviewer(s)
...
```

### 3.30.5 In action

Run by an editor:



The screenshot shows a GitHub issue thread. At the top, a comment from **editor21** says "@ropensci-review-bot seeking reviewers". Below it, a comment from **ropensci-review...** (labeled as a Collaborator) responds. The response text says: "Please add this badge to the README of your package repository:" followed by a code block containing a GitHub badge URL: `[[Status at rOpenSci Software Peer Review](https://badges.ropensci.org/534_status)]`. Below the code block, it says: "Furthermore, if your package does not have a NEWS.md file yet, please create one to capture the changes made during the review process. See <https://devguide.ropensci.org/releasing.html#news>". At the bottom of the thread, a label summary shows: "ropensci-review-bot added 2/seeking-reviewer(s) and removed 1/editor-checks labels".

## 3.31 ROpenSci :: Approve

This responder is used to approve a package. It performs a series of tasks:

- Adds `date-accepted` to the body of the issue
- Clears reviewers' `current_assignment` in AirTable
- Creates a new team named like the package-name and invites the creator of the issue to it (owner right needed)
- Can reply with a *template*
- Allows *labeling*
- Closes the issue
- If the submission-type is `stats` it checks if `stasgrade` is present and if so adds the proper label

### 3.31.1 Listens to

```
@botname approve package-name
```

### 3.31.2 Requirements

The *package-name* must be specified in the command, otherwise an error message will be sent as reply.

If the *submission-type* of the issue is *stats*, then for the responder to work there must be a valid value for a *statsgrade* variable (marked with HTML comments) in the body of the issue:

```
# the responder will add the label: '6/approved-silver'
<!--statsgrade-->silver<!--end-statsgrade-->
```

### 3.31.3 Settings key

ropensci\_approve

### 3.31.4 Params

For the **Airtable** connection to work two parameters must be present in the `env` section of the settings file, configured using environment variable:

```
...
env:
  airtable_api_key: <%= ENV['AIRTABLE_API_KEY'] %>
  airtable_base_id: <%= ENV['AIRTABLE_BASE_ID'] %>
...
```

For labeling the approved *stats* submissions an external service is used to get the proper versioned label. The url for the external service is by default: `http://138.68.123.59:8000/stats_badge`. This value can be changed using the optional `:stats_badge_url` param:

```
...
responders:
  ropensci_approve:
    only: editors
    stats_badge_url: https://test.ropensci:3030
...
```

### 3.31.5 Examples

Simplest case:

```
...
responders:
  ropensci_approve:
...
```

With labeling, template response, limiting access and only if there's an editor already assigned:

```
...
responders:
  ropensci_approve:
    only: editors
    template_file: approved.md
    data_from_issue:
      - reviewers-list
    remove_labels:
      - 5/awaiting-reviewer(s)-response
    add_labels:
      - 6/approved
...
```

## 3.32 ROpenSci :: Finalize transfer

This responder is used to assing a recent approved and transfered package to a rOpenSci team. It needs owner rights to work. It performs a series of tasks:

- Checks for the presence of the package-name repo in the rOpenSci GitHub organization
- Creates a new team named like the package-name and invites the creator of the issue to it, if the team does not exists already.
- Adds the package-name repo to the package-name team with admin rights so the members of the team can manage it

### 3.32.1 Listens to

```
@botname finalize transfer of package-name
```

### 3.32.2 Requirements

The *package-name* must be specified in the command, otherwise an error message will be sent as reply. The bot must have owner rights.

### 3.32.3 Settings key

ropensci\_finalize\_transfer



### 3.32.4 Example:

```
...
  responders:
    ropensci_finalize_transfer:
      only: editors
...
```

## 3.33 ROpenSci :: Invite author

This responder is used by the author of an approved package to receive an invitation to join the team that will manage the package and will allow them to transfer it to rOpenSci. Usually this invitation is sent automatically when the package is approved but it expires in a week. This responder allows the author to have the invitation sent again.

### 3.33.1 Listens to

```
@botname invite me to ropensci/package-name
```

### 3.33.2 Requirements

The command must be run by the author of the package (the user that created the review issue), otherwise an error message will be sent as reply.

### 3.33.3 Settings key

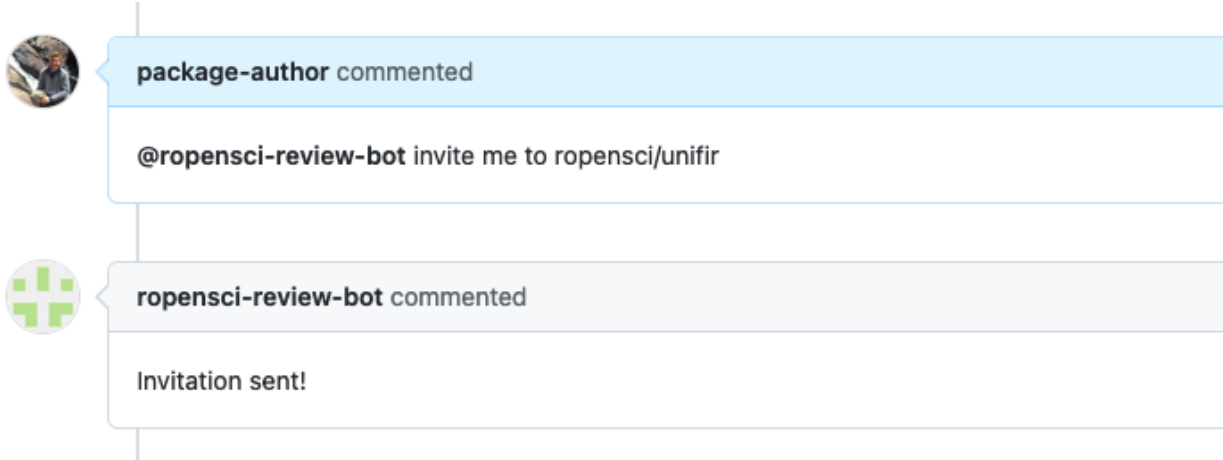
ropensci\_invite\_author

### 3.33.4 Example:

Allow the command to run only if package is already approved:

```
...
  responders:
    ropensci_invite_author:
      if:
        labels: 6/approved
        reject_msg: "Can't invite author because the package is not approved yet"
...
```

### 3.33.5 In action



## 3.34 ROpenSci :: Mint

This responder mints a submission: it can be used to add a valid badge grade value (currently *bronze/silver/gold*) to the kind of submissions accepting them (currently *stats*)

### 3.34.1 Listens to

```
@botname mint <grade>
```

Where <grade> must be a valid value. For example:

```
@botname mint silver
```

### 3.34.2 Requirements

The responder will read the value of *submission-type* in the body of the issue, for it to work this value must equal *stats*, then it will update (or add) the value of the *statsgrade* in the body of the issue.

### 3.34.3 Settings key

`ropensci_mint`

### 3.34.4 Examples

Only available to editors:

```
...
responders:
  ropensci_mint:
    only: editors
...
```

### 3.34.5 In action

- **Initial state:**

Issue's body with correct submission type

  vgnierard opened this issue 16 days ago · 4 comments



submitter\_37 commented

Submitting Author: Software Author (@submitter\_37)  
Repository: <https://github.com/great-package/kgram>  
Submission type: stats

- **Invocation:**



noamross commented

@ropensci-review-bot mint gold



ropensci-review-bot commented

Done, gold minted!

- **Final state:**

Issue's body updated with the badge grade

Open 1 of 19 tasks vgherard opened this issue 10 days ago · 4 comments



submitter\_37 commented

Badge grade: gold

Submitting Author: Software Author (@submitter\_37)

Repository: <https://github.com/great-package/kgram>

Submission type: stats

### 3.35 ROpenSci :: Submit review

This responder can be used to update Airtable entries with a review url, duration and date in the reviews table. Once the number of reviews in Airtable equals the number of reviewers in the issue a message will be configured for 12 days later to remind authors to submit their response.

#### 3.35.1 Listens to

```
@botname submit review <REVIEW_URL> time <REVIEW_HOURS>
```

Where <REVIEW\_URL> must be a valid link to a comment in the issue and <REVIEW\_HOURS> is numeric. For example:

```
@botname submit review https://github.com/ropensci/software-review/issues/338
↪ #issuecomment-536199121 time 7.5
```

#### 3.35.2 Requirements

REVIEW\_URL must be a complete url pointing to a comment in the review issue.

REVIEW\_HOURS is numeric. Example of valid values: 4, 10.5, 7, 5

#### 3.35.3 Settings key

ropensci\_submit\_reviews

### 3.35.4 Params

#### **label\_when\_all\_reviews\_in**

*Optional* Labeling to add to the issue once the number of reviews in Airtable equals the number of reviewers in the issue.

#### **unlabel\_when\_all\_reviews\_in**

*Optional* Labeling to remove from the issue once the number of reviews in Airtable equals the number of reviewers in the issue.

For the **Airtable** connection to work two parameters must be present in the `env` section of the settings file, configured using environment variable:

```
...
env:
  airtable_api_key: <%= ENV['AIRTABLE_API_KEY'] %>
  airtable_base_id: <%= ENV['AIRTABLE_BASE_ID'] %>
...
```

### 3.35.5 Examples

Simplest case:

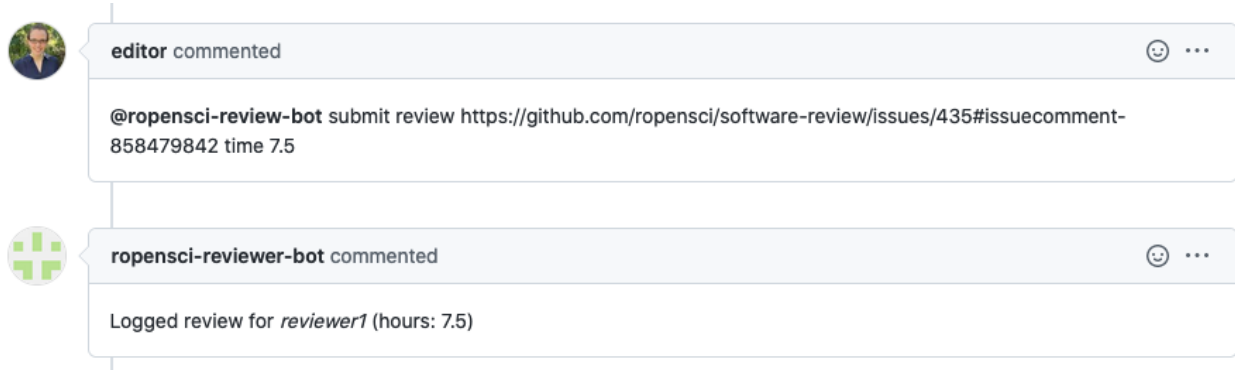
```
...
responders:
  ropensci_submit_reviews:
...
```

With labeling once all reviews are completed and limiting access to editors:

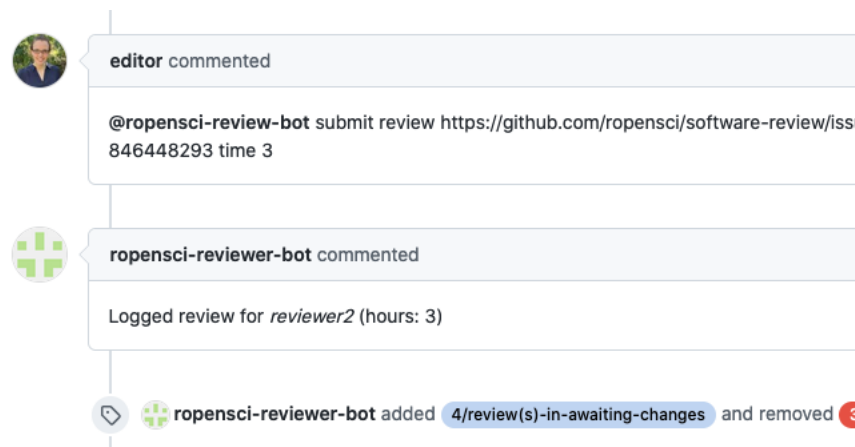
```
...
responders:
  ropensci_submit_reviews:
    only:
      - editors
  label_when_all_reviews_in: "4/review-in-awaiting-changes"
  unlabel_when_all_reviews_in: "3/reviewer(s)-assigned"
...
```

### 3.35.6 In action

- Invocation: `Log first review`



The screenshot shows two comments in a GitHub issue. The first comment is from 'editor' and says: '@ropensci-review-bot submit review https://github.com/ropensci/software-review/issues/435#issuecomment-858479842 time 7.5'. The second comment is from 'ropensci-reviewer-bot' and says: 'Logged review for reviewer1 (hours: 7.5)'.



The screenshot shows three comments in a GitHub issue. The first comment is from 'editor' and says: '@ropensci-review-bot submit review https://github.com/ropensci/software-review/iss 846448293 time 3'. The second comment is from 'ropensci-reviewer-bot' and says: 'Logged review for reviewer2 (hours: 3)'. The third comment is from 'ropensci-reviewer-bot' and says: 'added 4/review(s)-in-awaiting-changes and removed 3'.

- Logging last review applies labeling:

## 3.36 ROpenSci :: Submit author response

Used by the authors of a submission after responding to a review or to reviewer's comments, this responder can be used to create Airtable entries with a response url and a date for the reviewed package.

### 3.36.1 Listens to

```
@botname submit response <AUTHOR_RESPONSE_URL>
```

Where <AUTHOR\_RESPONSE\_URL> must be a valid link to a comment in the issue. For example:

```
@botname submit response https://github.com/ropensci/software-review/issues/550
↪ #issuecomment-1229032049
```

### 3.36.2 Requirements

AUTHOR\_RESPONSE\_URL must be a complete url pointing to a comment in the review issue.

### 3.36.3 Settings key

ropensci\_submit\_author\_response

### 3.36.4 Params

For the **Airtable** connection to work two parameters must be present in the env section of the settings file, configured using environment variable:

```
...
env:
  airtable_api_key: <%= ENV['AIRTABLE_API_KEY'] %>
  airtable_base_id: <%= ENV['AIRTABLE_BASE_ID'] %>
...
```

### 3.36.5 Examples

Simplest case:

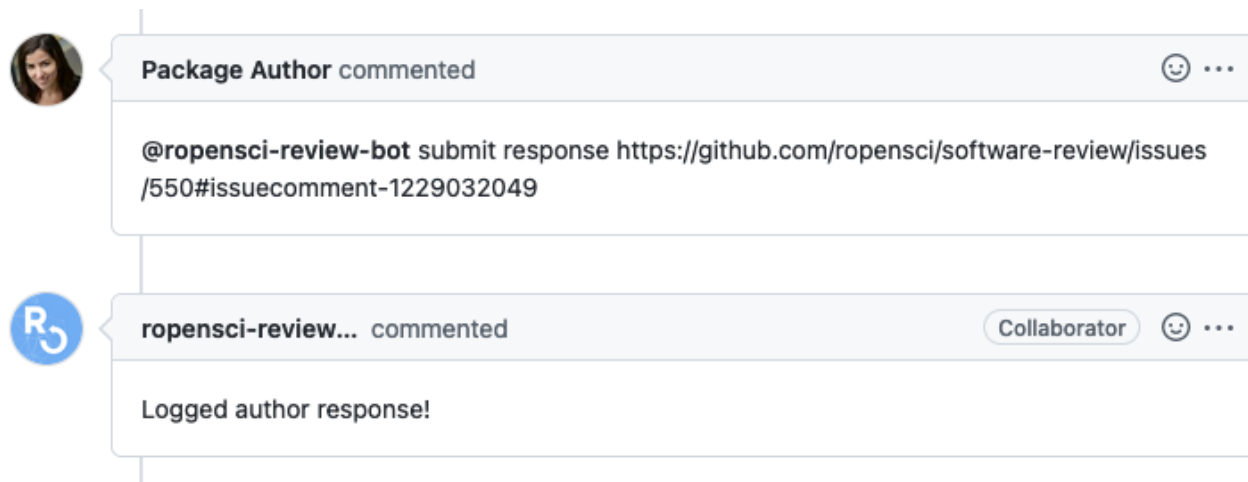
```
...
responders:
  ropensci_submit_author_response:
...
```

Use restricted to authors set in the body of the issue:

```
...
responders:
  ropensci_submit_author_response:
    authorized_roles_in_issue:
      - author1
      - author-others
...
```

### 3.36.6 In action

- **Invocation:** Log author's response



## 3.37 ROpenSci :: On hold

This responder is used by an editor to put the submission on hold (by default for 90 days, but that is configurable). The responder will label the issue with the holding label and once the time limit is reached the editor will be pinged to review the holding status and possibly close the issue.

### 3.37.1 Listens to

```
@botname put on hold
```

### 3.37.2 Settings key

```
ropensci_on_hold
```

### 3.37.3 Params

#### **on\_hold\_label**

*Optional.* The label to add to the issue. By default is **holding**

#### **on\_hold\_days**

*Integer* An optional number of days to have the issue on hold before reminding the editor. Default value is **90**.

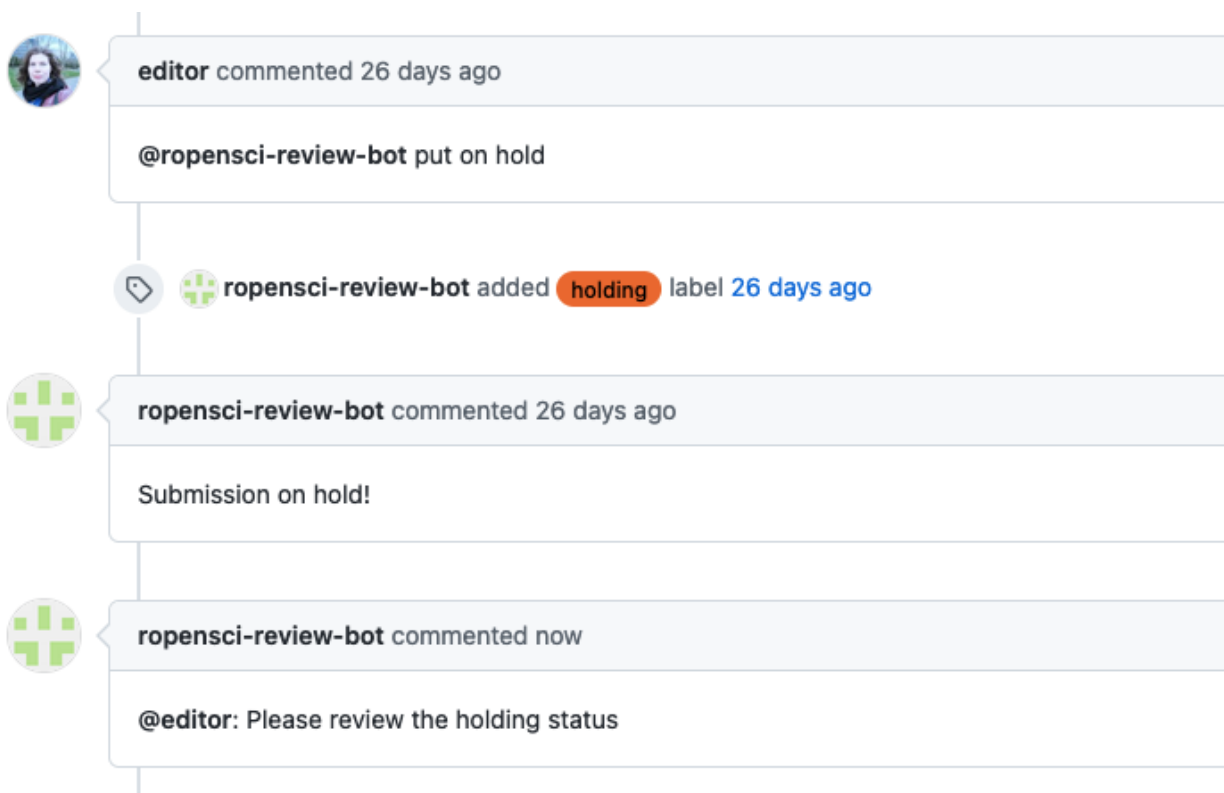



### 3.37.4 Example:

Allow the command to be run only by editors, and set reminder to 26 days:


```
...
responders:
  ropensci_on_hold:
    only: editors
    on_hold_days: 26
...
```


### 3.37.5 In action




 **editor** commented 26 days ago

**@ropensci-review-bot** put on hold

 **ropensci-review-bot** added **holding** label 26 days ago

 **ropensci-review-bot** commented 26 days ago

Submission on hold!

 **ropensci-review-bot** commented now

**@editor:** Please review the holding status



## LABELING

Several Buffy responders allow labeling. A responder allowing labeling means that if the responder finish successfully its main task, it can add and/or remove labels to the issue if they are specified in the settings file.

### 4.1 Settings

Responders allowing labeling will accept in their settings two keys:

**add\_labels**

an optional Array of labels to add

**remove\_labels**

an optional Array of labels to remove

**Example:**

```
...
responders:
  example_responder:
    add_labels:
      - review-finished
      - recommend publication
    remove_labels:
      - pending-review
...
```

If the example responder is successfull the `review-finished` and `recommend publication` labels will be added and the `pending-review` label will be removed from the issue.

### 4.2 Responders listening to Add/Remove actions

Some responders listen to two opposite add and remove actions (for instance the *[add\\_remove\\_assignee responder](#)*). In these cases, the add action will process the labeling normally –adding the specified `:add_labels` and removing the `:remove_labels`– and the remove action will undo that labeling, i.e. removing the `:add_labels` and adding the labels from the `:remove_labels` setting.



## USING TEMPLATES

Several Buffy responders can reply with a template. Please read each Responder documentation to know if a specific Responder allows this option.

### 5.1 Template files

Templates must be created in the repository using Buffy. Every template is a different file in the repo. To make use of them Buffy needs to know where the templates are located, and the individual name of each template file. As the comments in GitHub issues are rendered using markdown, usually the templates will be plain text or .md files, but that is not mandatory for Buffy to use them.

#### 5.1.1 Location

Buffy will look for the templates in the target repository. By default it will look under the `.buffy/templates` dir. This value can be modified in the settings file with the `templates_path` setting. If present, the value of this setting will be considered the relative value in the target repo where templates are located.

#### 5.1.2 Name

In the responders allowing templates for replies, the template is specified using the `template_file` setting for that responder. Value should be the name of the file including the extension if it has one.

#### 5.1.3 Example

If Buffy is configured to work on a repo with address `https://github.com/scientific-journal/astronomy` and the `settings.yml` file has the following value for `template_path`:

```
buffy:
  templates_path: .templates
...
```

and you declare a template in a responder using `template_file` with this value:

```
...
responders:
  welcome_template:
    template_file: welcome.md
...
```

Buffy will use the content of <https://github.com/scientific-journal/astronomy/.templates/welcome.md> to respond.

## 5.2 Populating templates

The content of a template can include placeholders to be filled with the actual values of a variable. The syntax is:

```
{{variable_name}}
```

When rendering a template, Buffy will use a hash of **key:value** pairs. When a placeholder is found in the template, it will look up for the corresponding key in the hash and insert the value in the template. The hash will always include *at least*:

- **issue\_id**: The id of the issue
- **issue\_author**: The handle of the user that opened the issue
- **repo**: the name of the repository
- **sender**: the handle of the user creating the comment/issue triggering the responder
- **bot\_name**: the name of the bot user responding

The hash can also include fields extracted from the body of the issue. To add fields use the `data_from_issue` setting. For example, to have the `target-repository` and `author` values from the issue available in the template this would do:

```
...
responders:
  welcome_template:
    template_file: welcome.md
    data_from_issue:
      - target-repository
      - author
...
```

Check each responder documentation for details on other values available to use in templates.

## CREATING A CUSTOM RESPONDER

Buffy will load and make available any responder that is located in the `app/responders` directory. The simplest way to organize your responders is to add them in a subfolder inside the `responders` dir, defining a module for the custom responders.

During this guide as an example, we'll create a simple responder to get the time.

### 6.1 Responder structure

A responder is a ruby class containing five elements:

- **keyname**: the handle for the responder in the configuration file
- **define\_listening** method: a place to declare what events the responder is listening to
- **process\_message** method: the code to perform whatever the responder does
- **description** method: to add a short description of the responder for documenting purposes
- **example\_invocation** method: to show users how to invoke the responder

#### 6.1.1 The Responder Ruby class

A responder object is a class inheriting from the `Responder` class, so you should require the `Responder` class located in `/lib` and create a child class.

When initialized, a responder will have accessor methods for the name of the bot (`bot_name`) and for the parameters of the responder coming from the config file (`params`).

For our example we add a `clock_responder.rb` file to the new `app/responders/myorganization` dir.  
It declares the responder class in the `myorganization` module.

```
require_relative '../..lib/responder'

module Myorganization
  class ClockResponder < Responder
    end
  end
end
```

## 6.1.2 Keyname

Using keyname you can define the handle for the responder to be used in the configuration file. Using a symbol is ok. For our example we'll just use *clock*:

```
require_relative '../..lib/responder'

module Myorganization
  class ClockResponder < Responder
    keyname :clock
  end
end
```

Now we can use the responder adding it to the *config.yml* file:

```
...
responders:
  clock:
...
```

## 6.1.3 Define listening

The `define_listening` method is the place to specify what the responder is listening to. You can set values for two instance variables here:

- **@event\_action**: the action that triggered the event the responder will listen to
- **@event\_regex**: (optional) a regular expression the text body of the event (a comment or the body of an issue) should match for the responder to respond

When an event is sent from the reviews repository to Bufly, only responders that match action and regex (if present) will be run.

### Event action

- If you are listening to creation of issues, *@event\_action* should be "issues.opened".
- If you are listening to new comments, *@event\_action* should be "issue\_comment.created".

### Event regex

The *@event\_regex* variable is where the syntax of every specific command is declared. If it is *nil* the responder will respond to every event that matches *@event\_action*.

Inside this method you have available the name of the bot in the *@botname* instance variable and all the parameters for this responder from the config file in the *@params* instance variable.

For our example, we will be listening to comments and we want the command to be "what time is it?":

```
require_relative '../..lib/responder'

module Myorganization
  class ClockResponder < Responder
```

(continues on next page)



(continued from previous page)

```

keyname :clock

def define_listening
  @event_action = "issue_comment.created"
  @event_regex = /\A@#{bot_name} what time is it\?\s*\z/i
end
end
end

```

## Mandatory parameters

You can also declare inside this method which parameters are required in the configuration using `required_params`. This will create a reader method for every required parameter.

For example, we could make the command for invoking our responder mandatory and declared in the `config.yml` file instead that in our regex, that way the command for our responder can be changed and be easily configured:

```

require_relative '../../lib/responder'

module Myorganization
  class ClockResponder < Responder
    keyname :clock

    def define_listening
      required_params :command

      @event_action = "issue_comment.created"
      @event_regex = /\A@#{bot_name} #{command}\s*\z/i
    end
  end
end

```

now the command must be added to the config file or the responder will error and not run:

```

...
responders:
  clock:
    command: tell me the time
...

```

But we don't want to be too strict so, we'll allow the command to be changed but by default we'll have one. For that we'll use an auxiliary instance method:

```

require_relative '../../lib/responder'

module Myorganization
  class ClockResponder < Responder
    keyname :clock

    def define_listening
      @event_action = "issue_comment.created"
      @event_regex = /\A@#{bot_name} #{clock_command}\s*\z/i
    end
  end
end

```

(continues on next page)

(continued from previous page)

```

    end

    def clock_command
      params[:command] || "what time is it\\?"
    end
  end
end
end

```

### 6.1.4 Process message

The `process_message` method will be called if an event reaches Buffy and it matches the action and the regex in the `define_listening` method. It accepts a single argument: the message that triggered the call.

This method is the place of all the custom Ruby code needed to perform whatever is the responder does. To interact back with the reviews repository there are several methods available:

- **respond(message)**: will post a comment with the specified *message* string
- **respond\_external\_template(template\_name, locals)**: will post a comment using *a template* and passing it the *locals* variables
- **update\_body(mark, end\_mark, text)**: will update the body of the issue between marks with the passed *text*
- **add\_assignee(user)**: will add the passed *user* to the issue's assignees
- **remove\_assignee(user)**: will remove the passed *user* from the issue's assignees
- **replace\_assignee(old\_user, new\_user)**: will replace the passed *old\_user* with *new\_user* in the issue's assignees
- **process\_labeling**: will add/remove labels as specified in the responder *config params*

If you need to access any matched data from the `@event_regex` you have them available via the `match_data` array.

For our example we'll just reply a comment with the time:

```

require_relative '../../lib/responder'

module Myorganization
  class ClockResponder < Responder
    ...

    def process_message(message)
      respond(Time.now.strftime(" The time is %H:%M:%S %Z, today is %d-%m-%Y "))
    end
  end
end
end

```

### 6.1.5 Description

Use the `description` method to add a short description of what the responder does.

Our example responder replies with the current time:

```
require_relative '../lib/responder'

module Myorganization
  class ClockResponder < Responder
    ...

    def description
      "Get the current time"
    end
  end
end
```

### 6.1.6 Example invocation

To help users understand how to use the responder, use the `example_invocation` to add an example of how the responder is triggered.

In our example responder we'll use the command declared via config or the default one:

```
require_relative '../lib/responder'

module Myorganization
  class ClockResponder < Responder
    ...

    def example_invocation
      "@#{bot_name} #{params[:command]} || 'what time is it?'"
    end
  end
end
```

## 6.2 Sample custom responder

The final version of our clock responder (in `app/responders/myorganization/clock_responder.rb`):

```
require_relative '../lib/responder'

module Myorganization
  class ClockResponder < Responder
    keyname :clock

    def define_listening
      @event_action = "issue_comment.created"
      @event_regex = /\A@#{bot_name} #{clock_command}\s*\z/i
    end
  end
end
```

(continues on next page)

(continued from previous page)

```
def process_message(message)
  respond(Time.now.strftime(" The time is %H:%M:%S %Z, today is %d-%m-%Y "))
end

def clock_command
  params[:command] || "what time is it\\?"
end

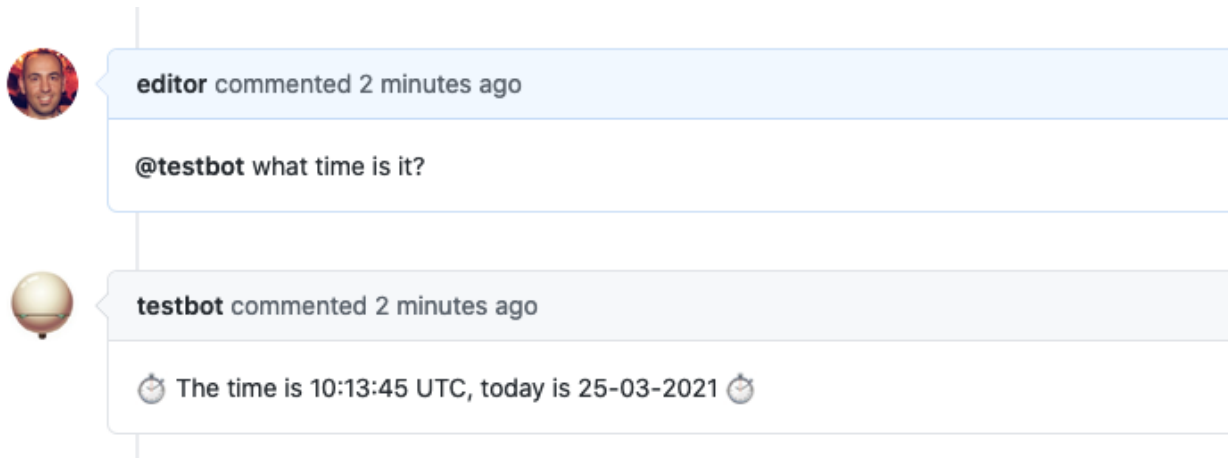
def description
  "Get the current time"
end

def example_invocation
  "@#{bot_name} #{params[:command]} || 'what time is it?'"
end
end
end
```

Adding its key to the configuration file in the responder settings:

```
buffy:
  responders:
    clock:
    ...
```

The responder should be available and ready to use:



## 6.3 Tests

Don't forget to add tests for any new Responder you create. Buffy uses the `RSpec` test framework.

For our sample responder, we would create `spec/responders/myorganization/clock_responder_spec.rb`

```
require_relative "../../spec_helper.rb"

describe Myorganization::ClockResponder do

  subject do
    described_class
  end

  describe "listening" do
    before { @responder = subject.new({env: {bot_github_user: "testbot"}}, {}) }

    it "should listen to new comments" do
      expect(@responder.event_action).to eq("issue_comment.created")
    end

    it "should define regex" do
      expect(@responder.event_regex).to match("@testbot what time is it?")
      expect(@responder.event_regex).to_not match("@testbot whatever")
    end

    it "should allow invocation with custom command" do
      custom_responder = subject.new({env: {bot_github_user: "testbot"}},
                                     {command: "tell me the time"})
      expect(custom_responder.event_regex).to match("@testbot tell me the time")
      expect(custom_responder.event_regex).to_not match("@botsci what time is it?")
    end

    describe "#process_message" do
      before do
        @responder = subject.new({env: {bot_github_user: "botsci"}}, {})
        disable_github_calls_for(@responder)
      end

      it "should respond to github" do
        timenow = Time.now
        expected_response = timenow.strftime(" The time is %H:%M:%S %Z, today is %d-%m-%Y")
        expect(Time).to receive(:now).and_respond(timenow)
        expect(@responder).to receive(:respond).with(expected_response)
        @responder.process_message("@testbot what time is it?")
      end
    end
  end
end
```

You can find more examples of responder specs in the `/spec/responders` directory.