Redmine - Defect #33881

No email notifications for status changes triggered by fixing keywords in repository commits.

2020-08-21 13:00 - Hilmar Blonn

Status:	New	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:	Email notifications	**Estimated time:**	0.00 hour
Target version:			
Resolution:		**Affected version:**	4.1.1

Description

When using fixing keywords in the git commit-messages, the references are added correctly to the issue and the status is changed to closed as intended, but no email notifications are sent out for the status change (for manual status changes the notifications are sent).

```
Environment:
Redmine version              4.1.1.stable
Ruby version                 2.6.6-p146 (2020-03-31) [x86_64-linux]
Rails version                5.2.4.2
Environment                  production
Database adapter             Mysql2
Mailer queue                 ActiveJob::QueueAdapters::AsyncAdapter
Mailer delivery              smtp
SCM:
Subversion                   1.13.0
Git                          2.25.1
Filesystem
Redmine plugins:
redmine_agile                1.5.3
redmine_checklists           3.1.16
redmine_people               1.5.1
redmine_recurring_tasks      0.3.4
redmine_theme_changer        0.4.0
redmine_zenedit              1.0.3
redmineup_tags               2.0.8
status_button                0.1.0
```

History

#1 - 2020-10-26 12:58 - Tobias Grimm

Same here. I digged a little bit deeper:

When I manually trigger fetching the git commits by going to the repositories page, then email sending of status changes triggered by a commit works fine. If the commits are fetched by the redmine:fetch_changesets (via cron), then no email notifications are sent.

Looking at the log I see, that in both cases an ActionMailer::DeliveryJob is enqueued. I also see "Performing ActionMailer::DeliveryJob" in both cases, but only in the first case I see "Rendering mailer/issue_edit.text.erb within layouts/mailer" actually happening.

This seems to be related to the async ActiveJob, which causes the queued delivery jobs to not be executed before the fetch_changesets process ends.

My lazy solution for this was to add a sleep(10) in the fetch_changesets rake task to give the async delivery some time before the rake process ends.

This might already be fixed in a later Redmine/Rails version. I'm running the Debian-packed version from buster-baclports.

```
Environment:
Redmine version              4.0.7.stable
Ruby version                 2.5.5-p157 (2019-03-15) [x86_64-linux-gnu]
Rails version                5.2.3
Environment                  production
Database adapter             Mysql2
Mailer queue                 ActiveJob::QueueAdapters::AsyncAdapter
Mailer delivery              sendmail
SCM:
```

Git	2.20.1
Filesystem	
Redmine plugins:	
no plugin installed	

#2 - 2021-08-02 18:47 - Andrew Skripnikov

Hi, Tobias!

Could you please give me more details what I need to edit to fix this issue.

Seems, it is not fixed in Redmine 4.2.1.stable

Thank you in advance

Looking forward for your reply

#3 - 2021-08-03 12:11 - Tobias Grimm

In redmine.rake (https://www.redmine.org/projects/redmine/repository/entry/trunk/lib/tasks/redmine.rake) in the :fetch_changesets-Task add a sleep after fetching the changesets:

```
task :fetch_changesets => :environment do
  Repository.fetch_changesets
  sleep(5)
end
```

This will give redmine the time to actually deliver the emails before the process ends.

#4 - 2021-08-05 17:50 - Andrew Skripnikov

Thank you very much!
This hales a lot :)

#5 - 2021-08-06 23:54 - Mischa The Evil

FYI, I think you're all using an in-memory queue which might be the issue here.

From source:/tags/4.0.7/app/models/mailer.rb#L604 and source:/tags/4.1.1/app/models/mailer.rb#L620:

```
    # Using the asynchronous queue from a Rake task will generally not work because
    # Rake will likely end, causing the in-process thread pool to be deleted, before
    # any/all of the .deliver_later emails are processed
```

FWIW: Jean-Philippe Lang wrote in Redmine 4.0.0, 3.4.7 and 3.3.9 released:

> Email delivery now relies on Rails ActiveJob. Emails are sent asynchronously by default. But you should consider configuring a persistent backend for ActiveJob since the default uses an in-memory queue that is not well suited for production environnements: https://guides.rubyonrails.org/v5.2/active_job_basics.html#job-execution