

Redmine与Gitlab深度集成 - CSDN博客

转载 2017年11月13日 17:13:50

• 982

一. 要求

- 1. 在redmine的issue页面能够看到该问题相关的提交记录
- 2. gitlab有提交后自动触发Redmine获取更新
- 3. 通过commit message能够关联Redmine的issue，设置issue状态，设置任务耗时

二. 实现

2.1. redmine设置启用版本库

设置如下图：

配置

一般显示认证API项目问题跟踪文件邮件通知接收邮件版本库

启用 SCM

	命令	版本
<input type="checkbox"/> Subversion		
<input type="checkbox"/> Darcs		
<input type="checkbox"/> Mercurial		
<input type="checkbox"/> Cvs		
<input type="checkbox"/> Bazaar		
<input checked="" type="checkbox"/> Git	git	2.7.4
<input type="checkbox"/> Filesystem		

您可以在config/configuration.yml中配置您的SCM命令。请在编辑后，重启Redmine应用。

自动获取程序变更

启用用于版本库管理的Web Service

版本库管理网页服务 API 密钥2CivrikuhwEOI2DmaCJ9生成一个key

在文件变更记录页面上显示的最大修订版本数量100

在提交信息中引用和解决问题

用于引用问题的关键字refs,references,issueID

允许引用/修复所有其他项目的问题

激活时间日志

记录的活动默认

跟踪标签	用于解决问题的关键字	应用后的状态	% 完成	
全部	finish,ok	解决	100 %	
全部	start,started	进行中	10 %	
全部	close,closed	关闭	100 %	

保存

- 1. 启用SCM：本地只安装了git，所以其他的取消掉
- 2. 启用用于版本库管理的Web Service：需要勾选，实现gitlab的web_hook访问
- 3. 版本库管理网页服务API密钥：点击生成或者手动输入，记录一下后面配置gitlab需要用到
- 4. 允许引用/修复所有其他项目的问题：勾选后就可以在commit message中使用上一项定义的关键字实现git提交和redmine issue的关联，如 refs:#123 表示将本次提交关联到redmine的123号任务

5. 激活时间日志：勾选后可以在commit message中设置当前提交耗时记录，方式为 `Implement feature #1234 @2h` 或者 `Implement feature #1234 @15m`
6. 最下面一个设置框中就是自定义关键字，实现commit message控制redmine中的issue的状态，比如我这里预设了解决、进行中和关闭三个，在commit message中的用法为： `ok:#123` 或 `start #123` 或 `close:#123 @2`，这最后一个实例还把时间也带上了。

2.2. redmine服务器克隆代码版本库

为了redmine能够读取到代码版本库，需要在redmine服务器上克隆gitlab上的项目的代码版本库。同时由于我是单独挂了一块盘在目录 `/git-repo`，同时redmine运行时使用的用户是daemon，所以我需要设置合理的权限。

- 设置代码库所在目录的用户和用户组均为daemon

```
1  
2
```

```
cd /git-repo  
chown daemon:daemon /git-repo
```

- 给daemon用户生成ssh key

```
1  
2  
3
```

```
mkdir /usr/sbin/.ssh  
chown daemon:daemon /usr/sbin/.ssh  
sudo -u daemon -H ssh-keygen -t rsa
```

- 将上一步生成的文件 `/usr/sbin/.ssh/id_rsa.pub` 的内容添加到自己gitlab的SSH Keys中，以便无密码获取代码库
- 克隆代码版本库

2.3. redmine的项目启用版本库

- 在redmine的项中配置中启用版本库这个模块：



- 在 配置-版本库 中新建版本库，配置信息如下：

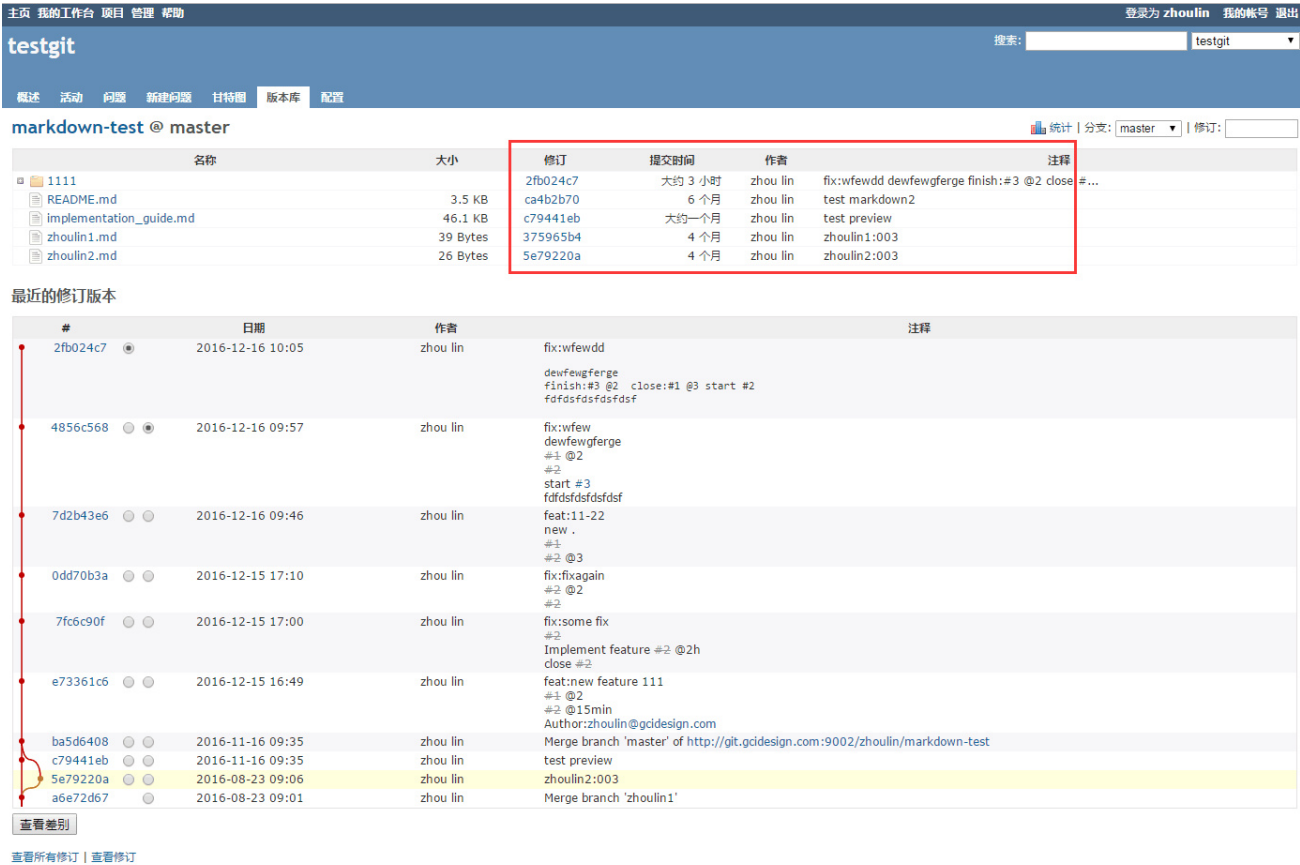


1. SCM: 选择git，在 2.1的设置 中只勾选了git

- 2. 主版本库：如果勾选了这个在项目的导航条上就会新增一个tab—版本库，不勾选，就需要通过 配置—版本库 查看
- 3. 标识：是版本库的一个名字
- 4. 库路径：就是前面执行git clone生成的目录的路径，注意：redmine需要有该目录的读写权限，否则点击版本库会出现404错误
- 5. 路径编码：优选UTF-8
- 6. 报告最后一次文件/目录提交：建议勾选，选上后打开版本库会看到相关文件最后一次提交的相关信息如作者、commit message等，不勾选的效果如下图，勾选的效果见第7点。



7. 设置完成后点击设置好的版本详情如下图：



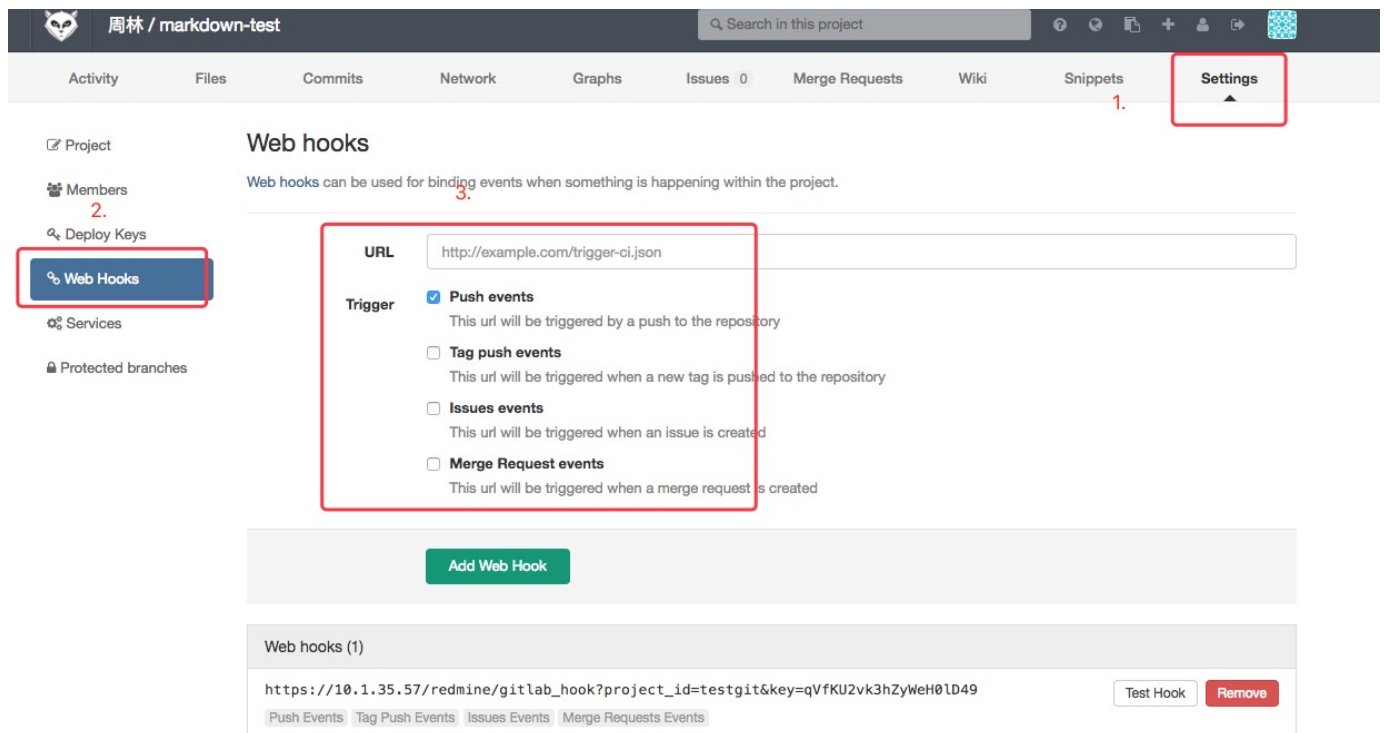
2.4. 利用gitlab的web_hook和redmine的插件实现自动更新

经过前面几步的配置我们已经基本实现了要求里面的功能，但是有一点很关键目前还没有实现，那就是如何实现咋我们提交代码之后redmine本地的代码版本库能够自动同步获取更新以便实时显示在redmine上呢？目前的解决方案有如下几种：

1. 本地利用cron定时获取更新，
2. redmine官方有 Fetch commits automatically，我没有实现
3. 利用redmine的插件和gitlab的web_hook，我觉得这是最好的方式，也是本文介绍的方式

2.4.1.gitlab的web_hook配置

web_hook的配置其实十分简单，只需要知道redmine中的项目名和API密钥，如下图：



1. URL: url连接形如 `https://10.1.35.57/redmine/gitlab_hook?project_id=testgit&key=qVfKU2vk3hZyWeH01D49`
 - `https://10.1.35.57/redmine` :表示当前redmine的url地址，注意：这里的https需要跟你的redmine设置一致，否则容易出现404错误，当然如果你设置了所有http自动转为https，你这里写http也没有问题；其次是这里的redmine，如果你没有更改默认的域名设置，务必加上redmine，否则也是404，[修改根域名方式](#)。
 - project_id: 这个就是redmine中项目的名称
 - key: 这个是前面的 2.1节 设置中的 启用用于版本库管理的Web Service
2. Trigger: 这个很好理解，哪些操作触发这个hook，默认都勾选就好。

2.4.2.redmine_gitlab_hook插件

这个插件出自github，代码地址见: [redmine_gitlab_hook](#)，插件的安装按照Redmine常规插件安装即可，插件配置页面如下：

插件 » Redmine GitLab Hook plugin

Git pull type

All branches ☒ This option fetches all branches from all remotes.

Prune ☐
After fetching, prune removes any remote-tracking references that no longer exist on the remote. Tags are not subject to pruning if they are fetched only because of the default tag auto-following or due to a --tags option. However, if tags are fetched due to an explicit refspec (either on the command line or in the remote configuration, for example if the remote was cloned with the --mirror option), then they are also subject to pruning.

Auto create ☐
If the repository does not exist try to initialize it.

Local repositories path
Path where auto-created repositories should be stored.

Git command prefix
If you need to add some additional parameters before git command use this field. For example if you want to use GIT_SSH parameter.

Fetch updates from repository ☒ This option fetches updates from remote repository. Set it to "No" if repository shared (for example if Gitlab and Redmine on the same machine).

应用

这里我只启用了这两项，说明已经很详细了，我就不赘述了。

三. 错误

在整个配置过程中我碰到了下面几个错误，可能解决途径不是最优或者其实只是瞎碰上的，仅供参考

3.1. 点击redmine版本库出现404

出现这个问题有两个原因：

1. 权限：redmine没有足够权限读写你配置的目录。更换目录或者修改目录权限
2. 版本库：你指定的目录是实际的代码而不是版本库，常规克隆出来的版本库在隐藏文件夹 `.git` 下，或者 clone 是添加 `-mirror` 参数

3.2. 测试gitlab的web_hook时404

这又要分两种情况：

3.2.1. redmine的日志里面没有任何错误信息

在apache的error_log日志中有一条404记录，这个说明你配置的Web_hook的url错了，可能原因：

1. 你启用了https，但是url写的是http，而且没有[设置所有http自动跳转到https](#)
2. 你没有修改默认的根URL指向，默认要访问redmine，地址是：<http://IP-or-domain/redmine>，你的url中忘记了这个redmine，或者请[修改redmine默认根URL](#)。

3.2.3. redmine的日志有错误信息，提示处理index时404

在production.log中的错误日志如下：

1
2
3
4
5
6
7
8

```
Started GET "/gitlab_hook?project_id=storage&key=2CivrlkuhwE012DmaCJ9" for 10.1.41.165 at 2016-12-15 17:46:23 +0800
Processing by GitlabHookController#index as HTML
  Parameters: {"before"=>"8e279aldaed291d059bf626246a0165e7cc83940",
"after"=>"7c6e0d14d4b96fe0f908f326da51629230fd1428", "ref"=>"refs/heads/master", "user_id"=>3,
"user_name"=>"周林", "project_id"=>
"storage", "repository"=>{"name"=>"translations",
"url"=>"git@git.gcidesign.com:zhoulin/translations.git", "description"=>""},
"homepage"=>"http://git.gcidesign.com:9002/zhoulin/translations"}, "commi
ts"=>[{"id"=>"7c6e0d14d4b96fe0f908f326da51629230fd1428", "message"=>"fix:auto fetch2\n\ntest
auto fetch between redmine and gitlab.\nrefs:#1268 @2\nclose:#1268\n\nSigned-off-by: zhoulin
<zhoulin@gcid
esign.com>", "timestamp"=>"2016-12-15T17:47:51+08:00",
"url"=>"http://git.gcidesign.com:9002/zhoulin/translations/commit/7c6e0d14d4b96fe0f908f326da516
29230fd1428", "author"=>{"name"=>"unknown", "emai
l"=>"zhoulin@gcidesign.com"}]], "total_commits_count"=>1, "key"=>"2CivrlkuhwE012DmaCJ9"}
Completed 404 Not Found in 1ms (ActiveRecord: 0.0ms)
```

这个错误也是在仔细阅读插件文档后发现，错误原因是gitlab的web_hook发送过来的请求是GET，而不是插件要求的POST。这个问题我也是偶然发现的，因为我部署了两个redmine测试，同一个gitlab，其中一个redmine就正确的，一直是POST方式。还没有搞明白为啥有个时候是GET，有个时候是POST，我的解决方法是修改插件代码，去掉判断request的类型，修改如下：

1
2
3
4
5
6
7
8
9
10
11
12
13
14

15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34

```
## 修改文件 redmine_gitlab_hook/app/controllers/gitlab_hook_controller.rb
## def index 函数修改如下： 注释掉4行
def index
  # if request.post?
  repositories = find_repositories
  if repositories.empty?
    render(:text => 'No repository configured!', :status => :not_found)
  else
    errors = []
    repositories.each do |repository|
      # Fetch the changes from GitLab
      if Setting.plugin_redmine_gitlab_hook['fetch_updates'] == 'yes'
        git_success = update_repository(repository)
        unless git_success
          errors.push(repository.identifier)
        else
          # Fetch the new changesets into Redmine
          repository.fetch_changesets
        end
      else
        # Fetch the new changesets into Redmine
        repository.fetch_changesets
      end
    end
  end
  if errors.empty?
    render(:text => 'OK', :status => :ok)
  end
end
```



```
        else
          render(:text => "Git command failed on repository: #{errors.join(', ')}!", :status =>
:not_acceptable)
        end
      end
    # else
    #   raise ActionController::RoutingError.new(' Not Found')
    # end
  end
end
```