Косяки с установкой redmine_git_hosting в Debian Jessie

Shroom [OmskLUG]*

29 сентября 2015 г.†

В статье, как обычно, собраны все подводные камни, грабли, баги и прочий трэш, который успел собрать автор за время установки и настройки плагина `redmine_git_hosting`. Естественно, даны способы обхода и устранения, а также рекомендации по предупреждению всех этих весёлых глюков.

Преамбула

До меня время от времени доходили слухи, что установка и сопровождение Redmine — это не то, чтобы адЪ, но как минимум ~~геморрой~~ неслабые головняки. Вероятно, я подогнался под эту тему и в первый раз установка заняла у меня дня три. Во второй раз я уложился в пару дней. В последнее время я делаю успехи, поскольку установка плагина redmine_git_hosting прошла не более, чем за сутки. И это не может не радовать. Последовательность шагов по установке очень подробно описана в руководстве от разрабов, которое можно найти здесь: Get started. Собственно, именно такого порядка действий я и буду придерживаться в статье. Вследствие этого изложенный здесь материал может показаться фрагментарным и несколько несвязным. И если вам так и кажется, вы, пожалуй, правы. . .

Да, кстати. По этому тексту раскидана туева хуча полезных ссылок. Посему настоятельно рекомендую хотя бы мельком заглядывать в каждую из них по мере прочтения, иначе удовольствие будет казаться неполным.

†Дата обновления: 2 октября 2015 г.

Начало: клонирование и сборка

На всякий пожарный инструкция предупреждает нас о необходимости сбэкапить базу и остановить Redmine. Благоразумно воспользуемся этим советом и приступим...

Как ни странно, в самом начале процесс идёт на удивление гладко. Шаги по установке зависимостей и клонированию реп проходят без проблем. :)

Однако, как только дело доходит до `bundle install`, он тут же начинает орать. Во-первых, выдаёт предупреждение о дубликате зависимости «`gem 'redcarpet', '~> 3.1.2'`». Это не страшно, убирается комментированием соответствующей строчки в `Gemfile` внутри каталога плагина. Во-вторых, ошибка, связанная с несоответствием версий `rake`. Дескать, установлена 1.5.2, а плагин через зависимость от `gitlab-grack` требует аж 1.6.0. Это вызвано тем, что в репе jbox-web не указана версия `rake`; соответственно, `bundler` вытягивает и собирает самую последнюю. Собственно, сам автор плагина предлагает бороться с этим косяком, используя гитлабовский форк. В результате получаем такой кусок `Gemfile`[1]:

```
## Redmine 3.x
## Ruby/Rack Git Smart-HTTP Server Handler (use our own repository\
because Redmine uses Rails 4.2 and Rack 1.6)
#gem 'gitlab-grack', git: 'https://github.com/jbox-web/grack.git', \
require: 'grack', branch: 'fix_rails4'
#gem 'redcarpet', '~> 3.1.2'

# fix for Debian Jessie with rails 4.1
# (https://github.com/jbox-web/redmine_git_hosting/issues/\
497#issuecomment-133768646)
gem 'gitlab-grack', git: 'https://github.com/gitlabhq/grack.git', \
require: 'grack', branch: 'master'
```

После этих манипуляций все гемы собираются, как надо.

Следующий шаг — *migrate*

Здесь сразу же вылетает ошибка «*The git source https: github.com/jbox-web/gitolite-rugged.git is not yet checked out. Please run bundle install before trying to start your application*». Естественно, `bundle install` не помогает. Решение: закомментировать ссылки на git и поставить просто зависимость, как от обычных гемов. В результате получим что-то подобное:

```
# Gitolite Admin repository management
#gem 'gitolite-rugged', git: 'https://github.com/jbox-web/\
```

[1] Здесь и далее в листингах все слишком длинные строки разбиты на части с помощью «\». Это сделано исключительно ради удобства чтения в формате PDF.

```
gitolite-rugged.git', tag: '1.1.1'
gem 'gitolite-rugged'

#gem 'gitlab-grack', git: 'https://github.com/gitlabhq/grack.git', \
require: 'grack', branch: 'master'
gem 'gitlab-grack', require: 'grack'
```

Кстати, в Интернете можно найти довольно много вопросов, связанных именно с таким глюком, когда какой-нибудь гем ставится из гита. ИЧСХ, внятных ответов, как бороться с этой напастью, никто не даёт. В основном какие-то шаманские рецепты типа придуманного мной. Это намекает на пару возможных причин: или эта ситуация настолько тривиальна, что отвечать просто нет смысла, или же действительно никто не знает, что там на самом деле происходит в недрах RoR вообще и Redmine в частности.

Ну так вот. После этого нужно ещё раз запустить `bundle install`, после чего migrate будет ругаться на другие вещи. Или, если redmine изначально был поставлен по-человечески, всё пройдёт гладко. В моём случае пользователю БД не хватало прав на создание и удаление таблиц. Выполнение в mysql

```
grant all privileges on redmine_db.* to 'redmine_user'@'localhost';
flush privileges;
```

решило проблему радикально.

Установка и настройка Gitolite

Пользователи и ключи

Поскольку Gitolite работает от одного пользователя через ssh, нам нужно сгенерировать некоторое количество ключей. Этот процесс вызвал единственный вопрос относительно места их хранения. В инструкции для создания пары административных ключей для плагина рекомендовано создать каталог `ssh_keys` в `REDMINE_ROOT`. Однако, если внимательно посмотреть на `init.rb`, становится понятно, что в дереве плагина уже есть `ssh_keys`, и создан он там отнюдь не случайно. Плагин в первую очередь именно там и ищет ключи. Впрочем, когда всё заработает, каталог с ключами можно переопределить в настройках.

Во время установки Gitolite нужно будет указать пользователя, из-под которого он будет работать (и в которого будут логиниться все имеющие доступ к репозиториям). Для общности пусть это будет юзер *git*. В качестве административного ключа нужно указывать *открытый* ключ того человека, который будет админом (ну или сразу ключ, сгенерированный для `redmine`). Дополнительные ключи (для `redmine`, в частности) кладутся в `GITOLITE_HOME/.gitolite/conf/gitolite.conf`. После этого нужно сделать

`gitolite compile` от пользователя *git*. **Но на самом деле** ничего такого делать не нужно. Gitolite управляется через git-репу. Поэтому после установки с добавлением одного административного ключа можно и даже нужно просто склонировать репозиторий `gitolite-admin` тому пользователю, которого сделали git-админом. После этого можно настраивать конфиги как угодно, не забывая заливать коммиты обратно. Как и что делать, подробно разжёвано в соответствующем руководстве.

Дополнительно (для версии Gitolite 3) «патчится» пара скриптов, но это описывать не имеет смысла, поскольку процесс тривиален.

Если что-то пошло не так...

Но вот если забыть, как работает `ssh`, возникнут некоторые сложности с доступом по ключам. Во-первых, домашний каталог Gitolite должен принадлежать руту или тому юзеру, из-под которого работает Gitolite (и мы в самом начале решили, что это *git*). Следствие: если с репозиториями работает ещё кто-то (например, `gitdaemon`), этого кого-то нужно добавить в группу *git*. С другой стороны, что касается `gitdaemon`'а, то его-то как раз можно запустить от имени любого нужного пользователя. Во-вторых, нужно выставить адекватные права на `.ssh` и то, что в нём. В-третьих, Gitolite различает git-юзеров по их ключам, поддерживая соответствующую структуру файла `authorized_keys`: перед ключом должна идти строчка

```
command="/usr/share/gitolite3/gitolite-shell <git_user_name>",\
no-port-forwarding,no-X11-forwarding,no-agent-forwarding,no-pty
```

Если выполнены все эти условия, с доступом проблем не будет. В крайнем случае поможет анализ `/var/log/auth.log`. Вообще на сайте Gitolite есть специальная страница, где собраны все возможные косяки с `ssh` и способы их устранения. Опять же, если не лазить в конфиги руками, а делать всё через `git`, как расписано в документации, всё должно работать адекватно.

Что же там всё-таки с именами?

Кстати, есть ещё один нюанс, связанный с тем, как Gitolite определяет имена пользователей. Всё очень просто: как обзовёшь открытый ключ, так тебя и будут звать дальше. Опять же, всё не настолько фатально, если вспомнить, что вся конфигурация хранится в git-репозитории. Однако же, вот такая штука. Руководство по установке плагина предлагает назвать сгенерированный ключ каким-то нечеловечески длинным именем с суффиксом rsa_id. И это означает, что административный пользователь redmine будет назван этим самым непроизносимым именем. В сущности, и redmine, и плагину, и Gitolite — всем пофиг, формально всё будет работать. Однако, мне лично удобнее, когда подконтрольные мне сущности называются просто и понятно. Поэтому я переобозвал ключи в `redmine-admin` и `redmine-admin.pub`. И теперь в конфиге всё прозрачно и тихо.

Запуск Redmine и настройка плагина через веб-морду

Наконец-то, наконец-то! Если вы дошли до этого торжественного момента с минимальными потерями, вас можно поздравить. Остальных, наверное, тоже можно. Просто им уже не до поздравлений.

В общем, выполнение заключительной части инструкции особых сложностей не вызывает. Тем более, что есть целая страница с описанием тех параметров, которые нам предлагают настроить. Более того, там уже практически ничего не поломаешь, и можно, если что, исследовать систему методом научного тыка. С минимальным знанием английского или французского языков (других локализаций в этом плагине я не заметил) вполне реально разобраться даже без документации. Однако, в документации обычно пишут много интересного и полезного. :) Так что хотя бы краем глаза взглянуть всё же сто́ит.

Реальные баги

Так или иначе все ошибки, связанные с софтом, суть ошибки в коде. И если во время установки и настройки это относится к генетическому коду того, кто настраивает, то во время работы самой программы уже ничего не попишешь: мы сталкиваемся с реальными багами.

Вывод ReadMe

Итак, первый косяк в коде, с которым я столкнулся в процессе эксплуатации плагина, находится в файле `REDMINE_HOME/plugins/redmine_git_hosting/lib/redmine_git_hosting/hooks/display_repository_readme.rb`. Не буду рассказывать, как я его отлавливал и локализовал в конце концов, поскольку дебаггинг — это тема для отдельной статьи. Лучше сразу покажу, что нужно менять:

```
diff old/display_repository_readme.rb new/display_repository_readme.rb
58c58
<   raw_readme_text = repository.cat(readme_file.path, rev)
---
>   raw_readme_text = Redmine::CodesetUtil.to_utf8_by_setting(\
      repository.cat(readme_file.path, rev))
```

А теперь поясню, ради чего, собственно, менять. Этот модуль должен выводить ReadMe под списком файлов и логом коммитов. Практически как на гитхабе. Однако же, без принудительного перевода прочитанного буфера в UTF-8 этот модуль падает на символах с кодами, выходящими за пределы таблицы ASCII. А внешне это выражается в том, что если в репе есть ReadMe с какими-то нестандартными символами (типа диакритики или расширенной пунктуации), то вместо репы мы увидим ~~шиш с маслом~~

Server Error 500. Это некрасиво да и вообще как-то неправильно. Поэтому вот такой нехитрый патч. Сделал pull request с этим «патчем»; будем ждать, когда окажется в апстриме[2].

Дополнительно нужно отметить, что для отображения ReadMe в формате ReStructuredText требуется установка `python-docutils`. Да, именно. Это питоновский пакет, который нужен программе на ruby. Так тоже бывает.

Вывод новостей

В процессе работы был выявлен ещё один косяк, характерный именно для этой версии redmine. В случае, когда у пользователя выставлен признак «Отображать комментарии в обратном хронологическом порядке», при попытке просмотреть отдельную новость (вне зависимости от проекта и наличия комментариев) сервер выдавал Error 500. После анализа логов, мониторинга базы и изучения исходников удалось обнаружить ошибку в `REDMINE_HOME/app/controllers/news_controller.rb`. Косяк выпрямляется вот таким вот незатейливым способом:

```
diff old/news_controller.rb new/news_controller.rb
61c61
<     @comments = @news.comments
---
>     @comments = @news.comments.to_a
```

Кстати, как оказалось впоследствии, в транке redmine этот баг был пофиксен в ревизии 13595 абсолютно так же.

Автоматическая подгрузка изменений

Собственно, это не то, чтобы баг... Скорее глюк. В общем, я заметил, что репы, переведённые в Redmine на Gitolite, не подхватывали автоматом изменения, сделанные снаружи, хотя соответствующая настройка в конфигурации была включена. В результате оперативно был воплощён в жизнь такой рецепт.

Чтобы redmine могла автоматически отслеживать изменения в репозиториях, в `GITOLITE_HOME/.gitolite/hooks/common/post-receive.d` нужно добавить скрипт, вызывающий метод *fetch_changeset* либо через WebAPI, либо напрямую через запуск redmine. Например, такой:

```
#!/bin/bash

API_KEY=XXXXXXXXXXXXXXXXX

PROJECT_ID=`git config redminegitolite.projectid || echo ''`
```

[2] **Upd[2015-10-02]:** патч принят в основной репозиторий

```
[ "${PROJECT_ID}" ] && {
    wget "https://<redmine.server.name>/sys/fetch_changesets?\
        id=${PROJECT_ID}&key=${API_KEY}" >&/dev/null
    echo Redmine repository updated successfully
}

exit 0
```

Естественно, для доступа через веб должна быть включена соответствующая опция в настройках и получен ключик. Это делается в пару кликов на странице «Administration → Settings» на вкладке «Repositories».

Собственно, на этом всё.

Спасибо за внимание!