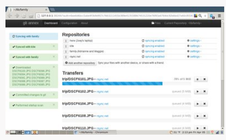
Покрекламирую-ка git-annex

15 December 2013 • git, linux

Вчера я уже упоминал один из проектов Joey Hess — etckeeper. Сегодня же хочу обратить ваше внимание на другую его разработку, которая называется git-annex. В рунете почему-то имеется всего несколько упоминаний о нём, в основном с тех времён, когда разработчик собирал деньги на Kickstarter. Есть также пара развёрнутых обсуждений и даже ряд идивидуальных отзывов, но этого, как по мне, все равно слишком мало. Надеюсь, этот пост позволит новым пользователям открыть для себя эту замечательную программу.

Перед тем, как отправиться в путь, последнее замечание: я предпочитаю консольные интерфейсы, и говорить буду именно о них. У git-annex есть веб-интерфейс и я уверен, что через него можно удобно управлять вашим репозиторием, но я этим средством почти не пользовался и потому не знаю, насколько оно совершенно. Если у вас острая нетерпимость CLI, пожалуйста, не читайте дальше.



Итак, предлог следующий. У многих из нас есть коллекции фотографий, музыки, научных статей, видео с конференций — кучи немаленьких файлов, которые хотелось бы использовать на нескольких машинах сразу, или же удобно архивировать на DVD (или что вы там используете). Эти (как и некоторые другие) задачи и призван решать git-annex.

Технически говоря, git-annex — это надстройка над системой контроля версий Git, но пусть вас это не пугает — с собственно Git'ом вам общаться не придётся, если только вы сами этого не захотите. Git-annex добавляет Git'у новые команды, с помощью которых вы можете инициализировать и управлять своим annex'ом.

Кстати говоря, annex может быть добавлен к любому git-репозиторию, но об этом я здесь говорить не буду — если интересно, почитайте сайт проекта.

Мощь и красоту этой утилиты лучше показывать, а не описывать, так что перейдём к делу. Joey — разработчик Debian, так что не странно, что в нашем с вами любимом дистрибутиве для git-annex есть готовенький пакет:

```
$ sudo aptitude install git-annex
```

Я доверяю git-annex все свои мультимедийные данные, и пока что он меня не подводил. Тем не менее, я не советую вам вот так сразу помещать под его управление единственную копию ваших фотографий, публикаций или что вы там собрались в нём хранить — сделайте копию, поэкспериментируйте на ней, а как свыкнитесь с командами, пробуйте уже на «живых» данных.

Перейдём-ка в какую-нибудь директорию с файлами и попробуем создать в ней annex:

```
$ cd my-first-annex
$ ls -R
.:
avatar.jpg  avatar-myopenid.png  summer_camp_2012  wallpaper.png

./summer_camp_2012:
DSC05599.jpg  DSC05602.jpg  DSC05604.jpg  DSC05605.jpg
$ git init .
Initialized empty Git repository in /tmp/my-first-annex/.git/
$ git annex init 'notebook'
init notebook ok
(Recording state in git...)
```

Как видите, сперва мы создали git-репозиторий, а затем уже инициализировали в нём annex. Обязательным параметром `git annex init` является название annex — зачем он нужен, вы узнаете чуть позже.

Добавим и закоммитим файлы:

```
$ git annex add .
add avatar-myopenid.png (checksum...) ok
add avatar.jpg (checksum...) ok
add summer_camp_2012/DSC05599.jpg (checksum...) ok
add summer_camp_2012/DSC05602.jpg (checksum...) ok
add summer_camp_2012/DSC05604.jpg (checksum...) ok
add summer_camp_2012/DSC05605.jpg (checksum...) ok
add wallpaper.png (checksum...) ok
(Recording state in git...)
$ git commit -m"Добавил аватарки и пару летних фоток"
[master (root-commit) d312aca] Добавил аватарки и пару летних фоток
 7 files changed, 7 insertions(+)
 create mode 120000 avatar-myopenid.png
 create mode 120000 avatar.jpg
 create mode 120000 summer_camp_2012/DSC05599.jpg
 create mode 120000 summer_camp_2012/DSC05602.jpg
 create mode 120000 summer_camp_2012/DSC05604.jpg
 create mode 120000 summer_camp_2012/DSC05605.jpg
 create mode 120000 wallpaper.png
$ ls -l avatar.jpg
lrwxrwxrwx 1 minoru minoru 192 Dec 14 06:38 avatar.jpg ->
.git/annex/objects/7v/Q7/SHA256E-s7164--b442aaed464409198c19f37614dd6e7fd82ee658b897deaddf9a51b08c1aa19f.jpg/SHA256E-s7164--b442aaed464409198c19f37614dd6e7fd82ee658b897deaddf9a51b08
```

Ой, что случилось? Не переживайте, всё именно так, как должно быть. В отличие от git, который не трогает ваши файлы, а только запоминает их состояние, git-annex переместил все наши фотографии в глубины `.git/annex/objects`, а в рабочей директории расставил симлинки на них. Должен предупредить, что симлинки могут негативно сказаться на скорости работы `ls -l`, `ls --color` и прочих штук, которые читают содержимое симлинка с целью выяснить, куда он ведёт — на моём ноутбуке с 5400rpm HDD `ls -l --color` в директории с почти тремя сотнями файлов отрабатывает около секунды. Впрочем, при последующих вызовах кеширование сглаживает это время до нуля, так что ничего страшного я в этой ситуации не вижу.

Ну ладно, коммитить файлы можно было и в простом Git — какие же именно преимущества даёт нам git-annex? Давайте-ка попробуем клонировать наш git-репозиторий куда-нибудь ещё, например, на внешний винчестер:

```
$ cd /media/storejet
$ git clone /tmp/my-first-annex my-first-annex
```

Инициализируем здесь annex под названием "storejet" и сообщим первому репозиторию о втором:

```
$ cd /media/storejet/my-first-annex
$ git annex init 'storejet'
$ git annex sync
(merging origin/git-annex into git-annex...)
(Recording state in git...)
commit
ok
pull origin
ok
push origin
Counting objects: 11, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (6/6), done.
Writing objects: 100% (8/8), 722 bytes | 0 bytes/s, done.
Total 8 (delta 3), reused 0 (delta 0)
To /tmp/my-first-annex
 * [new branch]      git-annex -> synced/git-annex
 * [new branch]      master -> synced/master
ok
$ # идём обратно в первый репозиторий
$ cd /tmp/my-first-annex
$ git remote add storejet /media/storejet/my-first-annex
$ git annex sync
commit
ok
pull storejet
From /media/storejet/my-first-annex
 * [new branch]      git-annex  -> storejet/git-annex
 * [new branch]      master     -> storejet/master
 * [new branch]      synced/git-annex -> storejet/synced/git-annex
 * [new branch]      synced/master -> storejet/synced/master
ok
pull origin
From /media/storejet/my-first-annex
   b10e6d9..6706944  git-annex  -> origin/git-annex
ok
```

Вот эта магия с `git remote add` — это такой способ объяснить репозиториям, как они могут достучаться до остальных (второй уже знает о первом, потому что был с него клонирован). Возможно, в будущем git-annex начнёт предоставлять какие-то команды, которые эту магию от пользователя спрячут, но пока что работаем так.

Посмотрим на наш свежесозданный клон:

```
$ ls -l avatar.jpg
lrwxrwxrwx 1 minoru minoru 192 Dec 14 06:59 avatar.jpg ->
.git/annex/objects/7v/Q7/SHA256E-s7164--b442aaed464409198c19f37614dd6e7fd82ee658b897deaddf9a51b08c1aa19f.jpg/SHA256E-s7164--b442aaed464409198c19f37614dd6e7fd82ee658b897deaddf9a51b08
$ test -e avatar.jpg && echo "Link is ok" || echo "Broken link"
Broken link
```

Упс, а линк-то битый! (Как и все остальные в репозитории, кстати говоря). Это, господа, вовсе не баг, это фича. Напомню, что файлы хранятся не в в самом git-репозитории, а в `.git/annex/objects`, поэтому `git clone` их не копирует. Это даёт вам возможность создавать так называемые partial checkouts — клоны репозитория, в которых присутствуют не все файлы. В то же время, в каждом клоне есть полный список симлинков на все существующие файлы, и их можно легко получить, что мы сейчас и сделаем:

```
$ git annex get avatar.jpg
get avatar.jpg (from origin...) ok
SHA256E-s7164--b442aaed464409198c19f37614dd6e7fd82ee658b897deaddf9a51b08c1aa19f.jpg
      7,164 100%    1.01MB/s    0:00:00 (xfr#1, to-chk=0/1)
ok
(Recording state in git...)
$ test -e avatar.jpg && echo "Link is ok" || echo "Broken link"
Link is ok
```

Что сейчас произошло? git-annex сообразил, что файла в текущем клоне нет, но он есть в origin (это тот репозиторий, с которого был клонирован данный, то есть наш "notebook"), после чего файл был скопирован в текущий клон. Как видно из вывода, для копирования был использован rsync, что как бы намекает, что передача файлов неплохо оптимизирована (передаётся только разница, есть докачка).

Вы всегда можете выяснить, где именно находится тот или иной файл:

```
$ git annex whereis avatar.jpg
whereis avatar.jpg (2 copies)
    31ef94bc-12d5-4af4-96a2-fb73bac12039 -- here (storejet)
    8644f457-86ec-4b15-bdc5-022ded68d4fa -- origin (notebook)
ok
```

"storejet" и "notebook" взяты не с потолка — это те самые имена, которые мы передавали параметром в `git annex init`. Как видите, они нужны для того, чтобы вы могли легко понять, какой именно репозиторий скрывается за непонятным численно-буквенным именем.

С помощью команд `get`, `copy` и `move` можно удобно рулить файлами (вывод опущен):

```
# получить все файлы в этот репозиторий
$ git annex get .
# скопировать директорию с фотографиями обратно на notebook
$ git annex copy --to=notebook summer_camp_2012
# переместить всё со старого ноутбука сюда
$ git annex move --from=old_notebook .
```

Уже существующие файлы не передаются, то есть вторая команда не передаст ничего — фотографии с “notebook” никуда не девались, так что и копировать ничего не нужно.

Последняя команда git-annex, без которой вам не удастся жить — это `git annex sync`. Я уже запускал её, когда говорил о клонировании репозитория, но не объяснял, что она делает. А выполняет она очень важную операцию — синхронизирует текущий annex со всеми остальными, подтягивая из них данные о новых файлах и отправляя информацию о том, что поменялось в текущем. Для полной синхронизации, насколько я понимаю, нужно два круга (то есть по очереди выполнить `git annex sync` во всех репозиториях, а потом сделать то же самое ещё раз), но на деле вам вряд ли понадобится быть насколько строгим и последовательным. Лично мне с с ноутбуком, внешним винчестером и нетбуком хватает синхронизации между ноутом и винтом по крону, плюс ручная синхронизация нетбука, выполняемая время от времени по настроению. Git гарантирует, что ничего не потеряется, просто если добавить или удалить какой-то файл, информация об этом может не сразу добраться до всех репозиториев.

Ну всё, базовые функции я показал, свой первый annex мы создали — думаю, вы готовы читать git-annex walkthrough и идти внедрять git-annex в свою повседневную жизнь. Впереди вас ждёт куча интересных фич, таких как special remotes (возможность использовать Amazon Glassier, Flickr и прочие в качестве удалённых репозиториев), unused content (удаление или перемещение в архив старых версий файлов) и многое другое. Удачи!

Drop me a line! (wonder where’s the comments form?)