Использование GIT .

printable page

29 August, 2008 - 19:31 — alfss

Повесть о git .
О том как его оседлать и с чем его есть.

Данный чудо-агрегат является децентрализованной системой управления версиями.
Что обеспечивает возможность работы не имея доступа к серверу или, если произойдет крах сервера, не к таким фатальным последствиям.

Думаю, стоит рассмотреть на примере задачи и отталкиваться от этого.

Предположим, у нас есть 2-3 человека, работающие над одним проектом или документацией.
Задача :
Сделать работу по изменению и поддержанию более легкой.
Обеспечить возможность работы над проектом всем одновременно.
Вести историю изменений.

Для начала поставим сам git.
Он есть в портах (полагаю и в пакетах).

После, выберем место, где будем хранить репозитории.

Создадим пользователя, к примеру, git с домашней директорией, где будут репозитории.
home :/var/project/

Вселимся в его душу)) :

```
su git
cd ~
```

И создадим наш репозитории.

```
git-init --shared=group
```

После проинициализируем его.

```
touch start
git-add start #добавляем файл в репозитории
git-commit -m 'start rep'#оставляем запись о изменениях
```

На данный момент в нашем репозитории находится ветка master в которой есть 1 коммит и файл start.

Создадим пользователей, которые будут работать с этим репозиторием и дадим им группу git,
поскольку он был проинициализирован с возможностью доступа группы владельца репозитория **git-init --shared=group**.

После открываем на локальной машине консоль(я полагаю что у вас уже стоит git на локальной машине) и делаем начальную копию репозитория.

```
git-clone ssh:/server/var/project ~/project
```

Создаем файл **.gitconfig**
и делаем там записи о том кто вы такой и где вас найти)

```
[user]
name = Your Name Comes Here
email = you@yourdomain.example.com
```

переходим в **~/project**

И видим ,что там находится файл start
если сделать ls -a то мы увидим сам репозиторий ".git".

Собственно теперь можно спокойно начинать работу)).

Для начала стоит сделать новую ветку, в которой вы будете работать и не затрагивать ветку master вплоть до того момента, пока не решите что данные изменения должны попасть в master ветку.

```
git-branch my_new_branch
```

и сменим текущую ветку

```
git-checkout my_new_branch
```

Дальше спокойно делаем свои изменения, не забывая добавлять в локальный репозиторий.

Скажем мы создали файлы test1, test2.

Они находятся локально, но они не в репозитории.
Чтобы добавить их в репозиторий используем **git-add**.

```
git-add test1 test2
```

Данная команда дает указания какие файлы нужно добавить в репозиторий или какие файлы изменились.
Состояние того, что будет добавлено можно посмотреть с помощью **git-status**.

После, чтобы изменения вступили в силу, нужно оставить коммит или, проще говоря, краткую информацию об изменениях.

git-commit -m "add files test1 test2"

После можно просмотреть лог изменений в данной ветке **git log**.
Также можно посмотреть дерево всех изменений с помощью **git-gui,gitk**.

Предположим, что вы закончили работу над изменениями и хотите опубликовать для всех.
Для этого переходим в **master** ветку.

```
git-checkout master
```

Синхронизируем с удаленной машиной ветку мастер **git-pull**.
Это делается на тот случай, если кто-то за это время внес изменения в основной ветке и опубликовал их.

После, объединяем наши ветки

```
git merge my_new_branch
```

Если возникнут какие либо конфликты, то вас проинформируют об этом и их придется исправлять.
После того, как исправите конфликты, их можно посмотреть выполнив **git diff**.

```
git-add "файл в котором конфликт"
git commit
```

После опять делаем **git-pull**
И пробуем занести данные на сервер **git-push**

Если опять конфликты, то исправляем конфликтные файлы, синхронизируя с удаленным сервером и опять пытаемся добавить изменения.

Если вам так и не удалось слить ветви локально или вы отказались от изменений, то можно отменить изменения вернувшись на исходные.

```
git reset --hard HEAD
```

HEAD-вершина текущей ветки.

git show показывает последние коммиты.

Для того, чтобы зафиксировать каким либо символическим именем состояние в ветке в определенной точке времени, можно сделать tag.

Текущему коммиту можно присвоить тег таким образом
git-tag v0.1
Или указать на определённый коммит
git-tag v0.1 568c3c02a608aa759cc9b839d1a47a81c8a05d94(хэш коммита)
(данный тег добавляется в текущей ветке)

git-tag -l посмотреть все теги
После можно обращаться к этому тегу с помощью команд git-checkout или git reset, если понадобится.

Хэш коммита можно посмотреть или в **gitk** или в **git-log**
git branch показать ветки
git gc выполняет сжатие репозитория))- не забывайте выполнять)
git fsck проверка от порчи(сглаза))))

Собственно и вся премудрость - остальные тонкости можно узнать в **man git** или на сайте git.