Лабораторная работа №4

# Изучение системы контроля версий Git. Особые функции.

## Краткие теоретические сведения.

## Фундаментальные основы Git.

Для начала рассмотрим те вещи, которые вам нужно знать про git для его изучения на уровне выше начального:

* Git не хранит папки. Он хранит только файлы. Поэтому, например, невозможно положить в git пустую папку – можно только папку с файлами. Таким образом, git является не системой контроля версий каталогов, а системой контроля версий файлов, однако:
* Git не хранит файлы. На самом деле, он хранит объекты, в которых указаны пути к файлам. Так что в git-репозитории действительно есть файлы, но вот сам git (т.е. папка .git, как мы узнали из предыдущей лабораторной) их не хранит.
* Другими словами, git сохраняет в commit содержимое всех файлов (делает слепки содержимого каждого файла и сохраняет в objects). Если файл не менялся, то будет использован старый object. Таким образом, в commit в виде новых объектов попадут только **изменённые файлы**, что позволит хорошо экономить место на диске и даст возможность быстро переключиться на любой commit.
* Ревизии (Revision) не имеют порядкового номера. Разные файлы входят в разные ревизии, поэтому в каждый момент времени каждый файл принадлежит своему времени.
* Редакции (правки, commits) могут идти не по порядку.
* В Git’е, по сути дела, нет веток. *На самом деле,* у нас есть много коммитов, которые образуют граф. Выбираем любой путь от parent-commit к любому child-commit и получаем состояние проекта на этот коммит. Чтобы коммит «запомнить» можно создать на него именованный указатель. Такой именованный указатель и есть ветка (branch). `HEAD` работает по такому же принципу — показывает, где мы есть сейчас. Новые коммиты являются продолжением текущей ветки (туда же куда и смотрит HEAD).

Git – достаточно сложная система. Чтобы облегчить работу программиста, в поставке git есть две вещи, которые должны быть у вас под рукой всегда:

* git status — показывает состояние вашего репозитория (рабочей копии) и где вы находитесь.
* gitk— графическая утилита, которая показывает наш граф. В качестве ключей передаём имена веток или --all, чтобы показать все.

Если вы сделали что-то не так, запутались, не знаете, что происходит — эти две команды вам помогут.

## Расширенный список функций Git

git annotate, выводит измененные строки и информацию о коммитах, где это произошло: git annotate readme.txt

git diff, а именно:

git diff HEAD~ – показывает разница между текущим и предыдущим коммитом;

git diff 0da94be 59ff30c – показывает изменения между двумя коммитами.

git diff --name-only SHA1 SHA2 – показывает только имена файлов, которые изменились.

git diff --name-only HEAD~10 HEAD~5 – показывает только имена файлов, которые изменились, при этом, показывает только те файлы, которые изменились на расстоянии от 10 до 5 коммита до верхушки ветки. Т.е., если было 20 изменений, то это команда покажет изменения, которые были между десятым и пятнадцатым коммитом.

git show – покажет вам информацию о текущем коммите: как, когда, зачем (сообщение) он был сделан.

git log — просмотр логов, показывает дельту (разницу/diff), привнесенную каждым коммитом.

Для вывода информации git log использует просмотрщик, указанный в конфиге репозитория.

git log --pretty=oneline ­ – вывод данных о каждом коммите в одну строку.

git log | grep -e "first" ­– Поиск по ключевому слову в комментариях к коммиту:

git cherry-pick помогает применить один-единственный коммит из одной ветки к дереву другой. Для этого нужно выписать ветку, в которую будем вливать коммит:

git checkout master

Обновить ее: git pull origin master

Выполнить команду, указать код коммита: git cherry-pick eb042098a5

После этого обновить ветку на сервере: git push origin master

## 1.3 Откат изменений в Git:

Копируем идентификатор коммита, до которого происходит откат.

Откатываемся до последнего успешного коммита (указываем последний коммит):

git reset --hard 9a452d955bdb57e7e4f2b09 (на самом деле хватит и первых 5-6 символов)

Ожидаемый результат: HEAD is now at 9a45779 first commit

Можно откатить до последней версии ветки:

git reset --hard origin/dev

Ожидаемый результат: HEAD is now at 9a45779 first commit

После того, как откат сделан, и выполнен очередной локальный коммит, при попытке сделать push в удаленный репозиторий, git может начать ругаться, что версия вашей ветки младше чем на github и вам надо сделать pull. Это можно исправить принудительным коммитом:

git push -f origin master

Обратите внимание, насколько сложно это выполнить. На самом деле, так лучше не делать, так как если вы закомитили изменения на сервер и их кто-то забрал, то вышеуказанная операция приведёт к полной неразберихе в команде и попыткам исправить то, что вы натворили (люди работали с тем, что вы им дали, а потом вы без предупреждения это забрали). Гораздо лучше использовать

git revert — отмена изменений, произведенных в прошлом отдельным коммитом. Данная команда просто создаст ещё один коммит поверх предыдущего, отменив его. Таким образом, вы получите два коммита: один тот, который вы хотите убрать, и второй тот, которым вы убираете то, что вы хотите убрать ☺

За более подробной информацией обращайтесь в git help.

# 2. Задание.

## 2.1 Порядок выполнения работы

1. Пройти игру; ссылку получить у преподавателя (или <https://github.com/demns/git-game.git> ). Прохождение заключается в поиске решения при помощи командной строки; в отчёте нужно будет указать все этапы, т.е. какие функции вы использовали для прохождения.
2. Открыть gitk и посмотреть на игру с другой стороны; уметь объяснить преподавателю, что где происходит.
3. \* попробуйте выполнить следующие команды и посмотреть на результат:
   1. cherry-pick
   2. git for-each-ref --sort=-committerdate refs/heads/
   3. git log --graph --abbrev-commit --decorate --date=relative --all

# 3. Дополнительная информация:

P.S. выход из VIM -- :q ☺

<http://habrahabr.ru/post/174467/>