GITHub начало

Репозиторий

Итак, вот у вас уже есть git. Теперь нужно создать хранилище версий для него. Запомните, это хранилище называется **репозиторий** *(англ. repository)* — при случае можете вставить где-нибудь это словечко. В зависимости от того, какая у вас оболочка, соответствующей командой *создайте новую директорию*, *откройте ее* (в командной строке, она же оболочка, а не проводником или чем-то подобным) и *выполните:*

git init

Все, локальный репозиторий в этой папке создан. То, что здесь сейчас хранится, будет бекапом, поэтому, чтобы его не испортить, создадим рабочую копию *(англ. check out)* локальной версии:

git clone [url]

Где [url] — это путь до клонируемого репозитория. Мы разбираем сейчас случай, когда вы создаете рабочую копию собственного репозитория, поэтому в качестве [url] здесь вам нужно указать путь до директории, для которой мы выполняли git init.

Но если вы крутой чувак и уже работаете с удаленным сервером, то вот такая команда будет для вас в самый раз:

git clone username@host:/path/to/repository

Лес git’а

Немного теории. Git в своей работе управляет тремя структурами, которые называются деревьями. Первое — это рабочая директория, в ней хранятся файлы, с которыми вы прямо сейчас работаете. Ну, она ж рабочая, логично. Второе — это Index, этакий чек-поинт, который позволяет вам вносить изменения и ничего не портить. А третье — это HEAD, который указывает на последний сделанный вами коммит. (Чтобы вы не запутались в терминологии: коммит *(англ. commit)* — это сохранение состояния проекта в репозиторий. Короче, считайте, новая версия.)

Так вот, чтобы вы не заблудились в этих трех соснах, запомните две крутые команды: add и commit. Они позволят вашей работе спокойно бродить по git’у, сохраняясь, куда надо. Если вы придумали что-то гениальное и тут же внесли изменение в рабочую копию проекта, то не спешите сразу коммитить! Сначала испытайте в Index’е, для этого выполните:

git add [имя\_файла]

если вы внесли изменение только в один файл, или

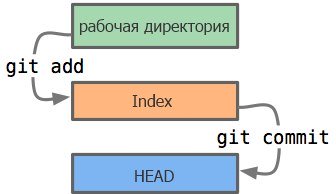
git add \*

если вы хорошо потрудились поменяли сразу кучу исходников. Изменения положительны? Хорошо потестили? Тогда скорее коммитить:

git commit -m "Commit message"

Вы, конечно же, плохой парень и не оставляете комментариев в коде. Но git — другое дело. Не ленитесь оставлять поясняющие сообщения: будьте уверены, вам хватит других проблем, кроме как разбираться, что же поменялось в этом коммите по сравнению с прошлой версией.

Вот вам, кстати, поясняющая картинка:

[](https://cdn.tproger.ru/wp-content/uploads/2016/06/EXcwUk86PWs.jpg)

Теперь файл(-ы) прочно обосновались в HEAD вашей рабочей локальной копии. Оттуда их не выгнать, но в вашем удаленном репозитории их все еще нет. Давайте сунем их еще и туда! Используйте:

для связи с удаленным репозиторием, находящемся на сервере git-a:

git remote add origin [сервер]

и далее:

git push origin master

Только вместо master напишите название нужной ветки. Ах да, вы же еще не знаете, что такое ветки. Ну ладно, пока что запомните это место, а когда прочитаете про ветвление, вернетесь сюда.

Ветвление

По-английски эта штука зовется *branching* — лучше как следует вникните в этот вопрос и почитайте про ветвление подробнее, я вас с ним только познакомлю. **Ветвление** используется для одновременной и независимой разработки разных фич (ну, или накопления большего количества багов, ведь исходного кода становится больше). Основной веткой является **master** — она появляется при создании репозитория. Другие ветки — это песочницы, когда достаточно в них наиграетесь, слейте в единое целое в master. Сейчас поясню, как это делается.

Создание новой ветки

Вот вы решили проработать какую-нибудь новую фичу. Создайте для нее новую ветку:

git checkout -b [новая\_ветка]

Ах да, фантазия-то у вас, наверное, работает на полную катушку, ну да поумерьте её в деле именования веток: назвать ветку можно только именем, допустимым для переменной в вашем любимом языке.

Переключение между ветками

Надо сделать перерыв в работе с этой фичей и переключиться на другую ветку? Используйте (если работаете с локальным репозиторием, то указывать его имя не обязательно):

git checkout [репозиторий]/[ветка]

Ну, а если вы уже совсем не хотите с ней работать, то удалите ее совсем:

git branch -d [ветка]

Со своей веткой вы можете творить любые непотребства: ее никто не увидит, пока вы сами ее не пропушите в удаленный репозиторий командой:

git push origin [ветка]

Слияние веток

Чтобы слить ветку в ту, с которой вы сейчас работаете, используйте:

git merge [ветка]

Но, понятное дело, это все приводит к конфликтам. И это реально проблема. Так что попробуйте исправлять все ручками прямо в директории с репозиторием. Только потом не забудьте пометить, что вы их «слили»:

git add [имя\_файла]

Кстати, ветки можно сравнить:

git diff [одна\_ветка] [другая\_ветка]

Так, теперь приступим к более решительным действиям. Будем обновлять свой репозиторий в соответствии с самым свежим коммитом. Сделать это очень просто (а вот вернуть обратно не очень, поэтому трижды подумайте, прежде чем совершать эту ужасную ошибку):

git pull

Я, конечно, понимаю, что вы слишком круты, чтобы оставлять какие-либо пометки на будущее — все держите в голове — но все-таки рекомендую вам оставлять тэги. И это не моя выдумка, так делают многие:

git tag [tag] [первые\_десять\_символов\_соответствующего\_коммита]

Вы не знаете, какие первые символы у имени нужного коммита? Не беда, смотрите в историю репозитория — его лог:

git log

Там есть куча разных параметров для использования этой полезной штуковины, ну да погуглите их сами. Ах да, кстати, мы уже писали как-то про то [как сделать git log более информативным](https://tproger.ru/translations/git-log-enhanced/).

Вот черт, я не то смержил!

Ну, а теперь давайте расскажу вам, как исправлять свои ошибки, хотя вы уверены, что не будете их делать. Если проблема только в одном файле, то вот вам Ctrl+Z для HEAD’a:

git checkout -- [имя\_файла]

Но если проблема находится аж в локальном репозитории, то зачистите там все и верните версию с сервера:

git fetch origin

git reset --hard origin/master

Да, чувак, тут все по-hard’овому. Это git.

Фичи git’а

Если вы ленивый, и вам не охота по-трупрогерски все писать в оболочке своей ОСи, то можете использовать GUI git’а:

gitk

Проще всего вызвать Gitk из командной строки: Просто перейдите в директорию с репозиторием и наберите: gitk.exe

В [источнике](https://rogerdudler.github.io/git-guide/) найдете еще кучу других GUI-шек.  
Если вам стандартный вывод git’а кажется скучным, раскрасьте его:

git config color.ui true

Ну, и есть еще такая штука — интерактивное индексирование. Когда у вас будет уже достаточно большой проект, то ужать представление index’а в log’е можно будет так:

git add -i

Надеюсь, это руководство поможет вам на ранних этапах не запутаться в работе с git’ом и **вы наконец научитесь следить за своими бекапами**.