Краткий гайд по Git

1. Что такое Git.

Git – распределенная система контроля версий. Также это набор консольных утилит для отслеживания и фиксации изменений в файлах. Git хранит версии файлов (файловая система).

Каждая копия репозитория Git самодостаточна.

2. Установка.

- **Linux**. В терминале наберите: sudo apt-get install git
- **OS X**. Установите homebrew. Затем в терминале наберите: brew install git
- **Windows**. Git for Windows. https://gitforwindows.org/ или https://git-scm.com/download/win. Последняя версия на момент написания – 2.17.0.

3. Настройка.

Откройте консоль гита.

- C:\Program Files\Git\ git-bash или git-cmd.
- Перейти к нужной папке, ПКМ по ней, в контекстном меню выберите Git bash here.

Введите настройки пользователя:

git config --global user.name "My Name"

git config --global user.email "myEmail@example.com"

Эти настройки обязательны. Без них Git не позволит делать коммиты.

4. Создание нового репозитория.

Репозиторий – настроенная в Git папка для хранения и обработки файлов/их истории. Внутри себя содержит скрытую папку .git с системными файлами, которые хранят данные (файлы) репозитория и их историю в сжатом виде.

Откройте консоль для пустой папки, в которой вы хотите создать репозиторий, и введите:

git init

Git должен показать примерно следующее:

Initialized empty Git repository in "Путь к репозиторию»

5. Состояния файлов и коммиты.

Unstaged – файл физически добавлен в папку репозитория, но для него является новым и не зарегистрированным. Не добавлен в индекс.
Staged – файл добавлен в папку репозитория и добавлен в индекс – область файлов, готовых к фиксации/коммиту в репозитории.
Modified – уже существующий в репозитории файл был изменен. Не добавлен в индекс.
Unmodified – файл находится в состоянии последнего коммита – т.е. актуален и без изменений.

Для получения текущего состояния репозитория и файлов используйте команду:

git status

Коммит – зафиксированное состояние всех файлов репозитория на определенный момент времени, обладающее уникальным идентификатором – хэшем – и сообщением. Коммит фиксирует только те файлы, которые находятся в индексе/имеют статус Staged.

Пример хэша: 24b9da6552252987aa493b52f8696cd6d3b00373

6. Как сделать коммит.

В созданном репозитории добавьте файл hello.txt (имя и расширение может быть любым). Введите в него любое содержимое. Например, строку: «Hello, world!»
Получите статус репозитория в консоли. Он должен отобразить добавленный только что файл как **Unstaged.**

Добавьте файл в индекс с помощью команды:

git add hello.txt

Для добавления всех файлов в папке в индекс:

git add -A

Снова получите статус репозитория. Добавленный файл должен иметь статус **Staged**.
Теперь следует зафиксировать состояние репозитория:

git commit

Такая команда предложит вам ввести сообщение коммита в текстовом редакторе (по умолчанию Vim, но может быть и Notepad++). В Vim желательно использовать латиницу.
Для выхода из Vim с сохранением изменений нажмите Escape и введите :wq
Короткий вариант:

git commit –m "Commit message"

Снова получите статус репозитория. Он должен показать, что текущая рабочая область (индекс) пуста.

7. Удаленные репозитории.

Удаленный репозиторий – такой же репозиторий, как и созданный только что вами, но расположенный на специальном сервере. Например, Github. Если что-то случится с локальным репозиторием, его версия на сервере останется доступной для использования.

Есть два вариант для начала работы с удаленным репозиторием.

Первый способ – создать его на гитхабе и клонировать себе (создать у себя локальную копию репозитория). Для этого создайте репозиторий, нажмите на “Clone or Download” и скопируйте адрес репозитория.

Затем откройте git bash для папки, в которую хотите клонировать репозиторий и введите:

git clone https://github.com/your-project.git

Второй способ – указать для локального репозитория ссылку на удаленный репозиторий с его названием. Для этого введите в консоли git команду:

git remote add origin https://github.com/your-project.git

Origin – имя удаленного репозитория. Можно задать и другое. Для локального репозитория можно добавить ссылки на несколько удаленных репозиториев.

8. Связь с удаленным репозиторием.

Получить состояние удаленного репозитория:

git fetch origin

Получить состояние удаленного репозитория и попытаться влить (merge) его в выбранную ветку:

git pull origin master

Здесь master – имя основной ветки репозитория. У каждого репозитория есть такая ветка.

9. Состояние репозитория.

Получить состояние репозитория:

git log

Получить историю действий в репозитории:

git reflog

Head – указатель на текущий коммит.

Получить содержимое коммита:

git show commit_hash

Получить разницу между коммитами:

git diff commit_hash_1..commit_hash_2

10. Ветки.

Хорошей практикой в разработке считается иметь основную **ветку**, в которой хранится актуальная и рабочая версия вашей программы/продукта, а для добавления нового функционала создавать новые ветки, которые будут являться копией основной ветки на состояние выбранного коммита. По умолчанию в репозитории есть главная ветка master. Все прочие ветки растут из нее.

Плюсы такого подхода:

- Основная и рабочая версия кода сохраняется.
- Ветки изолированы друг от друга, что позволяет нескольким разработчикам параллельно вносить изменения в проект/продукт/программу.
- Изменения в других ветках не затронут вашу, пока вы явно не притяните их себе.

Создание новой ветки (из коммита ветки, на которой вы сейчас находитесь):

git branch branch_name

Переход на другую ветку:

git checkout branch_name

Короткий вариант (создать ветку и сразу перейти в нее):

git checkout –b branch_name

11. Слияние веток.

Слияние веток позволяет перенести изменения из одной ветки в другую, создавая в результате коммит с новым состоянием целевой ветки.

Слияние происходит из ветки, на которой вы находитесь, с указанной веткой.

Имея ветки master и feature ветки, merge производится так:

git checkout master
git merge feature

Обычно принято удалять стороннюю ветку после слияния с основной.

Команда git pull производит слияние вашей локальной ветки и выбранной ветки на удаленном репозитории.

12. Перебазирование ветки.

Если рабочая ветка, от которой вы создали свою, изменилась, и вам нужно внести эти изменения себе, можно применить **перебазирование** (**rebase**), как бы заставляя вашу ветку вырасти из другого, обычно более свежего коммита. Иными словами, происходит последовательное применение всех коммитов ветки feature к выбранному коммиту ветки master.

Переходим на нужную ветку и выполняем rebase на выбранную ветку:

git checkout feature
git rebase master

Старые коммиты ветки feature остаются в недрах гита, но теряются для просмотра в git log, т.к. после перемещения они отображаются уже с новым хэшем.

13. Решение конфликтов.

При выполнении merge или rebase может произойти конфликт, если в каком-то файле на одном и том же месте в разных ветках было указано разное содержимое. Консольная версия уведомит вас о наличии конфликта и предложит его решить.

Открыв файл с конфликтом, вы увидите саму конфликтную позицию с текстом HEAD и разделителем =======. Над разделителем вы увидите ваш последний коммит, а под ним – конфликтующий из другой ветки. Вы можете как выбрать коммит из любой из веток, так и внести совершенно независимые для них изменения. Решение конфликта затем нужно оформить отдельным коммитом.

14. Откат файла к другому состоянию.

Команду checkout можно применить к отдельному файлу, чтобы откатить его до состояния выбранного коммита:

git checkout 1234abcd hello.txt

15. Перемещение между коммитами.

Для **перемещения** между коммитами используется следующая команда:

git reset --reset_type 1234abcd

Есть три **вида** перемещения:

Hard – жесткое перемещение HEAD текущей ветки к выбранному коммиту. Все лишнее из ветки до перемещения стирается, в т.ч. и коммиты между старым HEAD и новым.

Soft – мягкое перемещение HEAD. Никакие коммиты не удаляются.

Mixed – перемещение HEAD к выбранному коммиту, разница между старым и новым HEAD помещается в индекс. Используется для другого разделения коммитов или их схлопывания в один.

16. Отмена/правки коммита.

Для отмены последнего коммита – HEAD – используется команда:

git revert HEAD

Для отмены изменений другого коммита в команде revert укажите хэш нужного коммита. Эта команда создаст новый коммит, который отменит изменения выбранного коммита. Внимание – могут возникнуть конфликты!

Для изменения последнего коммита можно использовать команду:

git commit –amend

С ее помощью можно поменять текст коммита или его содержимое. Его хэш при этом изменится.

17. Как скрыть файлы от Git.

Если вам нужно скрыть какие-либо файлы или директории от всевидящего ока Git, можно в папку с репозиторием добавить файл .gitignore (формально это файл без имени с таким вот расширением), в котором будут указаны исключения для отслеживания изменений. Обычно в исключение добавляются папки с результатами сборки, файлы сборки (.exe и .dll), файлы, хранящие данные вашей среды разработки и т.д.

Пример содержимого файла:

*.txt – шаблон для игнорирования всех файлов формата .txt

Build/ - рекурсивное исключение папки Build и всех вложенных из списка отслеживаемых.

Hello.txt – исключение отдельного файла.