README.md

Шпаргалка по Git

Команды Git на примере `GitCheatSheet`

Создать репозиторий

1. Создать каталог для проекта
2. Перейти в него
3. Инициализировать репозиторий

```
mkdir GitCheatSheet
cd GitCheatSheet/
git init
```

Удалить репозиторий

1. Перейти в каталог с репозиторием
2. Удалить каталог `.git`

```
cd GitCheatSheet
rm -rf .git
```

Проверить состояние репозитория

```
git status
```

Подготовить файлы к сохранению

```
git add file_name.txt # подготовить к сохранению 1 файл
git add file1.txt file2.txt # подготовить к сохранению несколько файлов
git add --all # подготовить к сохранению все файлы в репозитории
git add . # подготовить к сохранению всю текущую папку
```

Выполнить коммит

```
git commit # будет открыт текстовый редактор, для ввода комментария к коммиту
git commit -m 'Мой первый коммит!' # коммит с комментарием, введённым из командной строки
```

Вывести историю коммитов

```
git log
```

`git log` выводит коммиты в обратном хронологическом порядке - последние коммиты оказываются первыми сверху

Следующая команда выводит историю коммитов в сокращённом формате:

```
git log --oneline
```

Изменть последний коммит

```
git commit --amend
```

Чтобы изменить содержимое коммита без изменения сообщения:

```
git commit --amend --no-edit
```

Чтобы изменить только сообщение:

```
git commit --amend -m 'новый текст сообщения'
```

Клонирование удалённого репозитория

```
git clone <адрес_репозитория>
```

После клонирования репозитория не требуется связывание удалённого и локального репозитория, а также выполнение команды `git push -u origin main` или других комадн, связывающих ветви `main` в удалённом и локальном репозитории.

Связать удалённый и локальный репозиторий

В каталоге с репозиторием выполнить

```
git remote add origin <адрес_репозитория>
```

После связывания первую отправку фиксаций в удалённый репозиторий нужно выполнять следущим образом:

```
git push -u origin main
```

Проверить, что удалённый и локальный репозитории связаны

```
git remote -v
```

Отправить изменения в удалённый репозиторий

```
git push
```


Получить изменения из удалённого репозитория

```
git pull
```

Хеш

Хеш - основной идентификатор коммита.

Хеш - это результат обработки информации о коммите (когда был сделан коммит, содержимое файлов в репозитории на момент коммита и ссылка на предыдущий коммит) алгоритмом вычисляющим хеш-функцию (функцию свёртки).

Обычно хеш - это короткая строка (длина зависит от алгоритма хеширования), состоящая из цифр (0-9) и латинских букв (A-F). Хеш обладает следующими свойствами:

- если хеш получить дважды для одного и того же набора входных данных, то результат будет гарантированно

одинаковый;

- если в исходных данных поменяется бы один символ, то хеш тоже изменится (сильно).

HEAD

Файл **HEAD** - один из служебных файлов папки `.git`. Он указывает на коммит, который сделан последним (то есть на самый новый). Внутри **HEAD** - ссылка на служебный файл: `refs/heads/main`, содержащий хеш последнего коммита.

Жизненный цикл файла в Git

```
graph LR;
  untracked       -- git add --> staged\ntracked;
  staged\ntracked -- git commit --> tracked;
  tracked -- изменения --> modified\ntracked;
  modified\ntracked -- git add --> staged\ntracked;
```