Шпаргалка. Начало работы с Git

### Инициализация репозитория

git init (от англ. ***init***ialize, «инициализировать») — инициализируй репозиторий.

### Синхронизация локального и удалённого репозиториев

git remote add origin https://github.com/YandexPracticum/first-project.git (от англ. remote, «удалённый» + add, «добавить») — привяжи локальный репозиторий к удалённому с URL https://github.com/YandexPracticum/first-project.git;

git remote -v (от англ. ***v***erbose, «подробный») — проверь, что репозитории действительно связались;

git push -u origin main (от англ. push, «толкать») — в первый раз загрузи все коммиты из локального репозитория в удалённый с названием origin.

💡 Ваша ветка может называться master, а не main. Подправьте команду, если это необходимо.

git push (от англ. push, «толкать») — загрузи коммиты в удалённый репозиторий после того, как он был привязан с помощью флага -u.

### Подготовка файла к коммиту

git add todo.txt (от англ. add, «добавить») — подготовь файл todo.txt к коммиту;

git add --all (от англ. add, «добавить» + all, «всё») — подготовь к коммиту сразу все файлы, в которых были изменения, и все новые файлы;

git add . — подготовь к коммиту текущую папку и все файлы в ней.

### Создание и публикация коммита

git commit -m "Комментарий к коммиту." (от англ. commit, «совершать», фиксировать» + ***m***essage, «сообщение») — сделай коммит и оставь комментарий, чтобы было проще понять, какие изменения сделаны;

git push (от англ. push, «толкать») — добавь изменения в удалённый репозиторий.

### Просмотр информации о коммитах

git log (от англ. log, «журнал [записей]») — выведи подробную историю коммитов;

git log --oneline (от англ. log, «журнал [записей]» + oneline, «одной строкой») — покажи краткую информацию о коммитах: сокращённый хеш и сообщение.

### Просмотр состояния файлов

git status (от англ. status, «статус», «состояние») — покажи текущее состояние репозитория.

### Добавление изменений в последний коммит

git commit --amend --no-edit (от англ. amend, «исправить») — добавь изменения к последнему коммиту и оставь сообщение прежним;

git commit --amend -m "Новое сообщение" — измени сообщение к последнему коммиту на Новое сообщение.

💡 Выйти из редактора Vim: нажать Esc, ввести :qa!, нажать Enter.

### «Откат» файлов и коммитов

git restore --staged hello.txt (от англ. restore, «восстановить») — переведи файл hello.txt из состояния staged обратно в untracked или modified;

git restore hello.txt — верни файл hello.txt к последней версии, которая была сохранена через git commit или git add;

git reset --hard b576d89 (от англ. reset, «сброс», «обнуление» + hard, «суровый») — удали все незакоммиченные изменения из staging и «рабочей зоны» вплоть до указанного коммита.

### Просмотр изменений

git diff (от англ. ***diff***erence, «отличие», «разница») — покажи изменения в «рабочей зоне», то есть в modified-файлах;

git diff a9928ab 11bada1 — выведи разницу между двумя коммитами;

git diff --staged — покажи изменения, которые добавлены в staged-файлах.

### Алгоритм-шпаргалка для создания PR

1. Склонировать репозиторий.
   1. Если вы не участник проекта, предварительно сделать «форк» исходного репозитория.
   2. На странице репозитория или «форка» нажать кнопки: **Code → SSH → скопировать ссылку**.
   3. Выполнить команду git clone <ссылка на репозиторий>.
2. Создать ветку для вашей задачи: git checkout -b my-task-branch-name.
3. Добавить и «закоммитить» изменения, которые вы хотите внести в проект.
4. «Запушить» ветку: git push --set-upstream origin HEAD или git push -u origin my-task-branch-name.
   1. GitHub (с помощью Git) выведет ссылку на создание PR. По ней нужно перейти.
   2. PR можно также создать через интерфейс GitHub.
5. Сообщить о пул-реквесте ревьюеру.
   1. Иногда ревьюеры назначаются автоматически, тогда сообщать не нужно.
6. Обсуждать с ревьюером предлагаемые изменения и вносить правки, пока эти изменения не будут одобрены (пока не будет получен «апрув»).

6.1. Если кто-то добавил конфликтующие изменения в main, пока ваш PR был на ревью, нужно разрешить конфликт:

* 1. Обновить main: git checkout main && git pull.
  2. Влить main в свою ветку: git checkout my-task-branch-name && git merge main.
  3. Разрешить конфликты слияния с помощью IDE или вручную.
  4. Создать коммит слияния: git commit --no-edit или git commit -m 'merge main'.
  5. Сделать git push своей ветки.

1. Нажать кнопку **Merge** или подождать, пока её нажмёт кто-то ещё.
2. Ещё раз обновить main, чтобы «подтянуть» ваши изменения в основную ветку локального репозитория: git checkout main && git pull.
3. Вы великолепны! Можете начинать снова со второго пункта.

### Алгоритм-шпаргалка для разрешения конфликтов слияния

1. Открыть проект в IDE (VS Code, IDEA или другие).
2. Открыть файл, в котором есть конфликт.
3. Выбрать, какие части файла нужно взять из одной ветки, а какие — из другой.
4. Когда конфликты разрешены, сделать коммит: git commit --no-edit или git commit -m 'merge branch <название ветки>'.

Готово! Теперь вы можете приступать к работе над реальным проектом. Git будет вашим новым союзником!