pwd — просмотр вашего текущего местоположения;

ls — список папок и файлов в текущей директории, где была выполнена команда;

ls -a — список открытых и скрытых папок и файлов в текущей директории, где была выполнена команда;

cd ~ — переход в домашнюю директорию текущего пользователя;

cd .. — переход на один уровень вверх в иерархии файловой системы;

cd folder\_name — переход в выбранную папку;

mkdir folder\_name — создать папку с указанным именем.

**git config --global user.name "Tara Routray"**

**git config --global user.email** [**dev@tararoutray.com**](mailto:dev@tararoutray.com)

**git remote:** просмотр удалённых репозиториев связанных с локальным

**git remote add origin** Привязка локального репозитория к удалённому на GitHub

**git init** Создать пустой репозиторий Git или вновь инициализировать существующий можно параметром init. При инициализации он создаст скрытую папку. В ней содержатся все объекты и ссылки, которые Git использует и создаёт в истории работы над проектом.

**git clone [url]:** Клонирует удалённый репозиторий на локальную машину.

**git branch [имя]:** Создаёт новую ветку с указанным именем.

**git branch -D[имя]:** Удалить ветку

**git checkout -b [имя]:** создать ветку и автоматически перейти на нее

**git status:** Показывает текущее состояние рабочего дерева и индекса.

**git add [файл]:** Добавляет изменения в индекс для следующего коммита.

**git commit -m "сообщение":** Сохраняет изменения в локальном репозитории с сообщением.

**git log:** Отображает историю коммитов.

**git diff**: Показывает различия между изменениями в рабочем каталоге и последним коммитом.

**git commit --amend -m** "сообщение" эта команда изменяет только сообщение последнего коммита.

git commit --amend --no-edit Если хотите добавить файлы без изменения сообщения:

**git reset --hard [хеш\_коммита]:** Откатывает проект к указанному коммиту, удаляя все изменения в рабочем каталоге и индексе. Будьте осторожны, так как это приведёт к потере всех несохранённых изменений.

**git reset --soft [хеш\_коммита]:** Откатывает указатель ветки к заданному коммиту, сохраняя изменения в рабочем каталоге и индексе. Это полезно, если вы хотите сохранить изменения для дальнейшего редактирования.

**git revert [хеш\_коммита]:** Создаёт новый коммит, который отменяет изменения указанного коммита, сохраняя при этом историю. Это лучший вариант для публичных веток.

**Разница между revert и reset**

Команда git revert отменяет изменения, записанные только одним коммитом. Она не откатывает проект к более раннему состоянию, удаляя все последующие коммиты, как это делает команда git reset.

У команды revert есть два крупных преимущества по сравнению с reset. Во-первых, она не меняет историю проекта и производит операцию, безопасную для коммитов. Во-вторых, её объектом выступает конкретный коммит, созданный в любой момент истории, а git reset всегда берёт за точку отсчёта текущий коммит. К примеру, если нужно отменить старый коммит с помощью git reset, придётся удалить все коммиты, поданные после целевого, а затем выполнить их повторно. Следовательно, команда git revert — гораздо более удобный и безопасный способ отмены изменений.

**git restore (git restore --staged file\_name):** отмена изменений. Команда git restore возвращает файл к состоянию последнего коммита. Она отменяет все изменения, если файл не перенесён в индекс. Если файл попал в индекс, то вместе с названием команды нужно использовать опцию --staged.

**git checkout [хеш\_коммита]:** Позволяет просмотреть состояние проекта на момент указанного коммита, но не изменяет текущую ветку. Это временный способ вернуться к старому состоянию.

**git checkout** [ветка/коммит]: Переключается на указанную ветку или коммит.

**git switch branch\_name** Перед каждым переключением она автоматически проверяет рабочую директорию и не срабатывает, если переход на выбранную ветку может привести к потере данных

**git push [удалённый\_репозиторий] [ветка]:** Отправляет изменения на удалённый репозиторий.

**git pull [удалённый\_репозиторий] [ветка]:** Получает и объединяет изменения из удалённого репозитория.

**git stash:** Сохраняет текущие изменения и очищает рабочий каталог.

**git stash pop:** Восстанавливает последние сохранённые изменения из хранилища.

**git rm dirname/somefile.js** Удалять файлы из текущего рабочего дерева

**git merge [ветка]:** Сливает указанную ветку с текущей.

**git rebase [ветка]:** Переносит коммиты текущей ветки на вершину указанной ветки.

**git merge**

Создает новый коммит слияния, сохраняя полную историю изменений и ветвления.

Позволяет видеть, когда и как происходили слияния, что может усложнить историю.

**git rebase**

Перемещает коммиты из одной ветки на вершину другой, создавая линейную историю.

Упрощает понимание последовательности изменений, но может привести к потере истории, если используется неправильно