**Коммит** (commit) — **это** информация об измененных файлах. **Коммит** состоит из автора **коммита**, измененных файлов, HEAD и времени.

**Ветка** **в** Git **это** подвижный указатель на один из коммитов. Обычно **ветка** указывает на последний коммит в цепочке коммитов.

Pull request — предложение изменения кода в чужом репозитории.

**Хеши** — идентификаторы объектов. **Хэш** **коммита** 40-символьный, но можно использовать короткую запись - первые 7 символов **хэша**. Команда git log --oneline выводит именно короткий **хэш**.

**Fork** - **это** фича (свойство) **github**, которая позволяет создать полноценную копию репозитория в вашем аккаунте. **Fork** - это вcего навсего копия репозитория.

**Дифф** (**diff**) — разница двух состояний (**коммитов**, веток, подготовленных или модифицированных файлов). Трехсторонний **дифф** (three-way **diff**) — **дифф**, возникающий при мердже и решении конфликтов. Является разницей трех состояний: состояния репозитория в текущей ветке, состояния в целевой ветке слияния и общего состояния между **этими** ветками (состояния в базе слияния).

Статус изменений - Команда **git** **status** отображает состояние рабочего каталога и раздела проиндексированных файлов. С ее помощью можно проверить индексацию изменений и увидеть файлы, которые не отслеживаются **Git**.

Команда **git** **add** — **это** первая команда в цепочке операций, предписывающей **Git** «сохранить» снимок текущего состояния проекта в истории коммитов.

Например, «**git** **push**» означает мерж локальных изменений в удаленный репозиторий, а «**git** **pull**» — наоборот, мерж изменений из удаленного репозитория в локальный.

**git** **fetch** — забирает изменения удаленной ветки из репозитория по умолчания, основной ветки; той, которая была использована при клонировании репозитория.

При использовании pull, git пытается сделать всё за вас. Он сливает любые внесённые коммиты в ветку, в которой вы сейчас работаете. Команда pull автоматически сливает коммиты, не давая вам сначала просмотреть их. Если вы не пристально следите за ветками, выполнение этой команды может привести к частым конфликтам.

При использовании fetch, git собирает все коммиты из целевой ветки, которых нет в текущей ветке, и сохраняет их в локальном репозитории. Однако он не сливает их в текущую ветку. Это особенно полезно, если вам нужно постоянно обновлять свой репозиторий, но вы работаете над функциональностью, неправильная реализация которой может негативно сказаться на проекте в целом. Чтобы слить коммиты в основную ветвь, нужно использовать merge.

Грубо говоря, по дефолту git pull — это шоткод для последовательности двух команд: git fetch (получение изменений с сервера) и git merge (сливание в локальную копию).

**git** **init** — создание репозитория. Команда **git** **init** создает в директории пустой репозиторий в виде директории .**git**, где и будет в дальнейшем храниться вся информация об истории коммитов, тегах.

Команда **git** **config** — **это** удобная функция, которая используется для настройки значений конфигурации **Git** на глобальном и локальном уровнях проекта.

Git add имя файла (или \*), git commit -m …, git push

**Git** **clone** - команда клонирования репозитория - используется для первоначального копирования репозитория, т.е. для создания копии на вашей машине, когда у вас вообще ничего нет - скачает всю информацию.

**git** **reset** — возврат к определенному коммиту, откат изменений, «жесткий» или «мягкий». Помимо работы с индексом (см. выше), **git** **reset** позволяет сбросить состояние проекта до какого-либо коммита в истории.

Команда **git** **revert** — **это** операция для безопасной отмены изменений, действие которой направлено в будущее. Для отката изменений команда не удаляет из истории коммиты или родительские элементы, a создает новый коммит с отменой нужных действий. Использовать команду **git** **revert** безопаснее, потому что она не создает угрозу потери кода, в отличие от **git** reset.

1. **Git** — система управления версиями с распределенной архитектурой. В отличие от некогда популярных систем вроде CVS и Subversion (SVN), где полная история версий проекта доступна лишь в одном месте, в **Git** каждая рабочая копия кода сама по себе является репозиторием.
2. **Система контроля версий** страхует от ошибок и возвращает код в то состояние, когда всё работало. Контрольные точки называются коммитами. Один коммит — это пакет изменений, хранящий информацию с добавленными, отредактированными или удалёнными файлами кода.

Просмотр изменений в файловых системах

Команда git status отображает все файлы, которые различаются между тремя разделами. У файлов есть 4 состояния:

1. Неотслеживаемый (untracked) — находится в рабочей директории, но нет ни одной версии в HEAD или в области подготовленных файлов (Git не знает о файле).
2. Изменён (modified) — в рабочей директории есть более новая версия по сравнению с хранящейся в HEAD или в области подготовленных файлов (изменения не находятся в следующем коммите).
3. Подготовлен (staged) — в рабочей директории и области подготовленных файлов есть более новая версия по сравнению с хранящейся в HEAD (готов к коммиту).
4. Без изменений — одна версия файла во всех разделах, т. е. в последнем коммите содержится актуальная версия.