$ git init

Эта команда создаёт в текущем каталоге новый подкаталог с именем .git, содержащий все необходимые файлы репозитория — структуру Git репозитория.

Для получения копии существующего Git-репозитория, например, проекта, в который вы хотите внести свой вклад, необходимо использовать команду git clone.

$ git clone https://github.com/libgit2/libgit2

Эта команда создаёт каталог libgit2, инициализирует в нём подкаталог .git, скачивает все данные для этого репозитория и извлекает рабочую копию последней версии. Если вы перейдёте в только что созданный каталог libgit2, то увидите в нём файлы проекта, готовые для работы или использования. Для того, чтобы клонировать репозиторий в каталог с именем, отличающимся от libgit2, необходимо указать желаемое имя, как параметр командной строки:

$ git clone https://github.com/libgit2/libgit2 mylibgit

Эта команда делает всё то же самое, что и предыдущая, только результирующий каталог будет назван mylibgit.

Основной инструмент, используемый для определения, какие файлы в каком состоянии находятся — это команда git status. Если вы выполните эту команду сразу после клонирования, вы увидите что-то вроде этого:

$ git status

On branch master

Your branch is up-to-date with 'origin/master'.

nothing to commit, working tree clean

**Отслеживание новых файлов**

Для того чтобы начать отслеживать (добавить под версионный контроль) новый файл, используется команда git add. Чтобы начать отслеживание файла README, вы можете выполнить следующее:

$ git add README

git add. Это многофункциональная команда, она используется для добавления под версионный контроль новых файлов, для индексации изменений, а также для других целей, например для указания файлов с исправленным конфликтом слияния.

**Игнорирование файлов**

Зачастую, у вас имеется группа файлов, которые вы не только не хотите автоматически добавлять в репозиторий, но и видеть в списках неотслеживаемых. К таким файлам обычно относятся автоматически генерируемые файлы (различные логи, результаты сборки программ и т. п.). В таком случае, вы можете создать файл .gitignore. с перечислением шаблонов соответствующих таким файлам. Вот пример файла .gitignore:

$ cat .gitignore

\*.[oa]

\*~

**Просмотр индексированных и неиндексированных изменений**

Если результат работы команды git status недостаточно информативен для вас — вам хочется знать, что конкретно поменялось, а не только какие файлы были изменены — вы можете использовать команду git diff. Позже мы рассмотрим команду git diff подробнее; вы, скорее всего, будете использовать эту команду для получения ответов на два вопроса: что вы изменили, но ещё не проиндексировали, и что вы проиндексировали и собираетесь включить в коммит. Если git status отвечает на эти вопросы в самом общем виде, перечисляя имена файлов, git diff показывает вам непосредственно добавленные и удалённые строки — патч как он есть.

Допустим, вы снова изменили и проиндексировали файл README, а затем изменили файл CONTRIBUTING.md без индексирования. Если вы выполните команду git status, вы опять увидите что-то вроде:

$ git status

On branch master

Your branch is up-to-date with 'origin/master'.

Changes to be committed:

(use "git reset HEAD <file>..." to unstage)

modified: README

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git checkout -- <file>..." to discard changes in working directory)

modified: CONTRIBUTING.md

Чтобы увидеть, что же вы изменили, но пока не проиндексировали, наберите git diff без аргументов:

$ git diff

diff --git a/CONTRIBUTING.md b/CONTRIBUTING.md

index 8ebb991..643e24f 100644

--- a/CONTRIBUTING.md

+++ b/CONTRIBUTING.md

@@ -65,7 +65,8 @@ branch directly, things can get messy.

Please include a nice description of your changes when you submit your PR;

if we have to read the whole diff to figure out why you're contributing

in the first place, you're less likely to get feedback and have your change

-merged in.

+merged in. Also, split your changes into comprehensive chunks if you patch is

+longer than a dozen lines.

If you are starting to work on a particular area, feel free to submit a PR

that highlights your work in progress (and note in the PR title that it's

Эта команда сравнивает содержимое вашего рабочего каталога с содержимым индекса. Результат показывает ещё не проиндексированные изменения.

***Важно отметить, что git diff сама по себе не показывает все изменения сделанные с последнего коммита — только те, что ещё не проиндексированы. Такое поведение может сбивать с толку, так как если вы проиндексируете все свои изменения, то git diff ничего не вернёт.***

3. **Проиндексировать новые или измененные файлы** для последующего сохранения состояния.

git **add .** (проиндексировать все файлы в папке).

git **add** **index.html** (проиндексировать только файл index.html)

git **add** **css/button.css** **css/main.css** (проиндексировать сразу 2 файла, которые находятся в папке "css": button.css и main.css, указанные через пробел)

После: git status — можно проверить, проиндексировались ли все файлы.

4. **Зафиксировать изменения** — закоммитить. Сделать новые изменения отслеживаемыми.

Благодаря этой процедуре, в случае надобности — можно откатиться к предыдущему закоммиченному изменению. Иначе говоря: коммит — это как save в игре.

git **commit -m** **"Описание, что изменилось"**

Команда git branch делает несколько больше, чем просто создаёт и удаляет ветки. При запуске без параметров, вы получите простой список имеющихся у вас веток:

$ git branch

iss53

\* master

testing

Обратите внимание на символ \*, стоящий перед веткой master: он указывает на ветку, на которой вы находитесь в настоящий момент (т. е. ветку, на которую указывает HEAD). Это означает, что если вы сейчас сделаете коммит, ветка master переместится вперёд в соответствии с вашими последними изменениями.