Система контроля версий: Git.

git

Койлубаев Никита Тахирович

Ведущий разработчик

06.12.2022

Содержание

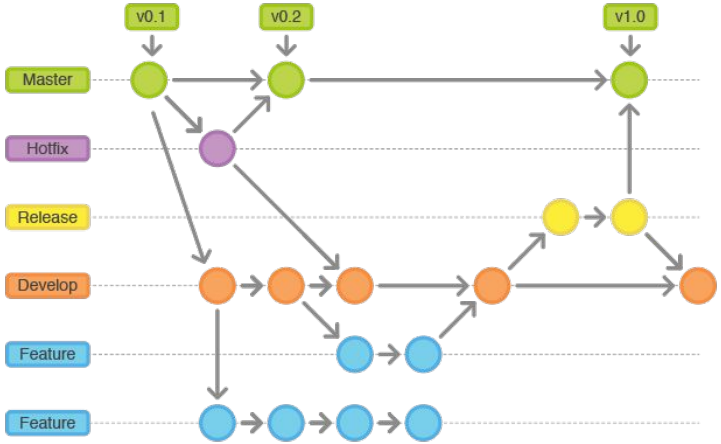
GIT. УСТАНОВКА

GIT. ВВЕДЕНИЕ

Для чего?

Git – это система управления версиями, которая имеет 2 задачи:

- Хранить информацию о всех изменениях в вашем коде.
- Обеспечить удобство работы над кодом в команде.

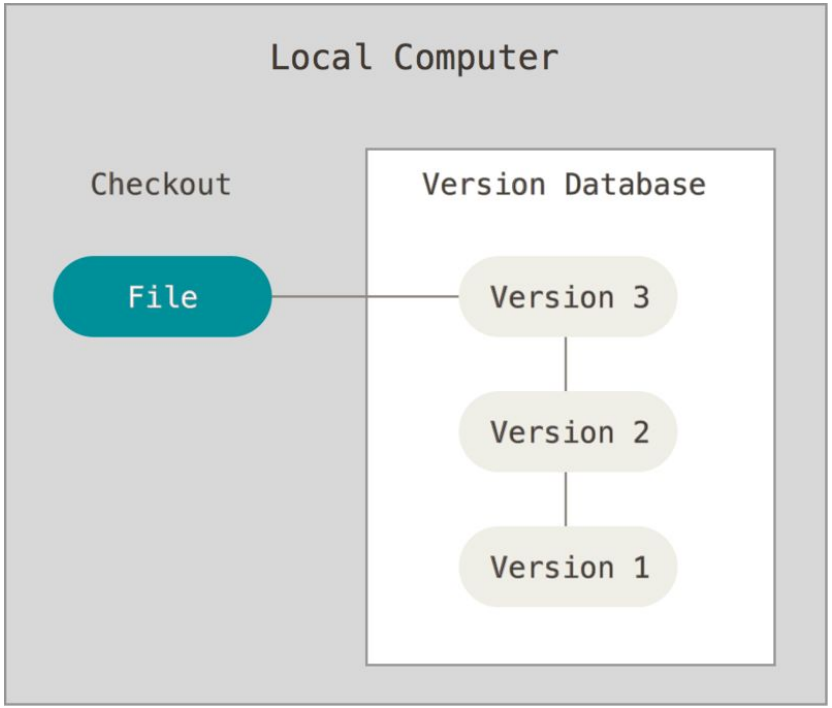


Преимущества:

- Бесплатный и open-source.
- Небольшой и быстрый.
- Резервное копирование.
- Простое ветвление

GIT. ВВЕДЕНИЕ

Виды системы контроля версий

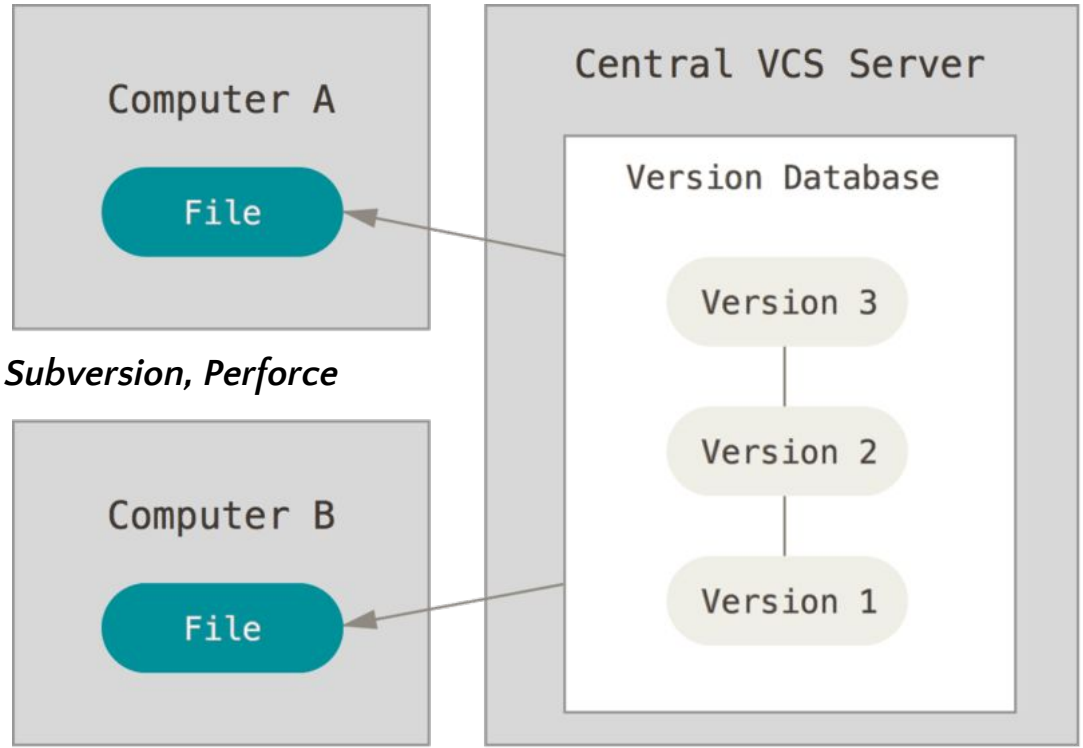


Локальные системы контроля версий

Плюсы	Минусы
Простота	Подвержен ошибкам.
	Сложность взаимодействия с другими участниками

GIT. ВВЕДЕНИЕ

Виды системы контроля версий



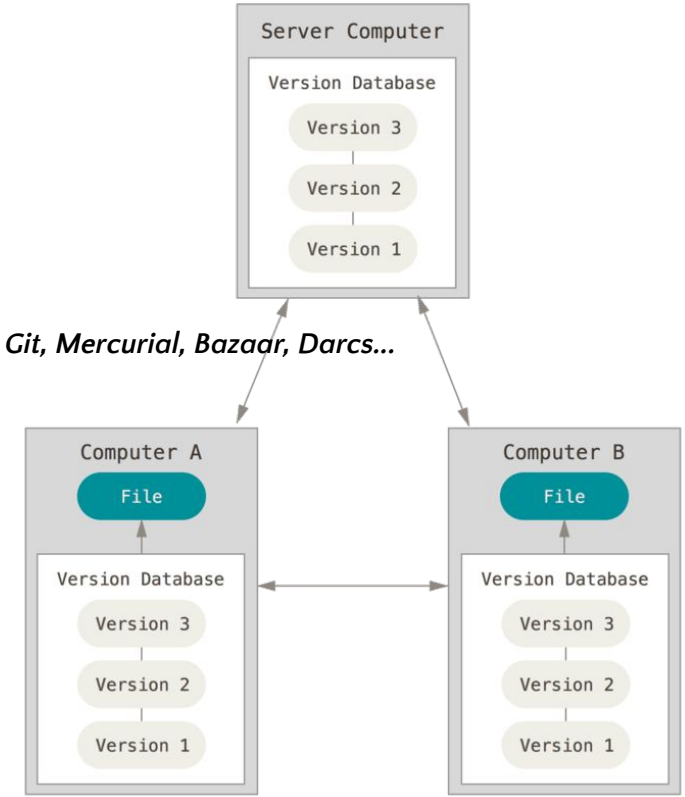
Subversion, Perforce

Плюсы	Минусы
Простота взаимодействия с другими участниками	Единственная точка отказа – сервер
Контроль над тем кто и что может делать	

Централизованные системы контроля версий – ЦСКВ

GIT. ВВЕДЕНИЕ

Виды системы контроля версий



Распределённые системы контроля версий – РСКВ

Плюсы	Минусы
Простота взаимодействия с другими участниками	Единственная точка отказа – сервер
Контроль над тем кто и что может делать	
Каждая копия репозитория – бэкап всех данных	
Возможность работать с несколькими удаленными репозиториями (большинство РСКВ)	

GIT. ВВЕДЕНИЕ

Краткая история GIT

Проект **Git** был создан ***Линусом Торвальдсом*** для упарвления разработки ядра *Linux*.

Первая версия **Git** была выпущена в **2005 году**. Причиной тому оказалось невозможность бесплатного использования децентрализованной СКВ BitKeeper для хранения изменений ядра Linux.

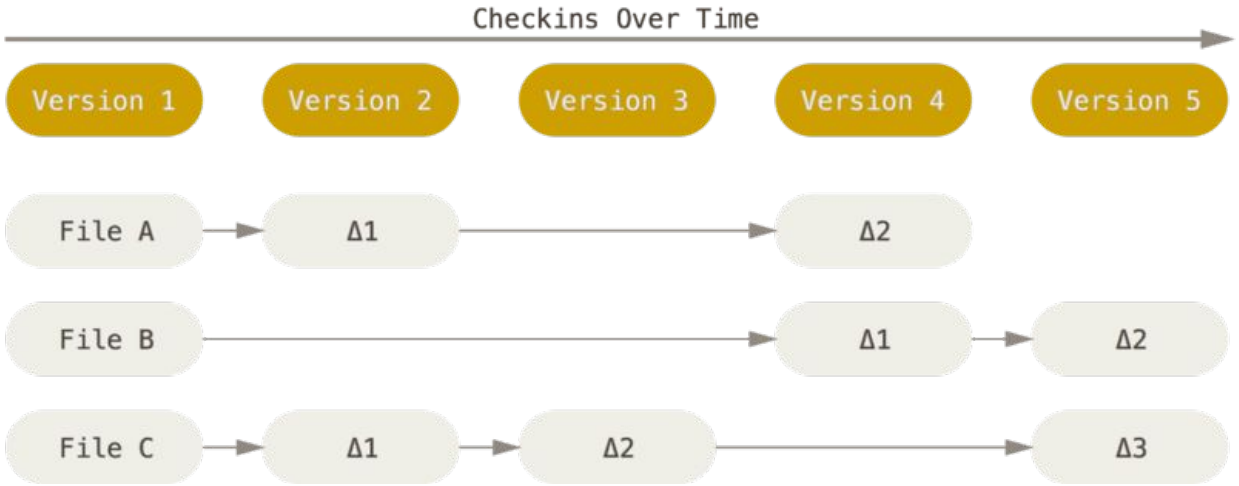
Некоторыми целями, которые преследовала новая система, были:

- Скорость
- Простая архитектура
- Хорошая поддержка нелинейной разработки (тысячи параллельных веток)
- Полная децентрализация
- Возможность эффективного управления большими проектами, такими как ядро Linux (скорость работы и разумное использование дискового пространства)

Git развился в простую в использовании систему, сохранив при этом свои изначальные качества.

GIT. ВВЕДЕНИЕ

Что такое Git?

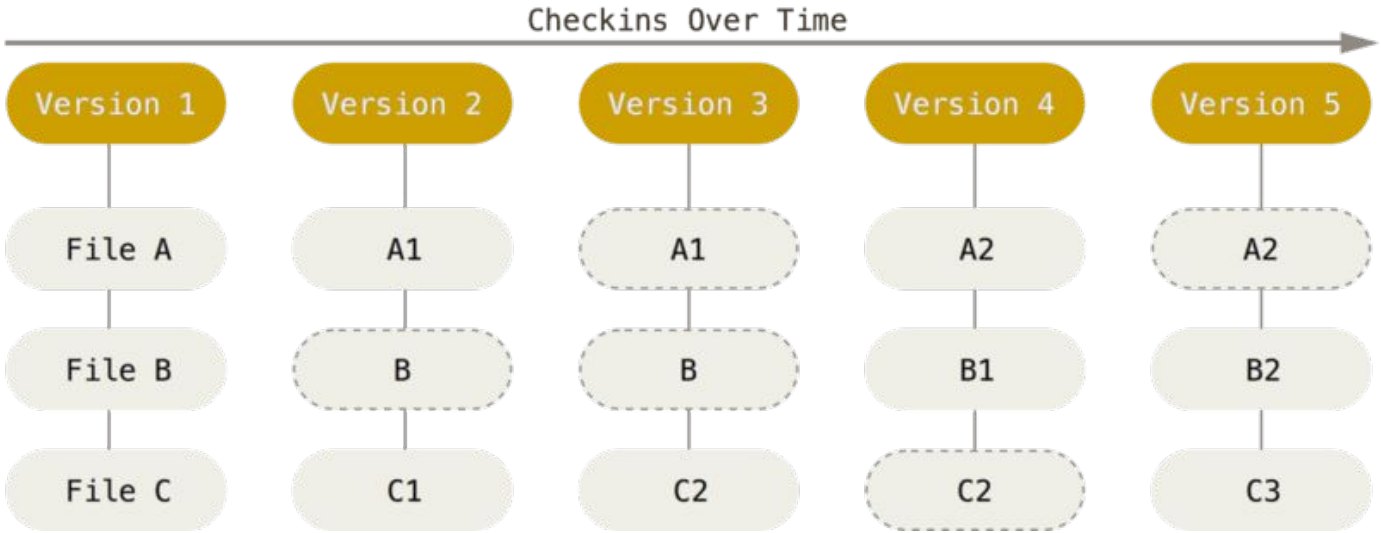


CSV, Subvesion, Perforce, Bazaar и т.д

Контроль версий, основанный на различиях

GIT. ВВЕДЕНИЕ

Что такое Git?



Git

Контроль версий, основанный на снимках. Git представляет свои данные, как поток снимков

GIT. ВВЕДЕНИЕ

Особенности

- Почти все операции выполняются локально.
- Целостность Git.
  - Для всего вычисляется хэш сумма. SHA-1 хеш. Пример: `24b9da6552252987aa493b52f8696cd6d3b00373`
  - Вы не потеряете информацию во время её передачи и не получите повреждённый файл без ведома Git. Git сохраняет все объекты в свою базу данных не по имени, а по хеш-сумме содержимого объекта
- Git обычно добавляет данные.
  - Когда вы производите какие-либо действия в Git, практически все из них только ***добавляют*** новые данные в базу Git.
  - Как и в любой другой СКВ, вы можете потерять или испортить свои изменения, пока они не зафиксированы, но, если изменения зафиксированы, то очень сложно их потерять, особенно, если регулярно синхронизируется база с другими репозиториями

GIT. ВВЕДЕНИЕ

Три состояния файлов

Состояния файлов	Описание
Изменен \ Modified	Это файлы, которые поменялись, но еще не были зафиксированы
Индексирован \ Staged	Это измененные файлы в текущей версии, отмеченные для включения в следующий коммит
Зафиксирован \ Commited	Это файлы, которые уже сохранены в вашей локальной базе

Секции проекта Git

Рабочая копия *(working directory)* является снимком одной версии проекта. Эти файлы извлекаются из сжатой базы данных в каталоге Git и помещаются на диск, для того чтобы их можно было использовать или редактировать.

Область индексирования *(staging area)* – это файл, обычно находящийся в каталоге Git, в нём содержится информация о том, что попадёт в следующий коммит. Техническое название на языке Git - ***“индекс”***

Каталог Git (Git directory) – это то место, где Git хранит метаданные и базу объектов вашего проекта. Это самая важная часть Git, которая копируется при *клонировании* репозитория с другого компьютера.

GIT. УСТАНОВКА

Linux

Дистрибутивы **RHEL** (Fedora, CentOs, ASPLinux, OpenSUSE, Linpus, Mandriva...):

$ sudo dnf install git-all

Дистрибутивы **Debian** (Debian, Ubuntu, Kali, PureOS...):

$ sudo apt install git

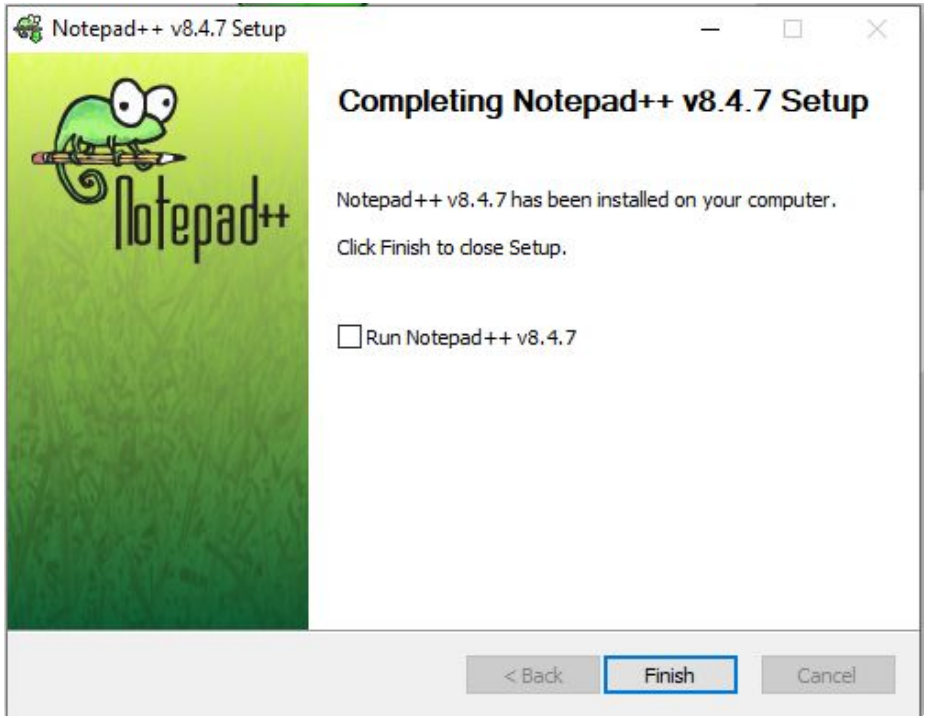
```
koylubaevnt@koylubaevnt-VirtualBox:~$ sudo apt install git
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc git-email git-gui gitk gitweb git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  git
0 upgraded, 1 newly installed, 0 to remove and 5 not upgraded.
Need to get 3 132 kB of archives.
After this operation, 18,8 MB of additional disk space will be used.
Get:1 http://ru.archive.ubuntu.com/ubuntu jammy-updates/main amd64 git amd64 1:2.34.1-1ubuntu1.5 [3 132 kB]
Fetched 3 132 kB in 1s (2 703 kB/s)
Selecting previously unselected package git.
(Reading database ... 174437 files and directories currently installed.)
Preparing to unpack .../git_1%3a2.34.1-1ubuntu1.5_amd64.deb ...
Unpacking git (1:2.34.1-1ubuntu1.5) ...
Setting up git (1:2.34.1-1ubuntu1.5) ...
koylubaevnt@koylubaevnt-VirtualBox:~$
```

Trash

GIT. УСТАНОВКА

Windows

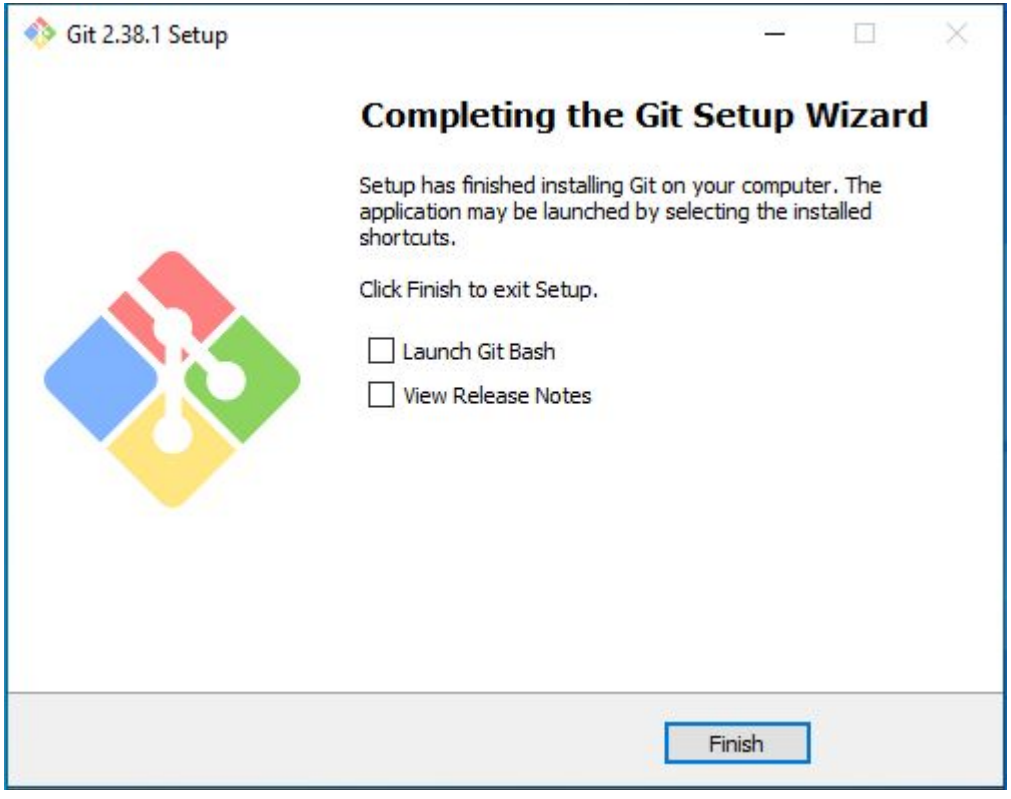
Установить Notepad++ (Необязательно). Можно пропустить. Сайт: **https://notepad-plus-plus.org/downloads/**



GIT. УСТАНОВКА

Windows

Скачать дистрибутив git с официального сайта. Ссылка на дистрибутив: **https://git-scm.com/download/win**.
Запустить установку скачанного файла.



GIT. УСТАНОВКА

Mac

Способ №1:
Установить Xcode Command Line Tools. В версии Mavericks (10.9) и выше вы можете добиться этого просто первый раз выполнив 'git' в терминале.
$ git --version
Если Git не установлен, вам будет предложено его установить.
Способ №2:
Воспользоваться бинарным установщиком. Доступен для скачивания с сайта Git: **https://git-scm.com/download/mac**.

GIT. УСТАНОВКА

Первоначальная настройка

Делается один раз после установки GIT. Но при необходимости можно поменять эти настрйоки выполнив те же команды.

Настройку выполняем через команду: git config.

Варианты сохранения параметров:

1. **[path]/etc/gitconfig** (Windows: относительно каталога msys)- настройки для всех пользователей системы и для всех их репозиториев. **git config --system**.
2. **~/.gitconfig** или **~/.config/git/config** (Windows: каталог $HOME) - настройки конкретного пользователя. **git config --global .**
3. **config** в каталоге Git репозитория (т. е. .git/config). **git config --local .** На самом деле это значение по умолчанию и вам нужно находиться где-то в репозитории Git, чтобы эта опция работала правильно.

Чтобы посмотреть все установленные настройки и узнать где именно они заданы, используйте команду:
$ git config --list --show-origin

GIT. УСТАНОВКА

Первоначальная настройка

- Имя и email пользователя:

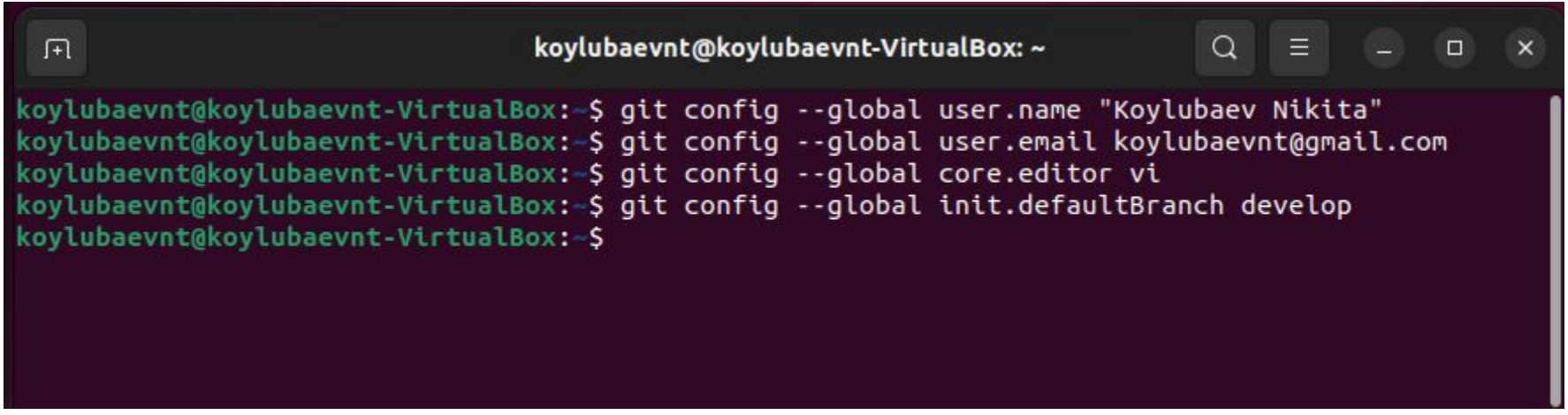
$ git config --global user.name “<username>”
$ git config --global user.email <email>

- Выбор редактора:

$ git config --global core.editor <editor-name or full-path-to-

- Настройка ветки по-умолчанию:

$ git config --global init.defaultBranch <branch-name>



GIT. УСТАНОВКА

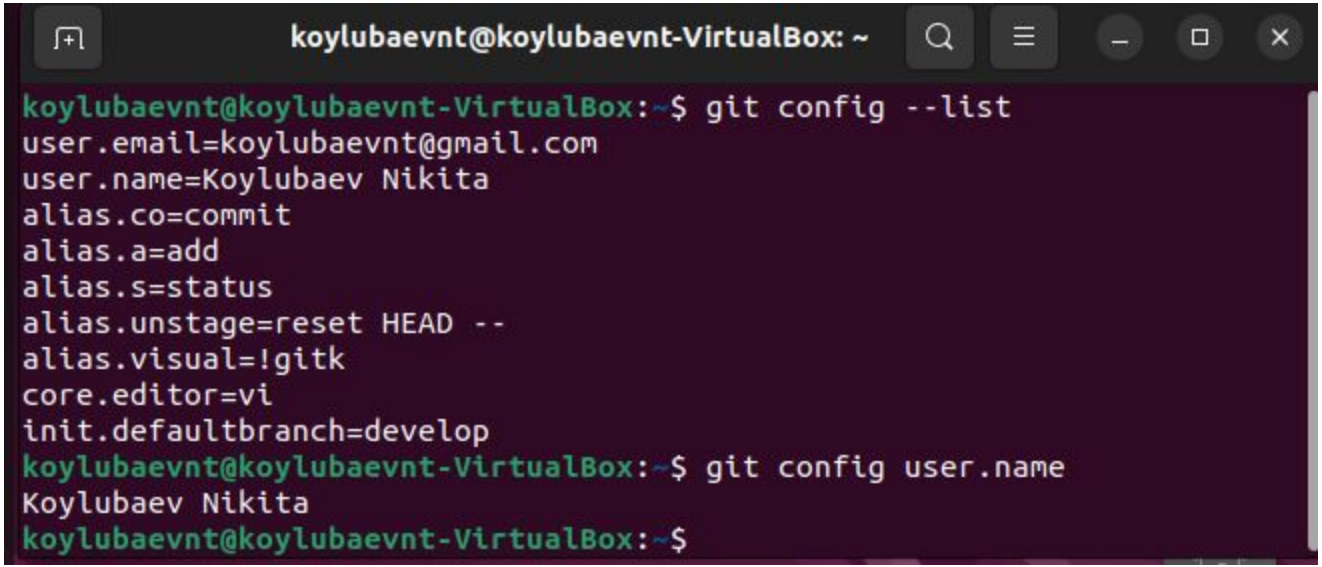
Первоначальная настройка

- Проверка настроек:

$ git config --list

- Проверка конкретного ключа **git config <key>**:

$ git config user.name

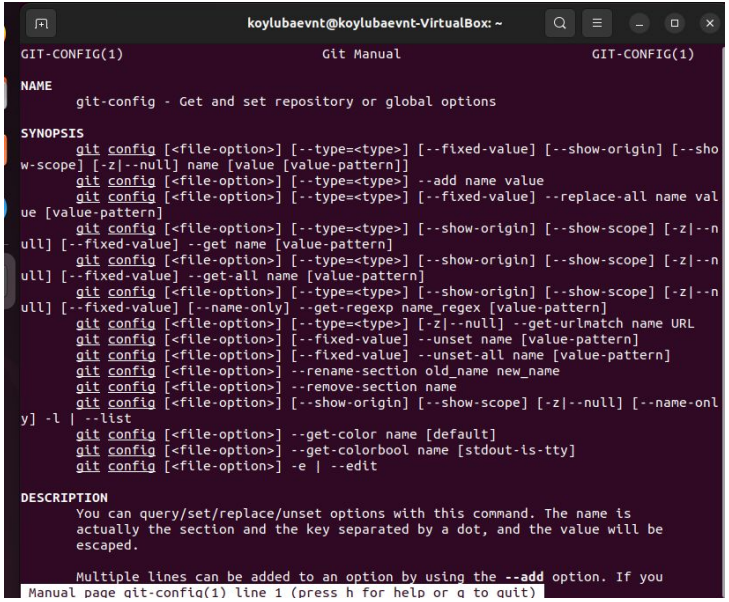


GIT. УСТАНОВКА

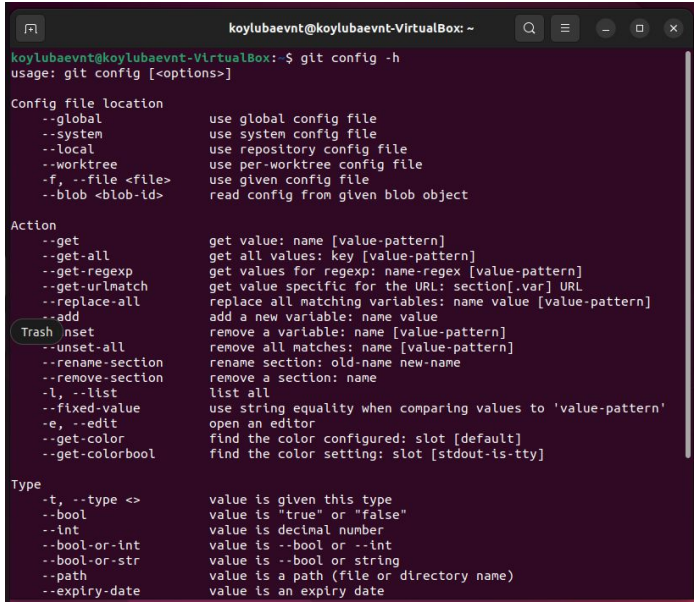
Помощь

- Помощь можно получить путем открытия страницы руководства. Это можно сделать следующими командами:

$ git help <command>
- $ git <command> --help
- $ man git <command>



- Получить список опций:
- $ git <command> -h



GIT. ОСНОВЫ

Создание git репозитория

- Превратить локальный каталог в репозиторий git:

```
$ mkdir git-course      -- создаем каталог (имитируем, что каталог существует)
$ cd git-course         -- переходим в каталог
$ git init              -- создаем git репозиторий (создастся подкаталог с именем .git)
```

```
koylubaevnt@koylubaevnt-VirtualBox:~$ mkdir git-course
koylubaevnt@koylubaevnt-VirtualBox:~$ cd git-course/
koylubaevnt@koylubaevnt-VirtualBox:~/git-course$ git init
Initialized empty Git repository in /home/koylubaevnt/git-course/.git/
koylubaevnt@koylubaevnt-VirtualBox:~/git-course$
```

- Клонировать существующий репозиторий git:

```
$ git clone <url>                   -- создаст папку с именем клонируемого проекта и заберет все версии файлов с сервера
$ git clone <url> <folder_name>     -- создаст папку с именем <folder_name> и заберет все версии файлов с сервера
```

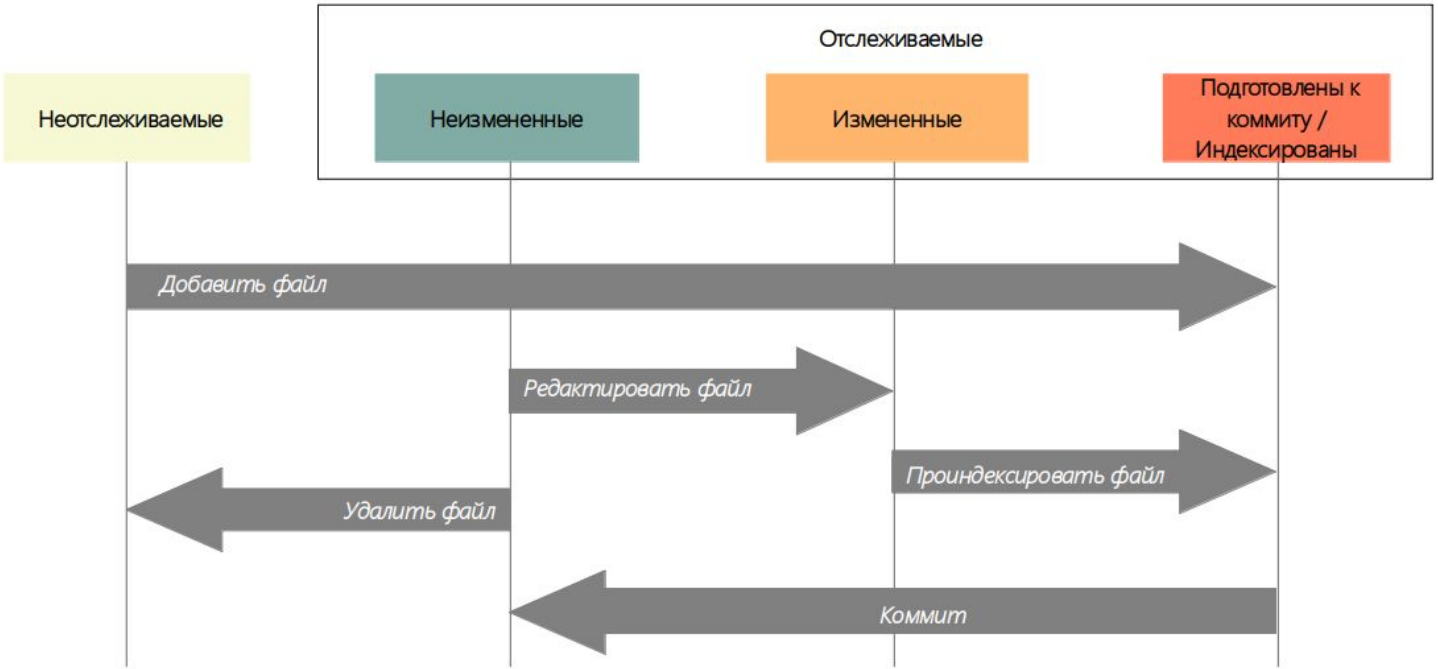
```
koylubaevnt@koylubaevnt-VirtualBox:~/Project$ ls
git-course  git-course-github
koylubaevnt@koylubaevnt-VirtualBox:~/Project$ git clone https://github.com/koylubaevnt/git-course-github.git git-course-github-renamed
Cloning into 'git-course-github-renamed'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), 12.76 KiB | 194.00 KiB/s, done.
koylubaevnt@koylubaevnt-VirtualBox:~/Project$ ls
git-course  git-course-github  git-course-github-renamed
koylubaevnt@koylubaevnt-VirtualBox:~/Project$
```

GIT. ОСНОВЫ

Запись изменений в git репозиторий

Каждый файл в репозитории может находиться в одном из двух состояний: под версионным контролем (**отслеживаемый**) и нет (**неотслеживаемый**).

Если вы ***клонируете*** репозиторий, то все файлы будут отслеживаемыми и неизменёнными, потому что Git только что их извлек и вы ничего пока не редактировали.



GIT. ОСНОВЫ

Запись изменений в git репозиторий: Определение состояния файлов

Основной инструмент, используемый для определения, какие файлы в каком состоянии находятся — это команда **git status**.
Результат выполнения этой команды сразу после клонирования можете увидеть на скриншоте ниже:

```
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$
```

Создадим в каталоге новый файл **NEW_FILE.md** и выполним команду **git status**:

```
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ echo 'My project' > NEW_FILE.md
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        NEW_FILE.md

nothing added to commit but untracked files present (use "git add" to track)
```

GIT. ОСНОВЫ

Запись изменений в git репозиторий: Отслеживание новых файлов

Чтобы начать отслеживать новый файл, используется команда **git add <file-name or pattern>**:

```
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git add NEW_FILE.md
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   NEW_FILE.md
```

GIT. ОСНОВЫ

Запись изменений в git репозиторий: Индексация изменённых файлов

Давайте модифицируем файл, который уже находится под версионным контролем и выполним команду **git status**:

```
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ echo 'Modify file under source control' >> README.md
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   NEW_FILE.md

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md
```

Отслеживаемый файл изменен в рабочем каталоге, но не проиндексирован

Проиндексируем файл **README.md**, для этого выполним команду **git add** и выполним команду **git status**:

```
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git add README.md
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   NEW_FILE.md
        modified:   README.md
```

Оба файла проиндексированы и войдут в следующий коммит

GIT. ОСНОВЫ

Запись изменений в git репозиторий: Индексация изменённых файлов

Мы вспомнили, что надо добавить еще кое-что в файл **README.md**, который уже проиндексирован. Добавим это в файл и выполним команду **git status**:

```
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ echo 'Add info while file is indexed' >> README.md
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   NEW_FILE.md
        modified:   README.md

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md
```

Нифига себе. А это что это такое?

Git индексирует файл в точности в том состоянии, в котором он находился, когда вы выполнили команду **git add**, а не в том, в котором он находится в вашем рабочем каталоге в момент выполнения **git commit**.

GIT. ОСНОВЫ

Запись изменений в git репозиторий: Индексация изменённых файлов

Чтобы попала именно последняя версия файла нужно выполнить команду **git add** еще раз. После выполним команду **git status**:

```
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git add README.md
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   NEW_FILE.md
        modified:   README.md
```

GIT. ОСНОВЫ

Запись изменений в git репозиторий: Сокращенный вывод статуса

Вывод команды **git status** довольно всеобъемлющий и многословный. Git также имеет флаг вывода сокращенного статуса, так что вы можете увидеть изменения в более компактном виде.
Для этого вам нужно выполнить команду **git status -s** или **git status --short**.

1. Отредактируем уже находящийся под версионным контролем **LICENSE**, проиндексируем его и еще раз изменим
2. Создадим файл **NEW_FILE.md** и проиндексируем его.
3. Создадим файл **ONE_ADDED_FILE.txt**, проиндексируем его и еще раз изменим.
4. Отредактируем уже находящийся под версионным контролем файл **README.md** и проиндексируем его
5. Создадим файл **UNTRACKED_FILE.txt**
6. Отредактируем уже находящийся под версионным контролем файл **COMMITED_FILE.txt**

Выполним команду **git status -s**:

```
~/Project/git-course-github  master  echo 'Add info to tracked file' >> LICENSE
~/Project/git-course-github  master  git add LICENSE
~/Project/git-course-github  master  echo 'More info to tracked file' >> LICENSE
~/Project/git-course-github  master  touch NEW_FILE.md
~/Project/git-course-github  master  git add NEW_FILE.md
~/Project/git-course-github  master  touch ONE_ADDED_FILE.txt
~/Project/git-course-github  master  git add ONE_ADDED_FILE.txt
~/Project/git-course-github  master  echo 'Add info' >> ONE_ADDED_FILE.txt
~/Project/git-course-github  master  echo 'README INFO' >> README.MD
~/Project/git-course-github  master  git add README.MD
~/Project/git-course-github  master  touch UNTRACKED_FILE.txt
~/Project/git-course-github  master  echo 'info' >> COMMITED_FILE.txt
~/Project/git-course-github  master  git status -s
 M COMMITED_FILE.txt
MM LICENSE
A  NEW_FILE.md
AM ONE_ADDED_FILE.txt
M  README.MD
?? UNTRACKED_FILE.txt
```

??	Неотслеживаемый файл
A	Добален
M	Отредактированн

GIT. ОСНОВЫ

Запись изменений в git репозиторий: Игнорирование файлов

Если есть группа файлов, которые вы не только не хотите автоматически добавлять в репозиторий, но и видеть в списках неотслеживаемых. К примеру: логи, результаты сборки, бинарные данные и т. п.
То для этого нужно создать специальный файл с именем **.gitignore** (точка в начале имени обязательна!) и в нем перечислить шаблоны, которые будут соответствовать таким файлам.

Простой пример файла **.gitignore**:

```
*.[ao]
*~
```

К шаблонам **.gitignore** применяются следующие правила:

- Пустые строки, а также строки, начинающиеся с #, игнорируются.
- Стандартные шаблоны являются глобальными и применяются рекурсивно для всего дерева каталогов.
- Чтобы избежать рекурсии используйте символ слеш (/) в начале шаблона.
- Чтобы исключить каталог добавьте слеш (/) в конец шаблона.
- Можно инвертировать шаблон, использовав восклицательный знак (!) в качестве первого символа.

GIT. ОСНОВЫ

Запись изменений в git репозиторий: Игнорирование файлов

Для настройки обычно применяются **Glob-шаблоны** (упрощенные RegExp) или прямые названия папок, файлов.

*	Соответствует 0 или более символам
[abc]	Соответствует любому символу из указанных в скобках (в примере a, b или c)
?	Соответствует одному символу

Хорошая практика – настроить .gitignore до того как начинать работать в репозитории!

*.a	Исключить все файлы с расширением .a
*.[oa]	Исключить все файлы с расширением .a или .o
!lib.a	Отслеживать файл lib.a даже если он подпадает под исключение
/TODO	Исключить файл TODO в корневом каталоге, но не файл в subdir/TODO
build/	Игнорировать все файлы в каталоге build/
doc/*.txt	Игнорировать файл doc/notes.txt, но не файл doc/server/arch.txt
doc/**/*.txt	Игнорировать все .txt файлы в каталоге doc/

GIT. ОСНОВЫ

Запись изменений в git репозиторий: Просмотр индексированных и неиндексированных изменений

Команда **git status** показывает только какие файлы были изменены.
Но если есть необходимость узнать что именно изменялось, то применяется команда **git diff.**

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

```
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git diff
diff --git a/LICENSE b/LICENSE
index 6829b58..f6ae6cc 100644
--- a/LICENSE
+++ b/LICENSE
@@ -673,3 +673,4 @@ the library.  If this is what you want to do, use the GNU Lesser General
 Public License instead of this License.  But first, please read
 <https://www.gnu.org/licenses/why-not-lgpl.html>.
 Add info to tracked file
+Add more info to tracked file
diff --git a/ONE_ADDED_FILE.txt b/ONE_ADDED_FILE.txt
index e69de29..13d1090 100644
--- a/ONE_ADDED_FILE.txt
+++ b/ONE_ADDED_FILE.txt
@@ -0,0 +1 @@
+Add more info
```

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

GIT. ОСНОВЫ

Запись изменений в git репозиторий: Просмотр индексированных и неиндексированных изменений

Чтобы увидеть, что вы проиндексировали и что войдет в следующий коммит, наберите **git diff --staged** (**--cached** синоним для **--staged**):

```
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git diff --staged
diff --git a/LICENSE b/LICENSE
index f288702..6829b58 100644
--- a/LICENSE
+++ b/LICENSE
@@ -672,3 +672,4 @@ may consider it more useful to permit linking proprietary applications with
 the library.  If this is what you want to do, use the GNU Lesser General
 Public License instead of this License.  But first, please read
 <https://www.gnu.org/licenses/why-not-lgpl.html>.
+Add info to tracked file
diff --git a/NEW_FILE.md b/NEW_FILE.md
new file mode 100644
index 0000000..56266d3
--- /dev/null
+++ b/NEW_FILE.md
@@ -0,0 +1 @@
+My Project
diff --git a/ONE_ADDED_FILE.txt b/ONE_ADDED_FILE.txt
new file mode 100644
index 0000000..e69de29
diff --git a/README.md b/README.md
index af1af3d..271533b 100644
--- a/README.md
+++ b/README.md
@@ -1,2 +1,4 @@
 # git-course-github
 Репозиторий для курса GIT
+Modify file under source control
+Add info while file is indexed
```

GIT. ОСНОВЫ

Запись изменений в git репозиторий: Коммит изменений

Всё, что до сих пор не проиндексировано — любые файлы, созданные или изменённые вами, и для которых вы не выполнили **git add** после редактирования — не войдут в этот коммит!

Добавляем все оставшиеся файлы в индекс, выполнив команду **git add .**, где «.» означает текущий каталог, т. е. добавить в индекс все файлы начиная от текущего каталога. Убеждаемся что все проиндексировано: **git status -s**

```
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git add .
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git status -s
M  LICENSE
A  NEW_FILE.md
A  ONE_ADDED_FILE.txt
M  README.md
A  UNTRACKED_FILE.txt
```

Простейший способ зафиксировать изменения – это набрать команду **git commit**. Эта команда откроет выбранный вами текстовый редактор со следующим текстом:

```

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch main
# Your branch is up to date with 'origin/main'.
#
# Changes to be committed:
#       modified:   LICENSE
#       new file:   NEW_FILE.md
#       new file:   ONE_ADDED_FILE.txt
#       modified:   README.md
#       new file:   UNTRACKED_FILE.txt
#
```

Комментарий по умолчанию для коммита содержит закомментированный результат работы команды `git status` и ещё одну пустую строку сверху.

GIT. ОСНОВЫ

Запись изменений в git репозиторий: Коммит изменений

Когда вы выходите из редактора, Git создаёт для вас коммит с этим сообщением, удаляя комментарии

```
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git commit
[main 6d1790e] M comment for first commit This is second line for presentation.
 5 files changed, 6 insertions(+)
 create mode 100644 NEW_FILE.md
 create mode 100644 ONE_ADDED_FILE.txt
 create mode 100644 UNTRACKED_FILE.txt
```

Ух ты смотрите что у нас
Да. Это хэш сумма

Есть второй способ:
Вы можете набрать свой комментарий к коммиту в командной строке вместе с командой **git commit -m “<text_comment>”**:

Создадим пустой файл **FILE_FOR_COMMIT_M.txt**.
Добавим файл **FILE_FOR_COMMIT_M.txt** в индекс.
Сохраним изменения в локальную БД выполнив команду **git commit -m ’My first commit from command line’**:

```
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ touch FILE_FOR_COMMIT_M.txt
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git add FILE_FOR_COMMIT_M.txt
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git commit -m 'My first commit from command line'
[main aa90acc] My first commit from command line
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 FILE_FOR_COMMIT_M.txt
```

GIT. ОСНОВЫ

Запись изменений в git репозиторий: Игнорирование индексации

Индекс может быть удивительно полезным для создания коммитов именно такими, как вам и хотелось, но он временами несколько сложнее, чем может быть нужно нам в процессе работы.
Поэтому можно пропустить этап индексирования путем добавления параметра **-a** к команде **git commit**. Данный параметр заставляет Git автоматически индексировать каждый уже ***отслеживаемый на момент коммита файл***, позволяя обойтись без **git add**

Модифицируем файл **README.md**, который является отслеживаемым файлом.
Выполним команду **git commit -a -m 'Use parameter -a in command'**:

```
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ echo 'Parameter -a example' >> README.md
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git commit -a -m 'Use parameter -a in command'
[main 81e5013] Use parameter -a in command
 1 file changed, 1 insertion(+)
```

GIT. ОСНОВЫ

Запись изменений в git репозиторий: Удаление файлов

Для удаления файла из Git необходимо удалить его из отслеживаемых файлов (удалить из индекса), а затем выполнить коммит. Все это можно сделать с помощью команды **git rm <file-name or pattern>**, которая также ***удаляет файл из вашего рабочего каталога***.

Если же просто удалить файл из рабочего каталога, он будет показан в секции “Измененных, но не проиндексированных файлов” (Changes not staged for commit):

```
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ echo 'Should delete file' > FILE_SHOULD_DELETE.md
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git add .
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git commit -m 'Add file to delete'
[main 11b136b] Add file to delete
 1 file changed, 1 insertion(+)
 create mode 100644 FILE_SHOULD_DELETE.md
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ ls
FILE_FOR_COMMIT_M.txt  LICENSE      ONE_ADDED_FILE.txt  UNTRACKED_FILE.txt
FILE_SHOULD_DELETE.md  NEW_FILE.md  README.md
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ rm FILE_SHOULD_DELETE.md
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git status
On branch main
Your branch is ahead of 'origin/main' by 4 commits.
  (use "git push" to publish your local commits)

Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    FILE_SHOULD_DELETE.md

no changes added to commit (use "git add" and/or "git commit -a")
```

GIT. ОСНОВЫ

Запись изменений в git репозиторий: Удаление файлов

Если теперь выполнить команду **git rm <file-name or pattern>**, то удаление файла попадет в индекс:

```
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git rm FILE_SHOULD_DELETE.md
rm 'FILE_SHOULD_DELETE.md'
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git status
On branch main
Your branch is ahead of 'origin/main' by 4 commits.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    FILE_SHOULD_DELETE.md
```

Выполним коммит и данный файл больше не будет отслеживаться Git:

```
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git commit -m 'Deleted file'
[main e81aa8c] Deleted file
 1 file changed, 1 deletion(-)
 delete mode 100644 FILE_SHOULD_DELETE.md
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git status
On branch main
Your branch is ahead of 'origin/main' by 5 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ ls
FILE_FOR_COMMIT_M.txt  LICENSE  NEW_FILE.md  ONE_ADDED_FILE.txt  README.md  UNTRACKED_FILE.txt
```

GIT. ОСНОВЫ

Запись изменений в git репозиторий: Удаление файлов

Если вы попытаетесь удалить файл, который был изменен (не важно в индексе он находится или нет), то вы получите ошибку. Это сделано для большей безопасности: чтобы предотвратить удаление данных, которые еще не были записаны в снимок состояния и которые нельзя будет восстановить из Git.

Измененный, но **не проиндексированный** файл:

```
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ echo 'Some' >> FILE_FOR_COMMIT_M.txt
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git rm FILE_FOR_COMMIT_M.txt
error: the following file has local modifications:
    FILE_FOR_COMMIT_M.txt
(use --cached to keep the file, or -f to force removal)
```

Измененный и **проиндексированный** файл:

```
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ echo 'Modify' >> README.md
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git add README.md
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git rm README.md
error: the following file has changes staged in the index:
    README.md
(use --cached to keep the file, or -f to force removal)
```

GIT. ОСНОВЫ

Запись изменений в git репозиторий: Удаление файлов

Чтобы все таки выполнить удаление файла (если вы уверены, что так надо), то нужно просто к команде **git rm** добавить параметр **-f**.

```
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git rm -f FILE_FOR_COMMIT_M.txt README.md
rm 'FILE_FOR_COMMIT_M.txt'
rm 'README.md'
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git status -s
D  FILE_FOR_COMMIT_M.txt
D  NEW_FILE.md
D  README.md
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ ls
LICENSE  ONE_ADDED_FILE.txt  UNTRACKED_FILE.txt
```

Как мы видим, файл **FILE_FOR_COMMIT_M.txt** и **README.md** были удалены и эта операция была проиндексирована

Если нужно удалить файл только из индекса, но оставить его в рабочем каталоге, то к команде **git rm** необходимо добавить параметр **--cached**.

```
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ echo 'TEST' >> LICENSE
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git add LICENSE
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git rm --cached LICENSE
rm 'LICENSE'
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git status -s
D  FILE_FOR_COMMIT_M.txt
D  LICENSE
D  NEW_FILE.md
D  README.md
?? LICENSE
```

Файл **LICENSE** теперь помечен на удаление из индекса, а также видим, что он стал неотслеживаемым.

GIT. ОСНОВЫ

Запись изменений в git репозиторий: Перемещение файлов

Git не отслеживает перемещение файлов явно. Когда вы переименовываете файл, то в Git не сохраняется никаких метаданных об этом.
Однако Git все таки умеет обнаруживать перемещения постфактум.
Таким образом, наличие в Git команды `mv` выглядит несколько странным. Если вам хочется переименовать файл в Git, вы можете сделать что-то вроде:

```
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git mv ONE_ADDED_FILE.txt MOVED_ONE_ADDED_FILE
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git status
On branch main
Your branch is ahead of 'origin/main' by 5 commits.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    FILE_FOR_COMMIT_M.txt
        deleted:    LICENSE
        renamed:    ONE_ADDED_FILE.txt -> MOVED_ONE_ADDED_FILE
        deleted:    NEW_FILE.md
        deleted:    README.md

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        LICENSE
```

Это эквивалентно выполнению следующих команд:

```
$ mv ONE_ADDED_FILE.txt
MOVED_ONE_ADDED_FILE
```

Выполним коммит, чтобы все наши изменения были проиндексированы и удалим файл **LICENSE** с файловой системы

GIT. ОСНОВЫ

Просмотр истории коммитов

Мы уже создали несколько коммитов и можем посмотреть историю коммитов. Одним из основных и наиболее мощных инструментов для этого является команда **git log**. Выполним ее.

По умолчанию команда **git log**. Перечисляет коммиты в обратном хронологическом порядке: последние коммиты сверху.

```
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git log
commit c5f4e9b66e019a5b0af52e42b2489826a623bef9 (HEAD -> main)
Author: Koylubaev Nikita <koylubaevnt@gmail.com>
Date:   Sat Dec 3 11:14:12 2022 +0500

    Flush all changes

commit e81aa8c8f0d978bf705916cb6bdd7e2738b0dd44
Author: Koylubaev Nikita <koylubaevnt@gmail.com>
Date:   Sat Dec 3 10:33:27 2022 +0500

    Deleted file

commit 11b136ba1241c2bc754e57dd3a65c34970fe9e37
Author: Koylubaev Nikita <koylubaevnt@gmail.com>
Date:   Sat Dec 3 10:26:32 2022 +0500

    Add file to delete

commit 81e5013b7c5a9be4d1659a37a64c1be01eeec515
Author: Koylubaev Nikita <koylubaevnt@gmail.com>
Date:   Sat Dec 3 00:04:28 2022 +0500

    Use parameter -a in command

commit aa90acc68737cba23d10a3f01ca0851926d1bdce
Author: Koylubaev Nikita <koylubaevnt@gmail.com>
Date:   Fri Dec 2 23:46:44 2022 +0500

    My first commit from command line

commit 6d1790e84f439d0d9544175967be3c19dd2a4e1e
Author: Koylubaev Nikita <koylubaevnt@gmail.com>
Date:   Fri Dec 2 23:31:09 2022 +0500

    My comment for first commit
    This is second line for presentation.

commit 4014662fe97f8733c554cec8d03a41ea22b668ce (origin/main, origin/HEAD)
Author: Koylubaev Nikita <koylubaevnt@gmail.com>
Date:   Fri Dec 2 18:26:54 2022 +0500

    Initial commit
```

GIT. ОСНОВЫ

Просмотр истории коммитов

Одним из самых полезных аргументов является **-p** или **--patch**, который показывает разницу (выводит *патч*), внесенную в каждый коммит. Так же вы можете ограничить количество записей в выводе команды; используйте параметр **-2** для вывода только двух записей.

Если вы хотите увидеть сокращенную статистику для каждого коммита, вы можете использовать опцию **--stat**

Еще есть полезная опция **--pretty**. Эта опция меняет формат вывода. Существует несколько встроенных вариантов отображения. Опция **oneline** выводит каждый коммит в одну строку, что может быть очень удобным если вы просматриваете большое количество коммитов. К тому же, опции **short**, **full** и **fuller** делают вывод приблизительно в том же формате, но с меньшим или большим количеством информации соответственно:

```
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git log --pretty=oneline
c5f4e9b66e019a5b0af52e42b2489826a623bef9 (HEAD -> main) Flush all changes
e81aa8c8f0d978bf705916cb6bdd7e2738b0dd44 Deleted file
11b136ba1241c2bc754e57dd3a65c34970fe9e37 Add file to delete
81e5013b7c5a9be4d1659a37a64c1be01eeec515 Use parameter -a in command
aa90acc68737cba23d10a3f01ca0851926d1bdce My first commit from command line
6d1790e84f439d0d9544175967be3c19dd2a4e1e My comment for first commit This is second line for presentation.
4014662fe97f8733c554cec8d03a41ea22b668ce (origin/main, origin/HEAD) Initial commit
```

Опции **oneline** и **format** являются особенно полезными с опцией **--graph** команды **log**. Показывает небольшой граф в формате ASCII.

```
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git log --pretty=oneline --graph
*   abbb6b3e9b3e7f98d3578caa541707a89e8ef41b (HEAD -> main) Merge branch 'test'
|\
| * 48d00a98dbfcfedf6afc89759fa66cc4f1308a29 (test) branch
* | 30ae240382547b9c1a218b7dc03d1bab72308a32 branch
|/
* c5f4e9b66e019a5b0af52e42b2489826a623bef9 Flush all changes
* e81aa8c8f0d978bf705916cb6bdd7e2738b0dd44 Deleted file
* 11b136ba1241c2bc754e57dd3a65c34970fe9e37 Add file to delete
* 81e5013b7c5a9be4d1659a37a64c1be01eeec515 Use parameter -a in command
* aa90acc68737cba23d10a3f01ca0851926d1bdce My first commit from command line
* 6d1790e84f439d0d9544175967be3c19dd2a4e1e My comment for first commit This is second line for presentation.
* 4014662fe97f8733c554cec8d03a41ea22b668ce (origin/main, origin/HEAD) Initial commit
```

GIT. ОСНОВЫ

Просмотр истории коммитов

Наиболее интересной опцией является **format**, которая позволяет указать формат для вывода информации. Особенно это может быть полезным когда вы хотите сгенерировать вывод для автоматического анализа – так как вы указываете формат явно, он не будет изменен даже после обновления Git:

$ git log --pretty=format:"%h - %an, %ar : %s"

```
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git log --pretty=format:"%h - %an, %ar : %s"
c5f4e9b - Koylubaev Nikita, 28 minutes ago : Flush all changes
e81aa8c - Koylubaev Nikita, 69 minutes ago : Deleted file
11b136b - Koylubaev Nikita, 76 minutes ago : Add file to delete
81e5013 - Koylubaev Nikita, 12 hours ago : Use parameter -a in command
aa90acc - Koylubaev Nikita, 12 hours ago : My first commit from command line
6d1790e - Koylubaev Nikita, 12 hours ago : My comment for first commit This is second line for presentation.
4014662 - Koylubaev Nikita, 17 hours ago : Initial commit
```

Опция	Описания вывода
%H	Хеш коммита
%h	Сокращенный хеш коммита
%T	Хеш дерева
%t	Сокращенный хеш дерева
%P	Хеш родителей
%p	Сокращенный хеш родителей
%an	Имя автора

Опция	Описания вывода
%ae	Электронная почта автора
%ad	Дата автора (формат даты можно задать опцией --date=option)
%ar	Относительная дата автора
%cn	Имя коммитера
%ce	Электронная почта коммитера
%cd	Дата коммитера
%cr	Относительная дата коммитера
%s	Содержание

GIT. ОСНОВЫ

Просмотр истории коммитов: Ограничение вывода

Команда **git log** принимает несколько опций для ограничения вывода информации. Одну уже вилели **-2**, которая показывает только последние 2 коммита. **-n** — это любое натуральное число и представляет собой **n** последних коммитов

Есть опции для ограничения вывода по времени: такие как **--since** и **--until**. Команда покажет список коммитов, сделанных за последние 2 недели:

$ git log –since=2.weeks

Можно фильтровать список коммитов по заданным параметрам:

--author – по автору коммита

--committer – по коммитеру,

--grep – позволяет искать по ключевым словам в сообщении коммита.

При этом можно указывать несколько данных параметров, тогда найдутся коммиты соотвествующие *любому* указанному шаблону. Но применение опции **--all-match** заставит искать коммиты соответствующие всем **--grep** шаблона

Опция	Описание
-(n)	Показывает только последние n коммитов.
--since, --after	Показывает только те коммиты, которые были сделаны после указанной даты.
--until, --before	Показывает только те коммиты, которые были сделаны до указанной даты.
--author	Показывает только те коммиты, в которых запись author совпадает с указанной строкой.
--committer	Показывает только те коммиты, в которых запись committer совпадает с указанной строкой.
--grep	Показывает только коммиты, сообщение которых содержит указанную строку.
-S	Показывает только коммиты, в которых изменение в коде повлекло за собой добавление или удаление указанной строки

GIT. ОСНОВЫ

Операции отмены

В любой момент может потребоваться что-либо отменить. **Будьте осторожны, не все операции отмены в свою очередь можно отменить!** Это одна из редких областей Git, где неверными действиями можно необратимо удалить результаты своей работы.

Отмена может потребоваться, если вы сделали коммит слишком рано, например, забыв добавить какие-то файлы или комментарий к коммиту. Если вы хотите переделать коммит — внесите необходимые изменения, добавьте их в индекс и сделайте коммит ещё раз, указав параметр **--amend**:

```
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ echo 'Some data' > NEW_FILE_1.txt
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ echo 'Some data' > NEW_FILE_2.txt
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git add NEW_FILE_1.txt
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git commit -m 'Forgot add file'
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git add NEW_FILE_2.txt
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git commit --amend
[main a7332dc] Add forgotten file Forgot add file
 Date: Sat Dec 3 12:17:38 2022 +0500
 2 files changed, 2 insertions(+)
 create mode 100644 NEW_FILE_1.txt
 create mode 100644 NEW_FILE_2.txt
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git log --pretty=oneline -4
a7332dc0972fefa3e7e0b62165c1f9629ebf734a (HEAD -> main) Add forgotten file Forgot add file
c5f4e9b66e019a5b0af52e42b2489826a623bef9 Flush all changes
e81aa8c8f0d978bf705916cb6bdd7e2738b0dd44 Deleted file
11b136ba1241c2bc754e57dd3a65c34970fe9e37 Add file to delete
```

В итоге получится единый коммит — второй коммит заменит результаты первого.

В итоге получился единый коммит — второй коммит заменил результаты первого.

GIT. ОСНОВЫ

Операции отмены

```
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git commit --amend -m 'Update message in last commit'
[main c8faa6c] Update message in last commit
 Date: Sat Dec 3 12:17:38 2022 +0500
 2 files changed, 2 insertions(+)
 create mode 100644 NEW_FILE_1.txt
 create mode 100644 NEW_FILE_2.txt
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git log --pretty=oneline -4
c8faa6cfa6f5b025e1caafb3d5347f7c9055ad2a (HEAD -> main) Update message in last commit
c5f4e9b66e019a5b0af52e42b2489826a623bef9 Flush all changes
e81aa8c8f0d978bf705916cb6bdd7e2738b0dd44 Deleted file
11b136ba1241c2bc754e57dd3a65c34970fe9e37 Add file to delete
```

Данной командой мы просто изменили сообщение последнего коммита, т.к. мы ничего не меняли с момента последнего коммита.

Если быть точнее, то при таком подходе последний коммит будет заменен новым!

```
a7332dc0972fefa3e7e0b62165c1f9629ebf734a (HEAD -> main) Add forgotten file Forgot add file
```

V

S

```
c8faa6cfa6f5b025e1caafb3d5347f7c9055ad2a (HEAD -> main) Update message in last commit
```

Смысл изменения коммитов в добавлении незначительных правок в последние коммиты и, при этом, в избежании засорения истории сообщениями вида «Ой, забыл добавить файл» или «Исправление грамматической ошибки»

GIT. ОСНОВЫ

Операции отмены. Отмена индексации файла

```
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ touch FILE_1.txt
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ touch FILE_2.txt
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git add *
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git commit -m 'For reset'
[main 29e22f2] For reset
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 FILE_1.txt
 create mode 100644 FILE_2.txt
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ echo 'Data' >> FILE_1.txt
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ echo 'Text' >> FILE_2.txt
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git add *
```

Например, вы изменили два файла и хотите добавить их в разные коммиты, но случайно выполнили команду **git add *** и добавили в индекс оба. Как исключить из индекса один из них?

GIT. ОСНОВЫ

Операции отмены. Отмена индексации файла

Последуем этому совету и выполним команду: **git restore --staged <file-name or patte**

```
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git restore --staged FILE_2.txt
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git status
On branch main
Your branch is ahead of 'origin/main' by 9 commits.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   FILE_1.txt

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   FILE_2.txt
```

Файл **FILE_2.txt** изменен, но больше не в индексе

В более старых версиях Git (до 2.23.0) для этой операции использовалась такая команда:

git reset HEAD <file-

GIT. ОСНОВЫ

Операции отмены: Отмена изменений в файле

Что если вы поняли что не хотите сохранять изменения в файле **FILE_2.txt**?
Чтобы его откатить к тому виду, как он выглядел при последнем коммите нам опять подскажет команда **git status** и это будет:
git restore <file-name or pattern>

```
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git status
On branch main
Your branch is ahead of 'origin/main' by 9 commits.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   FILE_1.txt

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   FILE_2.txt

koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git restore FILE_2.txt
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git status
On branch main
Your branch is ahead of 'origin/main' by 9 commits.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   FILE_1.txt
```

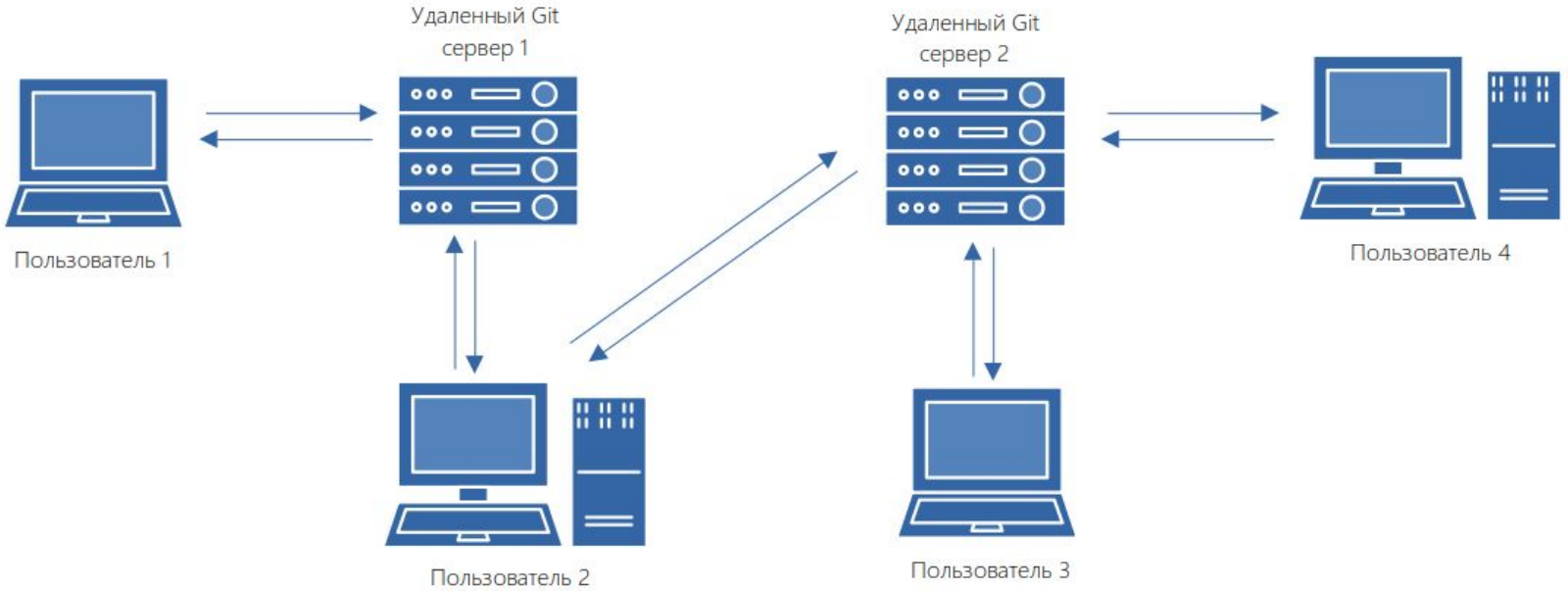
Важно понимать, что **git restore <file>** — опасная команда.
Любые локальные изменения, внесенные в этот файл, исчезнут — Git просто заменит файл последней зафиксированной версией.
Никогда не используйте эту команду, если точно не знаете, нужны ли вам эти несохраненные локальные изменения.

В более старых версиях Git (до 2.23.0) для этой операции использовалась такая команда:
git checkout -- <file-name or pattern>

GIT. ОСНОВЫ

Работа с удалёнными репозиториями

Удалённые репозитории представляют собой версии вашего проекта, сохранённые в интернете или ещё где-то в сети.
У вас может быть несколько удалённых репозиториев, каждый из которых может быть доступен для чтения или для чтения-записи.
Взаимодействие с другими пользователями предполагает управление удалёнными репозиториями, а также отправку и по



GIT. ОСНОВЫ

Работа с удалёнными репозиториями: Просмотр удаленных репозиториев

Для того, чтобы просмотреть список настроенных удалённых репозиториев, вы можете запустить команду **git remote**.

```
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git remote
origin
```

Т.к. мы клонировали удаленный репозиторий, то увидели **origin** – это имя по умолчанию, которое Git дает серверу, с которого произошло клонирование

Если в команду **git remote** добавить ключ **-v**, то мы увидем адреса для чтения и записи, которые привязаны к удаленному репозиторию

```
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git remote -v
origin  https://github.com/koylubaevnt/git-course-github.git (fetch)
origin  https://github.com/koylubaevnt/git-course-github.git (push)
```

GIT. ОСНОВЫ

Работа с удалёнными: Добавление

Для того чтобы добавить удаленный репозиторий и присвоить ему имя (*shortname*), просто надо выполнить комаду: **git remote add <shortname> <url>**.

```
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git remote add mirror http://192.168.0.233/git/remote-repo.git
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git remote -v
mirror  http://192.168.0.233/git/remote-repo.git (fetch)
mirror  http://192.168.0.233/git/remote-repo.git (push)
origin  https://github.com/koylubaevnt/git-course-github.git (fetch)
origin  https://github.com/koylubaevnt/git-course-github.git (push)
```

GIT. ОСНОВЫ

Работа с удалёнными: Получение изменений

Для получения данных с удаленного репозитория, следует выполнить команду **git fetch**

```
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git fetch mirror
warning: no common commits
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 248 bytes | 248.00 KiB/s, done.
From http://192.168.0.233/git/remote-repo
 * [new branch]      develop    -> mirror/develop
 * [new branch]      master     -> mirror/master
```

Ветка develop из репозитория mirror доступна под именем mirror/develop и т.д.

Данная команда связывается с указанным удалённым проектом и забирает все те данные проекта, которых у вас ещё нет. После того как команда выполнена, то в нашем локлаьном репозитории появятся ссылки на все ветки из этого удалённого проекта, которые можно будет просмотреть или слить в любой момент.
Когда мы клонировали репозиторий, команда **clone** автоматически добавляет этот удалённый репозиторий под именем «**origin**». Таким образом, **git fetch origin** извлекает все наработки, отправленные на этот сервер после того, как вы его клонировали (или получили изменения с помощью fetch).

Важно отметить, что команда **git fetch** забирает данные в ваш локальный репозиторий, но не сливает их с какими-либо вашими наработками и не модифицирует то, над чем вы работаете в данный момент. Вам необходимо будет вручную слить эти данные с вашими, когда вы будете готовы.

GIT. ОСНОВЫ

Работа с удалёнными: Получение изменений

Если ветка настроена на отслеживание удалённой ветки (про это будет расказано позже), то вы можете использовать команду **git pull** чтобы автоматически получить изменения из удалённой ветки и слить их со своей текущей.
Этот способ может для вас оказаться более простым или более удобным.
К тому же, по умолчанию команда **git clone** автоматически настраивает вашу локальную ветку **master** на отслеживание удалённой ветки **master** на сервере, с которого вы клонировали репозиторий. Название веток может быть другим и зависит от ветки по умолчанию на сервере.
Выполнение **git pull**, как правило, извлекает (**fetch**) данные с сервера, с которого вы изначально клонировали, и автоматически пытается слить (**merge**)[про это также будет рассказано позже] их с кодом, над которым вы в данный момент работаете.

GIT. ОСНОВЫ

Работа с удалёнными: Отправка изменений

Для того, чтобы поделиться своими наработками, необходимо их отправить в удаленный репозиторий. Команда для этого действия:
git push <remote-repository-name> <branch-name>

```
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git push mirror main
Enumerating objects: 30, done.
Counting objects: 100% (30/30), done.
Compressing objects: 100% (24/24), done.
Writing objects: 100% (30/30), 15.04 KiB | 3.01 MiB/s, done.
Total 30 (delta 9), reused 3 (delta 0), pack-reused 0
To http://192.168.0.233/git/remote-repo.git
 * [new branch]      main -> main
```

Эта команда срабатывает только в случае, если вы клонировали с сервера, на котором у вас есть права на запись, и если никто другой с тех пор не выполнял команду **push**. Если вы и кто-то ещё одновременно клонируете, затем он выполняет команду **push**, а после него выполнить команду **push** попытаетесь вы, то ваш **push** будет отклонён. Вам придётся сначала получить изменения и объединить их с вашими и только после этого вам будет позволено выполнить **push**.

GIT. ОСНОВЫ

Работа с удалёнными: Получение подробной информации

Если хотите получить побольше информации об одном из удалённых репозиториев, вы можете использовать команду: **git remote show <remote>**.
Выполнив эту команду с некоторым именем, например, **mirror**, вы получите следующий результат:

```
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git remote show mirror
* remote mirror
  Fetch URL: http://192.168.0.233/git/remote-repo.git
  Push  URL: http://192.168.0.233/git/remote-repo.git
  HEAD branch: master
  Remote branches:
    develop tracked
    main    tracked
    master  tracked
  Local ref configured for 'git push':
    main pushes to main (up to date)
```

Эта команда выдала URL удалённого репозитория, а также информацию об отслеживаемых ветках. Эта команда любезно выдаёт список всех полученных ею ссылок.

GIT. ОСНОВЫ

Работа с удалёнными: Удаление и переименование удалённых репозиториев

Для переименования удалённого репозитория можно выполнить:
git remote rename <old-remote-repository-name> <new-remote-repository-name>.

```
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git remote rename mirror second
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git remote
origin
second
```

Это также изменит имена удалённых веток в вашем репозитории. То, к чему вы обращались как **mirror/master**, теперь стало **second/master**

Если по какой-то причине вы хотите удалить удаленный репозиторий — вы сменили сервер или больше не используете определённое зеркало, или кто-то перестал вносить изменения — вы можете использовать команду:
git remote rm <remote-repository-name>

```
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git remote remove second
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git remote
origin
```

При удалении ссылки на удалённый репозиторий все отслеживаемые ветки и настройки, связанные с этим репозиторием, так же будут удалены

GIT. ОСНОВЫ

Работа с тэгами: Просмотр списка тегов

Git имеет возможность помечать определённые моменты в истории как важные. Как правило, эта функциональность используется для отметки моментов выпуска версий (v1.0, и т. п.). Такие пометки в Git называются тегами.

Просмотреть список имеющихся тегов в Git можно очень просто. Для этого используется команда
git tag [-l].
Данная команда перечисляет теги в алфавитном порядке; порядок их отображения не имеет существенного значения.

```
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git tag
v0.0.1-annotation
v0.0.1-light-weight
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git tag -l '*-annotation'
v0.0.1-annotation
```

С помощью команды **git show <tag-name>** вы можете посмотреть данные тега вместе с коммитом.

```
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git show v0.0.1-light-weight
commit 29e22f2f60ed9f75ef7605bdb27068b7552e6151 (HEAD -> main, tag: v0.0.1-light-weight, tag: v0.0.1-annotation, issue/fix)
Author: Koylubaev Nikita <koylubaevnt@gmail.com>
Date:   Sat Dec 3 21:41:19 2022 +0500

    For reset

diff --git a/FILE_1.txt b/FILE_1.txt
new file mode 100644
index 0000000..e69de29
diff --git a/FILE_2.txt b/FILE_2.txt
new file mode 100644
index 0000000..e69de29
```

GIT. ОСНОВЫ

Работа с тэгами: Создание тегов

Git использует два основных типа тегов: легковесные и аннотированные.

Аннотированный тег – хранится в базе данных Git, как полноценный объект. Имеет контрольную сумму, содержит имя автора, его email и дату создания, имеет комментарий и может быть подписан и проверен с помощью GNU Privacy Guard (GPG). Рекомендуется использовать этот вид тэга.

Создание аннотированного тега в Git выполняется легко: **git tag -a <tag-name>**

```
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git tag -a v0.0.1-annotation -m 'Tag message'
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git tag
v0.0.1-annotation
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git show v0.0.1-annotation
tag v0.0.1-annotation
Tagger: Koylubaev Nikita <koylubaevnt@gmail.com>
Date:   Mon Dec 5 11:59:41 2022 +0500

Tag message

commit 29e22f2f60ed9f75ef7605bdb27068b7552e6151 (HEAD -> main, tag: v0.0.1-annotation, issue/fix)
Author: Koylubaev Nikita <koylubaevnt@gmail.com>
Date:   Sat Dec 3 21:41:19 2022 +0500

    For reset

diff --git a/FILE_1.txt b/FILE_1.txt
new file mode 100644
index 0000000..e69de29
diff --git a/FILE_2.txt b/FILE_2.txt
new file mode 100644
index 0000000..e69de29
```

Опция **-m** задаёт сообщение, которое будет храниться вместе с тегом. Если не указать сообщение, то Git запустит редактор, чтобы вы смогли его ввести

GIT. ОСНОВЫ

Работа с тэгами: Создание тегов

Легковесный тег – это что-то очень похожее на ветку, которая не изменяется – просто указатель на определённый коммит. По сути, это контрольная сумма коммита, сохранённая в файл – больше никакой информации не хранится.
Для создания легковесного тэга используется та же команда что и для анотированного, просто не надо передавать ключи `-a`, `-s` и `-m`.

```
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git show v0.0.1-light-weight
commit 29e22f2f60ed9f75ef7605bdb27068b7552e6151 (HEAD -> main, tag: v0.0.1-light-weight, tag: v0.0.1-annotation, issue/fix)
Author: Koylubaev Nikita <koylubaevnt@gmail.com>
Date:   Sat Dec 3 21:41:19 2022 +0500

    For reset

diff --git a/FILE_1.txt b/FILE_1.txt
new file mode 100644
index 0000000..e69de29
diff --git a/FILE_2.txt b/FILE_2.txt
new file mode 100644
index 0000000..e69de29
```

Нет дополнительной информации, что была у аннотированного тэга: **Tagger, Date, Message**...

GIT. ОСНОВЫ

Работа с тэгами: Отложенная расстановка тегов

Также возможно помечать уже пройденные коммиты. Предположим, история коммитов выглядит следующим образом

```
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git log --pretty=oneline
29e22f2f60ed9f75ef7605bdb27068b7552e6151 (HEAD -> main, tag: v0.0.1-light-weight, tag: v0.0.1-annotation, issue/fix) For reset
149e907b60145aec5d93b862d8dc21582555636d For reset
c8faa6cfa6f5b025e1caafb3d5347f7c9055ad2a Update message in last commit
c5f4e9b66e019a5b0af52e42b2489826a623bef9 Flush all changes
e81aa8c8f0d978bf705916cb6bdd7e2738b0dd44 Deleted file
11b136ba1241c2bc754e57dd3a65c34970fe9e37 Add file to delete
81e5013b7c5a9be4d1659a37a64c1be01eeec515 Use parameter -a in command
aa90acc68737cba23d10a3f01ca0851926d1bdce My first commit from command line
6d1790e84f439d0d9544175967be3c19dd2a4e1e My comment for first commit This is second line for presentation.
4014662fe97f8733c554cec8d03a41ea22b668ce (origin/main, origin/HEAD) Initial commit
```

Теперь предположим, что вы забыли отметить версию проекта v0.0.1-alpha, которая была там, где находится коммит «Add file to delete». Вы можете добавить тег и позже. Для отметки коммита надо просто указать его контрольную сумму (или её часть) как параметр команды **git tag**.

```
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git tag -a v0.0.1-alpha 11b136b -m 'v0.0.1-alpha commit'
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git tag
v0.0.1-alpha
v0.0.1-annotation
v0.0.1-light-weight
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git show v0.0.1-alpha
tag v0.0.1-alpha
Tagger: Koylubaev Nikita <koylubaevnt@gmail.com>
Date:   Mon Dec 5 12:12:15 2022 +0500

v0.0.1-alpha commit

commit 11b136ba1241c2bc754e57dd3a65c34970fe9e37 (tag: v0.0.1-alpha)
Author: Koylubaev Nikita <koylubaevnt@gmail.com>
Date:   Sat Dec 3 10:26:32 2022 +0500

    Add file to delete

diff --git a/FILE_SHOULD_DELETE.md b/FILE_SHOULD_DELETE.md
new file mode 100644
index 0000000..168b4b0
--- /dev/null
+++ b/FILE_SHOULD_DELETE.md
@@ -0,0 +1 @@
+Should delete file
```

GIT. ОСНОВЫ

Работа с тэгами: Обмен тегами

По умолчанию, команда **git push** не отправляет теги на удалённые сервера. После создания тегов, их нужно отправлять явно на удалённый сервер. Для этого надо выполнить команду:
git push [remote-repository-name] <tag-name>.

```
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git push mirror v0.0.1-light-weight
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To http://192.168.0.233/git/remote-repo.git
 * [new tag]         v0.0.1-light-weight -> v0.0.1-light-weight
```

Если у вас много тегов, и вам хотелось бы отправить все за один раз, то можно использовать команду **git push [remote-repository-name] --tags**

```
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git push mirror --tags
Enumerating objects: 2, done.
Counting objects: 100% (2/2), done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 223 bytes | 223.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0), pack-reused 0
To http://192.168.0.233/git/remote-repo.git
 * [new tag]         simple--tag -> simple--tag
 * [new tag]         test-tag -> test-tag
 * [new tag]         v0.0.1-annotation -> v0.0.1-annotation
```

GIT. ОСНОВЫ

Работа с тэгами: Удаление тегов

Для удаления тега в локальном репозитории достаточно выполнить команду **git tag -d <tag-name>**

```
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git tag -d simple--tag
Deleted tag 'simple--tag' (was 29e22f2)
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git tag
test-tag
v0.0.1-alpha
v0.0.1-annotation
v0.0.1-light-weight
```

При удалении тега, удаление с внешних серверов не происходит!

Существует 2 способа изъятия тэга из удаленного репозитория:

1. **git push [remote-repository-name] :refs/tags/<tag-name>**
2. **git push [remote-repository-name] --delete <tag-name>**

```
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git push mirror :refs/tags/simple--tag
To http://192.168.0.233/git/remote-repo.git
 - [deleted]         simple--tag
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git push mirror --delete test-tag
To http://192.168.0.233/git/remote-repo.git
 - [deleted]         test-tag
```

GIT. ОСНОВЫ

Работа с тэгами: Переход на тег

Если вы хотите получить версии файлов, на которые указывает тег, то вы можете сделать **git checkout <tag-name>** для тега. Однако, это переведёт репозиторий в состояние «**detached HEAD**», которое имеет ряд неприятных побочных эффектов.

```
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git checkout v0.0.1-alpha
Note: switching to 'v0.0.1-alpha'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at 11b136b Add file to delete
```

Если все же нужно внести изменения, то это делается только через создание ветки. Этот процесс будет расмотрен в другой раз.

В состоянии «detached HEAD» изменения и коммиты не будут относиться ни к какой из веток и доступ к ним можно будет получить только по их хешам!!!

GIT. ОСНОВЫ

Псевдонимы в Git

Рассмотрим ещё одну маленькую хитрость, которая поможет сделать использование Git проще, легче, и более привычным: **псевдонимы** (aliases).
Если вы не хотите печатать каждую команду для Git целиком, вы легко можете настроить **псевдонимы (alias)** для любой команды с помощью **git config –global alias.<alias-name> <command>**. Вот несколько примеров псевдонимов, которые вы, возможно, захотите задать:
$ git config --global alias.ci commit
$ git config --global alias.st status
Это означает, что, например, вместо ввода **git status**, вам достаточно набрать только **git st**.

```
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git st
git: 'st' is not a git command. See 'git --help'.

The most similar commands are
        status
        reset
        s
        stage
        stash
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git config --global alias.st status
koylubaevnt@koylubaevnt-VirtualBox:~/Project/git-course-github$ git st
HEAD detached at v0.0.1-alpha
nothing to commit, working tree clean
```

GIT. ОСНОВЫ

Вопросы\Замечания

Мы рассмотрели все базовые локальные операции с Git: создавать или клонировать репозиторий, вносить изменения, индексировать и фиксировать эти изменения, а также просматривать историю всех изменений в репозитории.

ВОПРОСЫ ЗАМЕЧАНИЯ

Спасибо за внимание!