https://practicum.yandex.ru/trainer/git-basics/lesson/c85a3c25-ccb9-4d89-80a9-0ce6b78ae347/

Тема 1. Установка Git

### **Для Windows**

https://git-scm.com/download/win

### **Для Linux**

<https://git-scm.com/download/linux>

### **Проверка**

$ git version

Тема 2. Командная строка

**Навигация в командной строке**

### **Вывести путь к текущей директории – pwd (print working directory)**

$ pwd

Поможет узнать, где вы сейчас.

### **Сменить директорию – cd (change directory)**

$ cd github # переходим в папку github

$ cd github/projects/first # переходим в first через несколько директорий

Если в названии папки есть пробелы, при вводе нужно использовать кавычки

$ cd “github projects” # переходим в папку github projects

$ cd .. # переходим на уровень выше (в родительскую   
# директорию

По аналогии с .. текущая папка обозначается одной точкой ./

### **Вывести содержимое директории — ls (list directory contents)**

$ ls

$ ll # то же в подробном формате

$ ls –a # выведет все папки и файлы, в т.ч. скрытые,   
#которые начинаются с точки.

$ ls ~ # выведет содержимое домашней директории

$ ls .. # выведет содержимое родительской директории

**Операции с папками и файлами:**

### **Создание файла**

$ touch myfile1.txt myfile2.txt # создаст новые файлы " myfile1.txt " и   
# "myfile2.txt" на диске

### **Создание директории**

$ mkdir mynewfolder # создаст папку " mynewfolder" на диске

Можно создать целую структуру директорий одной командой с помощью флага -p.

$ mkdir -p dir1/dir-inside/dir-deeper-inside # создали папку dir-deeper-inside в папке dir-inside,  
# которая находится в папке dir1

### **Копирование файлов**

$ cp что\_копируем куда\_копируем

$ cp index.html style.css src/ # скопировали index.html и style.css в папку src

$ cp index.html ~ # скопировали index.html в домашнюю   
 # директорию

### **Перемещение файлов и папок**

$ mv что\_перемещаем куда\_перемещаем

$ mv data.csv very-important-files # переместили data.csv в директорию   
# very-important-files/

### **Чтение файлов**

$ cat myfile.txt # выведет содержимое файла на экран

file-content-1

file-content-2

### **Удаление файлов**

**Обратите внимание, что удаление файлов и директорий необратимо. Перед использованием команды rm убедитесь, что вы знаете, какие файлы и директории нужно удалить, и что они больше не нужны.**

$ rm example.txt # удалили файл example.txt из текущей папки

### **Удаление папок**

**Обратите внимание, что удаление файлов и директорий необратимо. Перед использованием команды rm убедитесь, что вы знаете, какие файлы и директории нужно удалить, и что они больше не нужны.**

$ rmdir images # команда удалит папку images из текущей   
# директории, **если папка images пуста**

Если в папке есть какие-то файлы, то командная строка не удалит её и выведет сообщение Directory not empty.

Если папку всё-таки нужно удалить вместе со всем её содержимым, можно использовать команду

$ rm -r images # рекурсивно удалит папку images со всем её   
# содержимым из текущей директории

**Эффективная работа с командной строкой**

### **Выполняйте сразу несколько команд с помощью &&**

$ mkdir second-project **&&** cd second-project **&&** touch index.html style.css # создаём папку second-  
# project, переходим в   
# папку second-project и   
# создаём в ней два   
# файла: index.html и   
# style.css

### **Вызывайте команды из буфера**

По их списку можно перемещаться: нажимайте на клавиатуре стрелку вверх (↑), стрелку вниз (↓).

### **Используйте автозаполнение**

Если набрать первые буквы команды в командной строке и **дважды** нажать клавишу **Tab**, терминал покажет список всех команд, которые начинаются с этих символов.

**Tab** автоматически дописывает не только команды, но и пути. Нажмите один раз для автозаполнения или 2 раза для вывода списка путей, совпадающих по первым буквам:

$ cd ~/ **Tab**

### **Применяйте команды для быстрой навигации**

* pwd — проверить, где мы находимся;
* ls — посмотреть список файлов/папок в директории;
* cd — перейти в выбранную папку.

С помощью этих команд можно быстро перемещаться между каталогами и изучать их содержимое.

### **Быстрый просмотр ключевых папок**

$ / **Tab Tab** # просмотр содержимого корневой папки Linux

$ cd c:/ или cd /c && ls # просмотр корневой папки диска в Windows

$ ~/ **Tab Tab** # просмотр содержимого домашней папки

|  |
| --- |
| Шпаргалка. Базовые команды в консоли Чтобы вам было удобнее взаимодействовать с командной строкой, мы подготовили шпаргалку. В ней собраны все команды, о которых мы рассказали в уроках, и их полезные вариации. Навигация  * pwd (от англ. ***p***rint ***w***orking ***d***irectory, «показать рабочую папку») — покажи, в какой я папке; * ls (от англ. ***l***i***s***t directory contents, «отобразить содержимое директории») — покажи файлы и папки в текущей папке; * ls -a — покажи также скрытые файлы и папки, названия которых начинаются с символа .; * cd first-project (от англ. ***c***hange ***d***irectory, «сменить директорию») — перейди в папку first-project; * cd first-project/html — перейди в папку html, которая находится в папке first-project; * cd .. — перейди на уровень выше, в родительскую папку; * cd ~ — перейди в домашнюю директорию (/Users/Username); * cd / — перейди в корневую директорию.  Работа с файлами и папками **Создание**   * touch index.html (англ. touch, «коснуться») — создай файл index.html в текущей папке; * touch index.html style.css script.js — если нужно создать сразу несколько файлов, можно напечатать их имена в одну строку через пробел; * mkdir second-project (от англ. ***m***a***k***e ***dir***ectory, «создать директорию») — создай папку с именем second-project в текущей папке.   **Копирование и перемещение**   * cp file.txt ~/my-dir (от англ. ***c***o***p***y, «копировать») — скопируй файл в другое место; * mv file.txt ~/my-dir (от англ. ***m***o***v***e, «переместить») — перемести файл или папку в другое место.   **Чтение**   * cat file.txt (от англ. con***cat***enate and print, «объединить и распечатать») — распечатай содержимое текстового файла file.txt.   **Удаление**   * rm about.html (от англ. ***r***e***m***ove, «удалить») — удали файл about.html; * rmdir images (от англ. ***r***e***m***ove ***dir***ectory, «удалить директорию») — удали папку images; * rm -r second-project (от англ. ***r***e***m***ove, «удалить» + ***r***ecursive, «рекурсивный») — удали папку second-project и всё, что она содержит.  Полезные возможности  * Команды необязательно печатать и выполнять по очереди. Можно указать их списком — разделить двумя амперсандами (&&). * У консоли есть собственная память — буфер с несколькими последними командами. По ним можно перемещаться с помощью клавиш со стрелками вверх (**↑**) и вниз (**↓**). * Чтобы не вводить название файла или папки полностью, можно набрать первые символы имени и дважды нажать Tab. Если файл или папка есть в текущей директории, командная строка допишет путь сама.   Например, вы находитесь в папке dev. Начните вводить cd first и дважды нажмите Tab. Если папка first-project есть внутри dev, командная строка автоматически подставит её имя. Останется только нажать Enter. |

Тема 4. Настройка Git

**Работа с файлом настройки .gitconfig**

Сделать это можно с помощью команды git config в терминале linux или Git bash в Windows, в любой директории.

$ git config --global user.name "UserName"

# имя или ник нужно написать латиницей и в кавычках

$ git config --global user.email username@yandex.ru

# здесь нужно указать свой настоящий email

Все глобальные настройки Git хранит в файле .gitconfig в домашней директории:

Просмотр файла:

$ cat ~/.gitconfig

То же:

$ git config --list

**Инициализируем репозиторий**

### **Сделать папку репозиторием — git init**

Чтобы Git начал отслеживать изменения в проекте, папку с файлами этого проекта нужно сделать **Git-репозиторием** (от англ. *repository* — «хранилище»). Для этого следует переместиться в неё и ввести команду git init

$ cd ~/first-project # перешли в нужную папку

$ git init # создали репозиторий

**Помните:**

не рекомендуется создавать репозиторий Git внутри другого Git-репозитория. Это может вызывать проблемы с отслеживанием изменений.

В данной папке создаётся папка /.git/ со служебной информацией.

### **«Разгитить» папку, если что-то пошло не так**

Если вы ошиблись и случайно инициализировали не ту папку, можно «разгитить» её — удалить папку /.git/ :

$ rm -rf .git

(-rf – удалить рекурсивно и принудительно (force, без подтверждающих вопросов)).

Вся история проекта будет стёрта без возможности восстановления — останется только последняя версия файлов.

### **Проверить состояние репозитория — git status**

$ git status # показывает текущее состояние репозитория.

выведет:

* название текущей ветки: On branch master или On branch main;
* сообщение о том, что в репозитории ещё нет коммитов: No commits yet;
* сообщение, которое говорит: «чтобы что-нибудь закоммитить (то есть зафиксировать), нужно сначала это создать» — nothing to commit (create/copy files and use "git add" to track).

**Добавляем файлы в репозиторий**

До добавления в репозиторий Git файлы папки имеют состояние «untracked» (неотслеживаемые).

$ git add filename1.txt filename2.txt # добавляет для отслеживания два файла

$ git add --all # добавляет для отслеживания все файлы

$ git add . # добавляет для отслеживания текущую папку

По команде git status зелёные файлы отслеживаются и готовы к сохранению. Состояние становится «Changes to be committed».

Команда git add не сохраняет содержимое файлов в репозитории. Само сохранение, или фиксацию состояния файлов, называют **коммитом** (от англ. *commit* — «совершать», «фиксировать»). «Сделать коммит» значит сохранить текущую версию файла.

После изменения отслеживаемого («зелёного») файла, он перейдёт в состояние modified и будет и в «зелёном», и в «красном» списках:

* зелёным отмечена предыдущая версия файла — в таком виде он был во время последнего запуска команды git add;
* красным отмечена версия с изменениями.

Чтобы запомнить новое состояние файла, нужно снова ввести команду git add.

$ git add filename.txt

*# или*

$ git add --all # будет сохранена (?) последняя добавленная версия # с изменениями

**Делаем коммит**

Коммит гарантирует, что изменения будут сохранены в истории и при необходимости к ним можно будет «откатиться».

Коммит (по названию команды git commit) — это по сути список файлов с их контентом.

### **Выполнить коммит — git commit**

$ git commit -m ‘Мой первый коммит!’ # текущая версия файлов будет сохранена в  
# репозитории с сообщением Мой первый коммит!

Команда git commit выведет информацию о коммите:

* [master (root-commit) baa3b6e] значит:
  + коммит был в ветке master;
  + root-commit — это самый первый, или «корневой» (англ. *root*), коммит в ветке, у следующих коммитов такой надписи не будет;
  + baa3b6e — сокращённый идентификатор коммита.
* 2 files changed, 1 insertion(+) значит:
  + изменились два файла;
  + одна строка была добавлена (строка, в которой произошли изменения).
* Строки вида create mode 100644 filename.txt — это более подробная информация о новых (добавленных в Git) файлах.
  + create (англ. «создать») говорит, что файл был создан. Если бы файл был удалён, на этом месте было бы слово delete (англ. «удалить»).
  + mode 100644 сообщает, что это обычный файл. Также возможны варианты 100755 для исполняемых файлов (например, что-нибудь.exe) и 120000 для файлов-ссылок в Linux. Файлы-ссылки не содержат данных сами по себе, а только ссылаются на другие файлы — как «ярлыки» в Windows.

**О разнице между**git add**и**git commit

Команда git add сообщает Git, какие именно файлы нужно сохранить и какую их версию.

Затем с помощью команды git commit происходит само сохранение.

Изменения, сделанные между git add и git commit сохранены не будут.

### **Просмотреть историю коммитов — git log**

$ git log

# Знакомство с GitHub