Первые шаги в Git и GitHub

Урок 2. Введение в Git

Система контроля версий (англ. Version Control System, или коротко VCS) — это программное обеспечение, которое помогает отслеживать изменения в программах, текстовых файлах, больших документах, веб-сайтах и так далее. Для обозначения систем контроля версий используют не только аббревиатуру VCS, но и SCM (от англ. Source Control Management — «система управления исходным кодом»). Одна из ключевых особенностей современных систем контроля версий — поддержка параллельной работы нескольких пользователей, в том числе над одним файлом. Именно поэтому VCS так популярны у IT-команд. Системы контроля версий: Git, Mercurial, Subversion и другие. Мы расскажем о Git.

Git (создатель Линус Торвальдс) — это программа, она требует установки и предварительной настройки. У Git нет привычного графического интерфейса, и работа с этим инструментом происходит иначе.

Работать с Git можно через командную строку либо графический интерфейс какой-либо IDE, например, VS Code.

В данном руководстве будет использоваться способ через консоль.

Для этого нужно установить специальный консольный инструмент для Windows, который называется **Git Bash** (способ установки описан в уроке 1). Он установит не только Bash, но и сам Git.

# Настройка Git

### Работа с файлом настройки .gitconfig

Сейчас вы работаете в одиночку, но в дальнейшем вам может понадобиться использовать Git в команде. Чтобы участникам проекта было понятно, кто и какие изменения вносил, нужно представиться и указать имя пользователя и адрес электронной почты.

Вы можете указать любую электронную почту и любое имя. Сделать это можно с помощью команды git config (от англ. *configuration —* «конфигурация», «настройка») с ключом --global (англ. «глобальный»). При этом не имеет значения, в какой директории вы находитесь прямо сейчас: вызов git config --global сработает везде.

В качестве значения user.name нужно указать своё имя или никнейм. Для настройки параметра user.email указывают электронную почту.

$ git config --global user.name "User Namovich"

# 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. Команда запишет в этот файл указанные имя и почту. Чтобы убедиться в этом, можно вызвать команду для чтения файлов.

$ cat ~/.gitconfig

Другой способ проверки — вывести содержимое файла конфигурации Git той же командой git config с флагом --list (англ. «список»).

$ git config --list

В ответ командная строка покажет текущие значения настроек.

user.name=Username

user.email=username@yandex.ru

Готово!

Таким образом, изучили базовые команды консоли, скачали Git и настроили основные параметры. Осталось только подключить Git к проекту и начать им пользоваться!

# 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 (от англ. **init***ialize* — «инициализировать»).

Например, создайте папку first-project и сделайте её Git-репозиторием: перейдите в неё с помощью команды cd и выполните git init.

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

$ git init # создали репозиторий

Вы можете создать папку в любом месте на компьютере. Но в этом случае не забывайте менять в наших примерах путь ~/dev/first-project на тот, который ведёт к вашей папке. Помните, что не рекомендуется создавать репозиторий Git внутри другого Git-репозитория. Это может вызывать проблемы с отслеживанием изменений.

### «Разгитить» папку, если что-то пошло не так, — rm -rf .git

Если вы случайно сделали Git-репозиторием не ту папку, её можно «разгитить». Для этого нужно удалить скрытую подпапку .git.

$ cd <папка с репозиторием> # перешли в папку

$ rm -rf .git # удалили подпапку .git

Разберём подробнее, что такое -rf:

* ключ -r (от англ. **r***ecursive* — «рекурсивно») позволяет удалять папки вместе с их содержимым;
* ключ -f (от англ. **f***orce* — «заставить») избавит вас от вопросов вроде «Вы точно хотите удалить этот файл? А этот? И этот тоже?».

Будьте осторожны: в подпапке .git хранится история изменений. Если удалить .git, то вся история проекта будет стёрта без возможности восстановления — останется только последняя версия файлов.

### Проверить состояние репозитория — git status

После инициализации репозитория first-project запустите команду git status (от англ. *status* — «статус», «состояние») — она показывает текущее состояние репозитория.

Команда git status 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).

Подробнее о том, что такое коммиты, мы расскажем в следующих уроках.

В отличие от git init, команду git status используют часто. В любой непонятной ситуации стоит посмотреть состояние (статус) репозитория, а потом решить, что делать дальше.

Добавляем файлы в репозиторий

Создайте файлы todo.txt и readme.txt в папке first-project и запустите git status, чтобы посмотреть, что изменилось.

$ touch todo.txt

$ touch readme.txt

# создали файлы todo.txt и readme.txt

$ git status # проверили статус

Git сообщит, что в папке first-project есть untracked files (от англ. track — «следить», untracked — «неотслеженный», «неотслеживаемый») — ещё не отслеживаемые файлы readme.txt и todo.txt.

Состояние untracked значит, что Git ещё не хранит информацию о версиях файла и не может отследить, как он изменялся.

Сейчас в first-project два файла. Команда git add - подготовить файлы к сохранению. Мы хотим отслеживать состояние обоих, поэтому можем использовать команду git add --all (от англ. add — «добавить» + от англ. all — «всё»). Ключ, или флаг, --all позволяет подготовить к сохранению все файлы в репозитории.

$ git add --all # подготовили к сохранению все файлы в репозитории

$ git status # проверили статус

Добавлять файлы можно и по одному, без ключа --all.

$ git add todo.txt

$ git add readme.txt

$ git status

Также можно добавить текущую папку целиком — в этом случае все файлы в ней тоже будут добавлены. Обратиться к текущей папке в Bash позволяет точка (.).

$ git add . # добавить всю текущую папку

$ git status

Вы можете использовать любой из этих вариантов — результат будет одинаковый.

Файлы, которые отмечены зелёным, теперь отслеживаются и готовы к сохранению. Но сохранения пока не произошло, потому что команда git add только запоминает текущее содержимое (контент) файла.

Делаем первый коммит

Коммит — это одна из основных сущностей в Git (и в других системах контроля версий). Коммит гарантирует, что изменения будут сохранены в истории и при необходимости к ним можно будет «откатиться». Это как если бы вы могли выполнить операцию Ctrl+Z для целой папки (репозитория).

В этом уроке вы сделаете свой первый коммит.

### Выполнить коммит — git commit

Сделать коммит можно командой git commit c ключом -m (от англ. **m***essage* — «сообщение»), который присваивает коммиту сообщение.

Обычно в таком сообщении поясняется, в чём именно состояли изменения. Это как заметки на полях: благодаря им проще читать и понимать текст. Сообщение коммита выполняет те же функции — улучшает понимание и упрощает навигацию. Оно пишется после ключа -m в кавычках.

Например, перейдите в папку first-project и выполните коммит со следующим комментарием.

$ git commit -m 'Мой первый коммит!'

После нажатия Enter текущая версия файлов будет сохранена в репозитории с сообщением Мой первый коммит!. **Коммит** (по названию команды git commit) — это по сути список файлов с их контентом.

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 (подробнее об этом мы ещё расскажем).
* 2 files changed, 1 insertion(+) значит:
  + изменились два файла (readme.txt и todo.txt);
  + одна строка была добавлена (1. Пройти пару уроков по Git.).
* Строки вида create mode 100644 readme.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.

Просматриваем историю коммитов

Просмотреть историю коммитов — git log

Обратите внимание, что по умолчанию git log выводит коммиты в обратном хронологическом порядке — последние коммиты оказываются первыми сверху. В этом можно убедиться, если посмотреть на дату и время их создания.

Если после выполнения команды вы видите, что в репозитории есть только один коммит или их нет вообще, вернитесь к прошлому уроку и убедитесь, что git add и git commit были вызваны в нужном порядке.