Первые шаги в Git и GitHub

Урок 4. Синхронизация репозиториев

**Что такое SSH.** **Генерируем SSH-ключ**

Когда компьютеры обмениваются данными в сети, они следуют сетевым протоколам (англ. network protocols) — правилам обмена данными между компьютерами.

Один из наиболее распространённых сетевых протоколов — SSH (от англ. Secure Shell Protocol). Он обеспечивает безопасный обмен данными в сети. С помощью этого протокола можно получать данные с удалённого компьютера или отправлять их на него. Трафик шифруется, поэтому протокол безопасен.

SSH использует пару ключей для обеспечения безопасности — публичный и приватный:

* Приватный ключ (англ. private key) хранится только на вашем компьютере и не должен передаваться кому-либо ещё. Он используется для расшифровки данных.
* Публичный ключ (англ. public key) доступен всем и используется для шифрования данных. Они могут быть расшифрованы парным приватным ключом.

Только вы можете расшифровать данные с помощью приватного ключа, но любой владелец публичного ключа может их для вас зашифровать. Эти два ключа связаны и образуют SSH-пару. В будущем вы наверняка будете использовать их для взаимодействия с GitHub и другими удалёнными серверами.

**Проверка наличия SSH-ключа**

Прежде чем генерировать SSH-ключи, убедитесь, что у вас их ещё нет. По умолчанию директория с SSH-ключами находится в домашней директории пользователя. Перейдите в неё.

$ cd ~ # перешли в домашнюю директорию

Обычно SSH-ключи находятся в директории .ssh/. Проверить наличие этой директории и файлов в ней можно с помощью следующей команды.

$ ls -la .ssh/ # вывели список созданных ключей

Если папка пустая или её нет, всё в порядке.

Если есть файлы с похожими названиями, SSH-ключи уже создавались:

* id\_dsa.pub;
* id\_ecdsa.pub;
* id\_ed25519.pub;
* id\_rsa.pub.

Если вы не создавали эти файлы, удалите их все.

**Инструкция по генерации SSH-ключа**

1. Для генерации SSH-пары можно использовать программу ssh-keygen. Откройте терминал и введите следующую команду.

$ ssh-keygen -t ed25519 -C "электронная почта, к которой привязан ваш аккаунт на GitHub"

Используйте электронную почту, к которой привязан ваш GitHub-аккаунт.

Если вы видите сообщение об ошибке, то, скорее всего, ваша система не поддерживает алгоритм шифрования ed25519. Ничего страшного: используйте другой алгоритм.

$ ssh-keygen -t rsa -b 4096 -C "электронная почта, к которой привязан ваш аккаунт на GitHub"

После ввода отобразится такое сообщение.

> Generating public/private rsa key pair. # сгенерированы публичный и приватный ключи

1. Укажите место хранения ключей. Простой вариант — сделать домашний каталог пользователя путём по умолчанию. Для этого нажмите Enter.
2. Программа запросит кодовую фразу (англ. passphrase) для доступа к SSH-ключу. Вы можете оставить поле пустым. Для этого нажмите Enter, а затем ещё раз Enter для подтверждения.

**Быть или не быть кодовой фразе — вот в чём вопрос**

Как бы странно ни звучало, кодовая фраза — это «пароль от ключа». Представьте, что SSH-ключ лежит в шкатулке. А на самой шкатулке — кодовый замок, который открывается кодовой фразой.

Многие пользователи Git не используют кодовую фразу для защиты своего SSH-ключа. Если такой фразы нет, то её не нужно вводить всякий раз при взаимодействии с удалённым репозиторием.

С другой стороны, применение кодовой фразы усиливает безопасность ключей. Если вы используете эту фразу, ключ будет надёжно защищён в случае несанкционированного доступа к вашему компьютеру.

1. Готово! Теперь осталось проверить, что ключи действительно сгенерировались. Для этого вызовите эту команду.

ls -a ~/.ssh

На экране должны появиться два файла — один с расширением .pub, другой — без. Файл в .pub — публичный, им можно делиться с веб-сайтами или коллегами. **Файл без расширения .pub — приватный. Ни в коем случае не передавайте его никому!**

Подведём итоги:

* SSH — протокол, который обеспечивает безопасный обмен данными в сети и использует для этого ключи.
* SSH-ключ — ваш виртуальный идентификатор в GitHub. Виртуальный идентификатор, или виртуальный ключ, - это специальное цифровое сочетание, которое позволяет идентифицировать вас как пользователя при работе с системой контроля версий Git. Оно нужно для того, чтобы другие разработчики могли точно знать, что именно вы совершили то или иное изменение в коде проекта. Виртуальные идентификаторы бывают двух видов: SSH-ключи и пароли. SSH-ключи считаются более безопасными, так как они не передаются по сети в открытом виде, а пароли иногда могут быть уязвимы для хакеров.
* SSH-ключ состоит из двух частей — публичной и приватной. Публичный ключ зашифрует данные, а приватный — расшифрует. Приватным ключом ни в коем случае нельзя делиться, иначе любой сможет расшифровать все ваши секреты!

Привязываем SSH-ключ к GitHub

Инструкция по связыванию SSH-ключа и GitHub-аккаунта

1. Скопируйте содержимое файла с публичным ключом (id\_ed25519.pub/id\_rsa.pub) в буфер обмена.

Windows

$ clip < ~/.ssh/id\_rsa.pub скопировать содержимое ключа в буфер обмена

или

$ clip < ~/.ssh/id\_ed25519.pub

Если clip не сработает, выведите содержимое файла с помощью cat ~/.ssh/id\_rsa.pub или cat ~/.ssh/id\_ed25519.pub и скопируйте вывод в буфер обмена из консоли.

1. Перейдите на GitHub и выберите пункт Settings (англ. «настройки») в меню аккаунта (справа).
2. В меню слева нажмите на пункт SSH and GPG keys.
3. В открывшейся вкладке выберите New SSH key (англ. «новый SSH-ключ»).
4. В поле Title (англ. «заголовок») напишите название ключа. Например, Personal key (англ. «личный ключ»).
5. В поле Key type (англ. «тип ключа») должно быть Authentication Key (англ. «ключ аутентификации»).
6. В поле Key скопируйте ваш ключ из буфера обмена.
7. Нажмите на кнопку Add SSH key (англ. «добавить SSH-ключ»).
8. Проверьте правильность ключа с помощью следующей команды в консоли:

$ ssh -T [git@github.com](mailto:git@github.com)

Если это первый раз, когда вы используете Git, чтобы поделиться проектом на GitHub, появится похожее предупреждение.

The authenticity of host 'github.com (140.82.121.4)' can't be established. ED25519 key fingerprint is SHA256:+DiY3wvvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvCOqU. This key is not known by any other names. Are you sure you want to continue connecting (yes/no/[fingerprint])?

Это предупреждение сообщает, что вы никогда не соединялись с сервером GitHub. Поэтому Git не может гарантировать, что сервер является тем, за кого он себя выдаёт.

Для подтверждения подлинности сервер генерирует и публикует ключи SHA256. Вы можете проверить ключи GitHub [по этой ссылке](https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/githubs-ssh-key-fingerprints).

<https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/githubs-ssh-key-fingerprints>

Если ключ в предупреждении совпадает с тем, что вы видите на сайте, значит, сервер является действительным. Введите yes, чтобы продолжить. Вы увидите приветствие на экране.

Hi %ВАШ\_АККАУНТ%! You've successfully authenticated, but GitHub does not provide shell access.

Теперь ваш ключ привязан к GitHub! Если вы установили кодовую фразу для SSH-ключа, её нужно будет вводить для работы с репозиторием.

**Связываем локальный и удалённый репозитории**

Сейчас у вас есть локальный репозиторий first-project, который хранится на вашем компьютере, и удалённый репозиторий на GitHub. Вы сгенерировали SSH-ключ для безопасной работы и теперь готовы связать удалённый репозиторий с локальным.

**Привязать удалённый репозиторий к локальному — git remote add**

Перейдите на страницу удалённого репозитория, выберите тип SSH и скопируйте URL. Кнопка справа позволит сделать это мгновенно.

Откройте консоль, перейдите в каталог локального репозитория и введите команду git remote add (от англ. remote — «удалённый» и add — «добавить»).

$ cd ~/dev/first-project

$ git remote add origin git@github.com:%ИМЯ\_АККАУНТА%/first-project.git

Команде необходимо передать два параметра: имя удалённого репозитория и его URL. В качестве имени например будем использовать слово origin. А URL вы скопировали со страницы удалённого репозитория.

**Как выполнить вставку в командную строку?**

На Windows (в Git Bash) для этого используется сочетание Ctrl+Shift+V.

Также можно нажать правую кнопку мыши и выбрать пункт **Paste** (англ. «вставить») в выпадающем меню.

origin (англ. «источник») — стандартный псевдоним, с помощью которого можно обращаться к главному удалённому репозиторию (обычно такой репозиторий один).

**Убедиться, что репозитории связаны, — git remote -v**

$ git remote -v

origin git@github.com:%ИМЯ\_АККАУНТА%/%ИМЯ-ПРОЕКТА%.git (fetch)

origin git@github.com:%ИМЯ\_АККАУНТА%/%ИМЯ-ПРОЕКТА%.git (push)

В выводе вы должны увидеть две строчки, аналогичные тем, что показаны выше.

Флаг -v — короткая форма флага --verbose (англ. «подробный»). Он позволяет показать больше информации в выводе.

**Синхронизируем локальный и удалённый репозитории**

Разберём, как выкладывать свои правки на удалённый репозиторий. Но сначала немного о ветках.

**Основная ветка**

Мы упоминали, что каждый коммит сохраняет актуальное состояние файлов. Сами же коммиты хранятся в ветках (англ. branch).

Если коммит — это снимок состояния файлов, то ветка — временна́я шкала, на которой расположены эти снимки. Ветка всегда начинается от одного из коммитов.

В репозитории может существовать сразу несколько веток — параллельных историй изменений. Также они могут соединяться друг с другом.

Самая первая ветка в репозитории появляется автоматически и называется main (англ. «основная») или master. Её имя нужно указывать при отправке коммитов на удалённый репозиторий или при получении их из него.

main или master?

Раньше основная ветка в репозиториях, созданных на GitHub, называлась master, но с 1 октября 2020 года (после волны протестов движения Black Lives Matter) её переименовали в main.

Во всех репозиториях, созданных раньше этой даты, название основной ветки не поменялось. Поэтому в проектах, которые начали именно с master, и в руководствах по работе с Git вы по-прежнему можете встретить имя master.

**Отправить изменения на удалённый репозиторий — git push**

Вы уже прошли весь «цикл коммита»: подготовили файлы с помощью git add, закоммитили их с комментарием командой git commit -m. Осталось загрузить содержимое локального репозитория на GitHub. За это отвечает команда git push (от англ. push — «толкать»).

В первый раз эту команду нужно вызвать с флагом -u и параметрами origin (имя удалённого репозитория) и main или master (название текущей ветки). Флаг -u свяжет локальную ветку с одноимённой удалённой. Как вы связывали локальный и удалённый репозитории в предыдущем уроке, так же и здесь нужно дополнительно связать ветки.

$ git push -u origin main # Если команда приведёт к ошибке, попробуйте заменить main на master.

При взаимодействии с удалёнными репозиториями Git выводит в консоль отладочную информацию: количество объектов (файлов), которые отправляются на сервер, информацию о прогрессе сжатия и записи и так далее.

Если вы указывали кодовую фразу при настройке SSH-ключей, её нужно будет ввести.

Зайдите в репозиторий first-project на GitHub. Вы увидите, что в репозитории появились файлы с изменениями.

В дальнейшем при работе с удалённым репозиторием флаг -u можно опустить и писать просто git push.

Работа с графическим интерфейсом GitHub

GitHub предоставляет удобный интерфейс для работы с репозиторием. Например, нажмите на кнопку commit в правой части страницы, чтобы просмотреть все коммиты в репозитории.

Откроется окно с коммитами и их авторами.

Сообщение коммита в репозитории тоже является ссылкой.

Перейдите по ссылке, кликните на текст последнего коммита над репозиторием — так вы сможете увидеть все изменения, которые были внесены в репозиторий в этом коммите.

Файл README.md

Чтобы другие пользователи, а также потенциальные клиенты или работодатели могли понять, что представляет собой проект, его нужно описать. Такое описание принято указывать в файле README.md (от англ. read — «прочитай» и me — «меня»). В этом уроке вы научитесь оформлять такие файлы.

**Подробнее о том, зачем нужен README.md**

Как правило, в README.md проекта можно найти следующую информацию:

1. Название проекта и его краткое описание: кем создан, для чего, какие решает задачи и какие закрывает проблемы.
2. Технологии, которые применяются в проекте. В чём его отличие от аналогичных.
3. Документация проекта — подробная инструкция о том, что представляет собой проект.
4. Планы проекта, если они есть.

**Как создать и оформить README.md**

README.md — текстовый файл, который можно создать командой touch, а затем редактировать так же, как и любой другой текстовый документ. Например, в блокноте.

Преимущество README.md в том, что средства командной работы (такие, как GitHub) могут отображать его содержимое в браузере в виде удобной разметки. Для этого нужно не просто залить текст, но и настроить шрифт, заголовки и отступы с помощью markdown. Маркда́ун — это специальный язык разметки. Он позволяет красиво отформатировать текстовый документ.

Разберём базовый синтаксис этого языка. Все правила запоминать не нужно: при оформлении репозитория вы всегда можете вернуться к этому уроку.

**Заголовки, абзацы и перенос**

* Заголовки разных уровней создают решётками.

# H1 — заголовок первого уровня, самый большой

## H2 — заголовок второго уровня, поменьше

### H3

#### H4

##### H5

###### H6 — заголовок шестого уровня, самый маленький

* Можно добавить черту под заголовком или абзацем.

#### Заголовок 4

Текст над чертой

---

Текст под чертой

* Чтобы сделать разрыв строки, нужно поставить два пробела (в примере ниже они обозначены точками ⋅⋅) или сочетание символов <br>.

Текст до переноса⋅⋅

Текст после переноса <br>

Текст после второго переноса

* Чтобы начать новый параграф, в конце предыдущей строки должно стоять два символа переноса. Для этого нужно нажать Enter два раза.

line

another line

Если сделать один перенос строки, как в примере ниже, и не поставить два пробела, текст сольётся в одну строку.

Скопировать кодMARKDOWN

line

another line

Вот пример текста в файле README.md.

**Выделение текста**

* Чтобы выделить текст курсивом (\*текст\*), его заключают в звёздочки (астериски) или нижние подчёркивания.

Курсив — это \*звёздочки\* или \_подчёркивания\_.

* Чтобы выделить текст полужирным шрифтом (\*\*текст\*\*), его окружают двойными звёздочками или двойными нижними подчёркиваниями.

Полужирный шрифт — двойные \*\*звёздочки\*\* или двойные \_\_подчёркивания\_\_.

Можно совместить выделение \*\*звёздочки и \_подчёркивания\_\*\*.

* Чтобы зачеркнуть текст (~~текст~~), его окружают двойными волнистыми линиями — тильдами.

~~Зачёркнутый текст.~~

**Списки**

* Для оформления нумерованного списка достаточно поставить в начало строки цифры с точкой.

1. Первый пункт нумерованного списка.

2. Второй пункт.

* Ненумерованный список создаётся звёздочкой с пробелом в начале строки либо дефисом с пробелом.

\* первый пункт ненумерованного списка;

\* второй пункт ненумерованного списка

- первый пункт ненумерованного списка;

- второй пункт ненумерованного списка

**Ссылки**

* Чтобы сделать ссылкой часть текста, его заключают в квадратные скобки, а затем указывают нужный адрес в круглых скобках.

[Яндекс](https://www.yandex.ru)

* Также можно добавить ссылке тайтл (от англ title — «название», «заголовок»). Тайтл — это всплывающая подсказка, которая появляется при наведении мыши на ссылку. Тайтл нужно заключить в кавычки и указать внутри скобок после адреса.

[Яндекс](https://www.yandex.ru "Я Yandex!")

**Код**

Чтобы оформить текст как код, нужно окружить его тройками косых кавычек — грависов. После первой тройки грависов указывают язык программирования, на котором написан код. В маркдауне есть поддержка синтаксиса почти всех популярных языков и инструментов.

```bash

ls - la

```

```html

<h1>А я просто текст</h1>

```

Обратите внимание: вторая тройка тройных кавычек стоит на отдельной строке.

**Пример файла README.md**

Если собрать всё вместе, файл README.md может выглядеть так.

Скопировать кодBASH

# Шпаргалка markdown

## Выделение текста

Вы можете выделять текст в markdown с помощью символов `\_` или `\*`. Например:

Пример \_курсива\_ и \*\*жирного\*\* текста.

## Заголовки

Заголовки можно создавать с помощью символа `#`. Чем больше `#`, тем меньше заголовок. Например:

# Заголовок первого уровня

## Заголовок второго уровня

### Заголовок третьего уровня

## Выделение кода

Чтобы выделить текст как код, поместите его в тройные кавычки `````.

```

mkdir my\_project

cd my\_project

git init

```

Это лишь некоторые функции markdown.

Файл readme.md будет встречать гостей репозитория.

