1.9. Установка SSH-ключей

~ 8 минут

В предыдущей главе мы установили и настроили терминал для работы с Git, теперь с его помощью добавим SSH-ключ. И можно будет приступать к работе с проектом.

Что такое SSH-ключ и зачем он нужен?

SSH — это протокол для безопасного соединения между компьютерами. Его используют не только для Git, например, через SSH системные администраторы управляют серверами.

Чтобы вы могли работать со своего компьютера с GitHub, иметь доступ к проектам, хранящимся на сервере, выполнять команды в терминале без постоянного подтверждения пароля, нужно пройти авторизацию у сервера. В этом помогают SSH-ключи.

Каждый SSH-ключ состоит из пары: открытый (публичный) и закрытый (приватный) ключ. Открытый ключ отправляется на сервер, его можно не прятать от всех и не переживать, что кто-то его увидит и украдёт. Он бесполезен без своей пары — закрытого ключа. А вот закрытый ключ — секретная часть. Доступ к нему должен быть только у вас.

Чтобы было понятнее, можно представить, что публичный ключ — это замок. Мы добавим его в GitHub, как будто поставим на GitHub свой замок. А приватный ключ — это ключ к этому замку. Если кто-то украдёт приватный ключ, он сможет попасть в ваш GitHub — откроет ваш замок краденным ключом. Поэтому никому не передавайте свой приватный ключ.

Общение с GitHub по SSH происходит так: вы отправляете какую-то информацию на сервер, где хранится ваш публичный ключ, сервер понимает, что вы это вы, то идентифицирует именно вас, и даёт вам какой-то ответ. И только вы можете расшифровать этот ответ, потому что только у вас есть подходящий закрытый кл

Чтобы пройти авторизацию по SSH-ключу, его надо сгенерировать или найти уже ранее созданный ключ на своём компьютере.

Проверяем наличие SSH-ключа

Перед созданием нового SSH-ключа проверим, есть ли на компьютере другие ключи. Все команды выполняем в git-bash, bash или zsh.

1. Открываем терминал.
2. Вводим `ls -al ~/.ssh`, чтобы увидеть список всех ключей, которые есть на компьютере.

```
$ ls -al ~/.ssh
# Список файлов в вашей директории .ssh, если они существуют
# Знак $ вставлять не нужно
```

3. Если у вас уже есть SSH-ключ, то в результате запроса вы увидите список файлов. Там могут встретиться файлы с такими именами:

   — *id_rsa.pub*

   — *id_ecdsa.pub*

   — *id_ed25519.pub*

   Примечание: Если вы получили сообщение об ошибке, что директории `~/.ssh` не существует, значит, у вас нет SSH-ключей. Можно переходить к шагу создания новых ключей.

4. Можно использовать существующий ключ или создать новый.

Генерируем новый SSH-ключ

Если проверка показала, что на компьютере нет SSH-ключей, можно сгенерировать новый SSH-ключ. Для этого делаем следующее:

1. Открываем терминал.
2. Копируем и вставляем текст ниже, подставив свой адрес электронной почты на GitHub:

```
$ ssh-keygen -t ed25519 -C "твоя@электронная.почта"
```

Если у вас старая версия системы, которая не поддерживает алгоритм Ed25519, используйте эту команду:

```
$ ssh-keygen -t rsa -b 4096 -C "твоя@электронная.почта"
```

В результате создаётся новый SSH-ключ, который привязан к вашей электронной почте. В терминале появится запись:

```
> Generating public/private имя-ключа key pair.
```

3. Нажмите Enter, когда увидите запись «Enter a file in which to save the key», чтобы сохранить получившийся файл с ключом. По умолчанию файл будет лежать в директории, на которую указывает запись:

```
> Enter a file in which to save the key (/c/Users/ваш-профиль/.ssh/id_имя-ключа):*[Press enter]*
```

4. Теперь можно добавить пароль. Проходить этот шаг необязательно. Можно пропустить ввод пароля, нажав Enter два раза подряд. При вводе пароля курсор не двигается. Но пароль вводится.

```
> Enter passphrase (empty for no passphrase):[вводи пароль]
> Enter same passphrase again:[вводи пароль снова]
```

Добавляем SSH-ключ в ssh-агент

`ssh-agent` — специальная программа для хранения и управления SSH-ключами. Ставить её не надо, скорее всего она уже есть на твоём компьютере. Чтобы добавить SSH-ключ делаем следующее:

1. Убедитесь, что ssh-агент запущен. Вы можете запустить его вручную с помощью команды `eval "$(ssh-agent -s)"`:

```
# start the ssh-agent in the background
$ eval "$(ssh-agent -s)"
> Agent pid 59566
```

2. Добавьте свой ключ SSH в ssh-агент. Если вы создали свой ключ с другим име замените название `id_ed25519` в команде именем вашего файла с ключом:

```
$ ssh-add ~/.ssh/id_ed25519
```

3. Можно переходить к шагу с добавлением ключа на GitHub.

Добавляем SSH-ключ на GitHub

1. Скопируйте SSH-ключ в буфер обмена.

   Если имя вашего файла с SSH-ключом отличается от того, что указано в примере кода, измените имя файла. При копировании ключа не добавляйте новые строки или пробелы.

   Windows:

   ```
   $ clip < ~/.ssh/id_ed25519.pub
   # Copies the contents of the id_ed25519.pub file to your clipboard
   ```

   Mac OS:

   ```
   $ pbcopy < ~/.ssh/id_ed25519.pub
   ```

   Если команда `clip` или `pbcopy` не работает, вы можете найти скрытую папку `.ssh`, открыть файл в своем любимом текстовом редакторе и скопировать его содержимое в буфер обмена. Или введите команду `cat ~/.ssh/id_ed25519.pub`, контент документа появится прямо в терминале и вы сможете скопировать ключ оттуда.

2. Перейдите на страницу для работы с ключами в вашем профиле на GitHub.

3. Нажмите кнопку с названием «New SSH key» (новый SSH-ключ).

4. В поле «Title» (название) можно добавить название для нового ключа, которое кратко опишет этот ключ. Например, если вы используете Mac, вы можете назвать ключ «MacBook Air».

5. Вставьте скопированный ключ в поле «Key» (ключ).

6. Нажмите «Add SSH key» (добавить SSH-ключ).

7. Если потребуется, введите свой пароль на GitHub, чтобы подтвердить сохранение ключа.

Если всё сделано верно, в списке появится новый ключ.

Проверяем работу ключа

1. Открываем терминал
2. Вводим команду:

```
$ ssh -T git@github.com
```

Если возникнет следующее уведомление, то нужно набрать yes

```
> The authenticity of host 'github.com (IP ADDRESS)' can't be established.
> RSA key fingerprint is SHA256:nThbg6kXUpJWGl7E1IGOCspRomTxdCARLviKw6E5SY8.
> Are you sure you want to continue connecting (yes/no)?
```

3. Вводим пароль (визуально пароль не отображается)
4. Должно появится следующее сообщение:

```
Hi имя_пользователя! You've successfully authenticated, but GitHub does not
provide shell access.
```

5. Если возникли ошибки, то идём в начало статьи и начинаем всё заново ;)

Теперь, наконец-то, мы можем начать работу с самим проектом.

Прочитали главу?

Нажмите кнопку «Готово», чтобы сохранить прогресс.

Готово

- [ ] Пройдите опрос
- [ ] Укажите персональные данные
- [ ] Изучите регламент
- [ ] Прочитайте FAQ
- [ ] Добавьте свой Гитхаб
- [ ] Выберите наставника
- [ ] Создайте проект

Мой наставник

Выбрать наставника

Работа с наставником

У вас осталось **10** из 10 консультаций.

История

Курсы

HTML и CSS. Профессиональная вёрстка сайтов

HTML и CSS. Адаптивная вёрстка и автоматизация

JavaScript. Профессиональная разработка веб-интерфейсов

JavaScript. Архитектура клиентских приложений

React. Разработка сложных клиентских приложений

Node.js. Профессиональная разработка REST API

Node.js и Nest.js. Микросервисная архитектура

TypeScript. Теория типов

Алгоритмы и структуры данных

Паттерны проектирования

Webpack

Vue.js 3. Разработка клиентских приложений

Git и GitHub

Анимация для фронтендеров

Блог

С чего начать

Шпаргалки для разработчиков

Отчеты о курсах

Услуги

Работа наставником

Для учителей

Стать автором

Информация

Об Академии

О центре карьеры

Остальное

Написать нам

Мероприятия

Форум

Соглашение

Конфиденциальность

Сведения об образовательной организации

Лицензия № 4696

Sk

Участник