**1 – практическая работа. Основы Git**

**Цель практической работы**: Ознакомится с основнами Git, установки Git, настройка Git, создание репозитария Git.

**Приступаем к Работе**

Установка и настройка git на Windows.

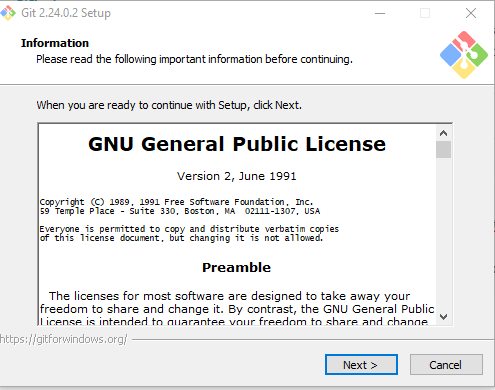
Windows в виду своих особенностей не имеет подходящего функционала для работы с git из под коробки. Для работы с git'ом будем использовать Git for Windows. [Подробнее о git](https://git-scm.com/book/ru/v1/%D0%92%D0%B2%D0%B5%D0%B4%D0%B5%D0%BD%D0%B8%D0%B5).

1. Заходим на страницу [загрузки Git for Windows](https://git-scm.com/download/win)
2. Выбираем и скачиваем нужную версию 64-bit или 32-bit
3. Устанавливаем, следуя инструкциям.
4. После установки появится несколько приложений: GIT GUI, GIT BASH и т.д. и т.п.

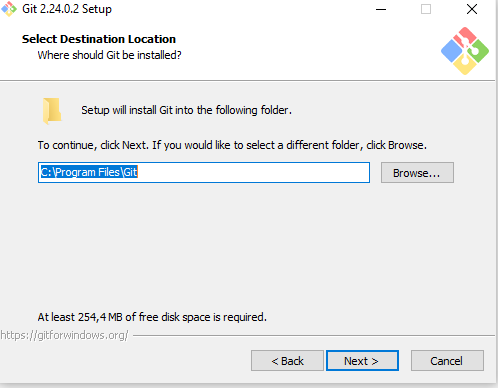
GIT GUI — графический клиент для работы с git.

GIT BASH — консольный клиент для работы с git.

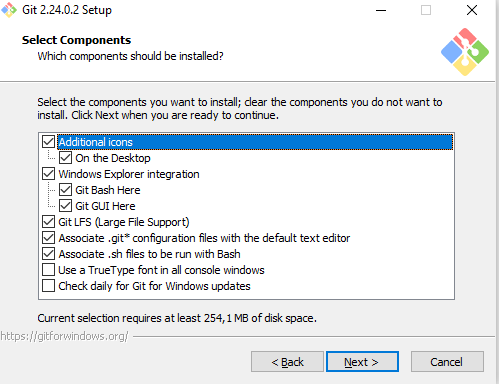
Установка Git



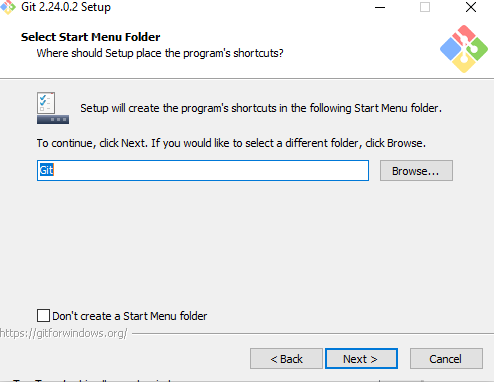
Нажимаем Next



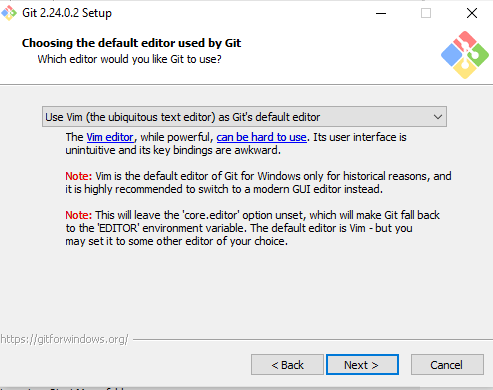
Выбираем пути и жмём Next



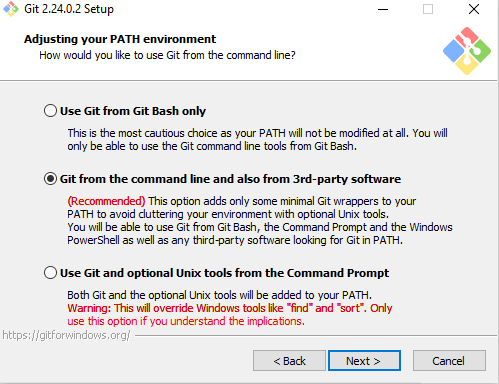
Ставим нужные нам галочки и жмем Next



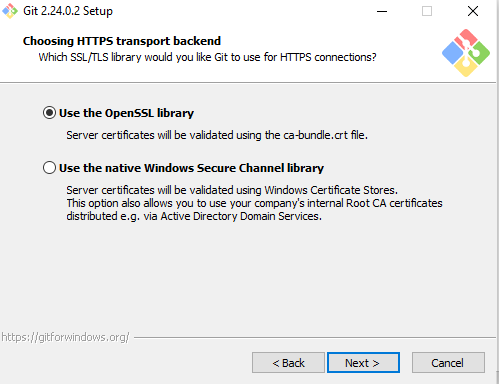
Жмем Next



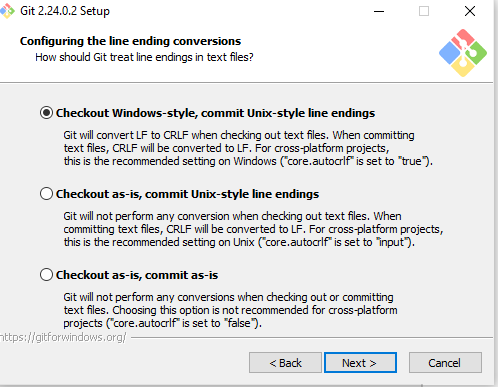
Выбираем нужный нам пункт и жмём Next



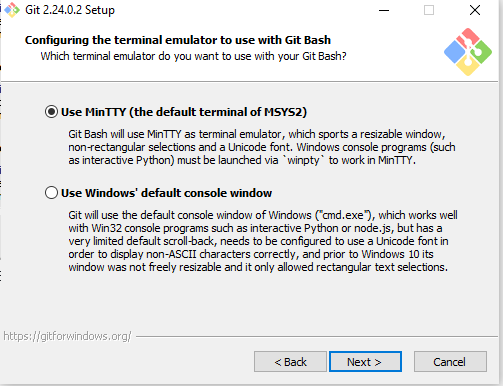
Выбираем пункт любой , но лучше выбрать Bash



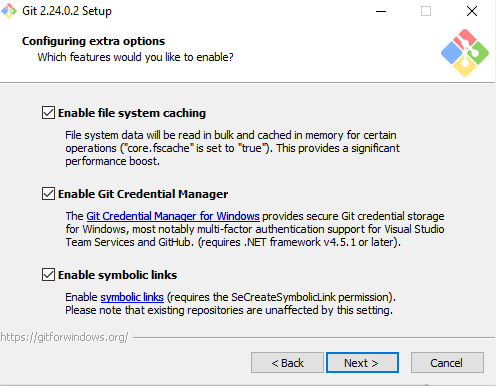
Выбираем пункт любой



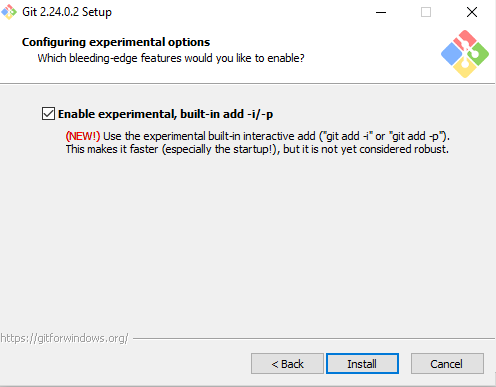
Выбираем пункт любой



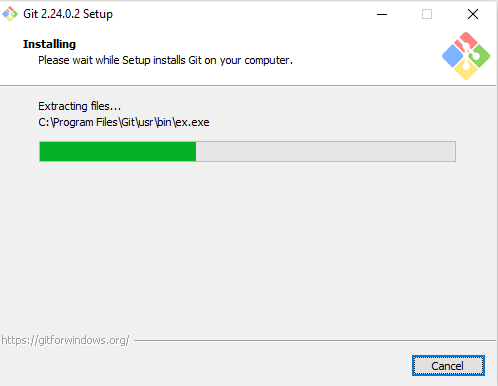
Выбираем пункт любой



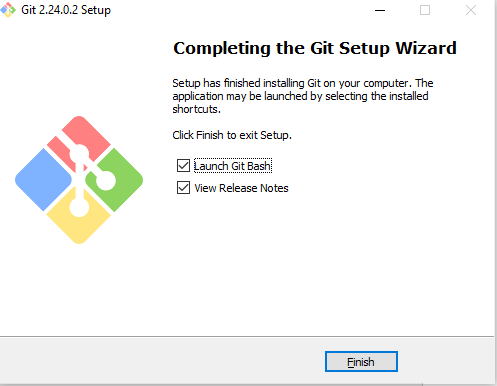
Ставим Галочки



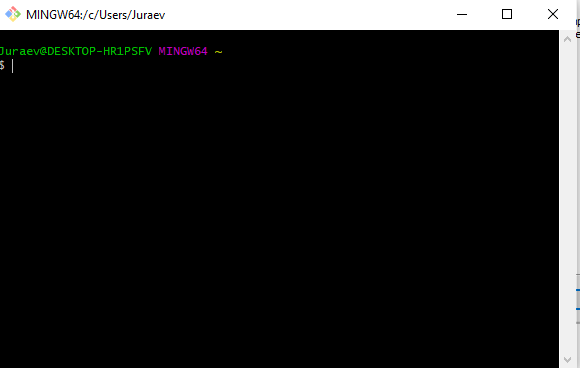
Ставим Галочки



Ждём завершение установки



Всё готово Ура! 😊

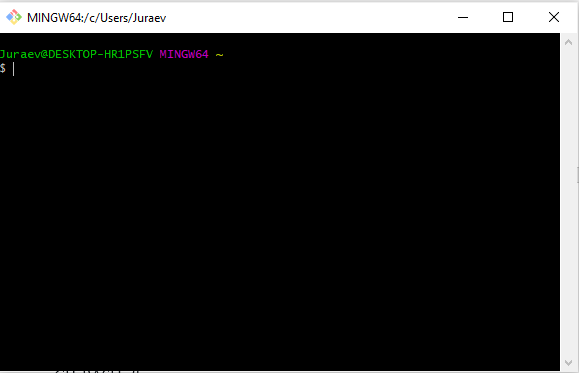


Можно приступать к работе

## **Настройка git для работы с Github**

Мы будем пользоваться GIT BASH.

1. Запускаем GIT BASH. Должно появиться следующее окно:



1. Чтобы проверить корректность установки вбиваем

git --version



1. Каждый коммит (изменение) подписывается мылом и именем. Для того, чтобы гит их запомнил, вводим (вместо Victor своё имя, вместо моей почты — свою):

git config --global user.name "Victor" git config --global user.email [tvs.private@gmail.com](mailto:tvs.private@gmail.com)

4. Для авторизации на гитхаб удобнее использовать SSH-ключи (Такой механизм защиты, альтернатива паре логин-пароль). Сначала этот ключ нужно сгенерировать.

`ssh-keygen -t rsa -b 4096 -C "your\_email@example.com"`

5. Появится сообщение о выборе файла в который нужно сохранить ключ. Чтобы использовать имя файла по умолчанию нажимаем ENTER:

`Enter a file in which to save the key (/Users/you/.ssh/id\_rsa): [Press enter]`

6. Дальше программа попросит ввести пароль и повторить его. Я использую без пароля (Ничего не ввожу). Нажимаем ENTER:

Enter passphrase (empty for no passphrase): [Type a passphrase] Enter same passphrase again: [Type passphrase again]

7. Теперь в папке ~/.ssh появилось два файлика:

![Файлы ключей](http://clip2net.com/clip/m175449/aa932-clip-12kb.png?nocache=1)

\*\*id\_rsa\*\* — приватный ключ, который нужно хранить на компьютере и никому никогда не показывать.

\*\*id\_rsa.pub\*\* — публичный ключ, который мы должны разместить в профиле на гитхабе.

8. Сначала "активируем" приватный ключ. Вбиваем:

eval "$(ssh-agent -s)" ssh-add ~/.ssh/id\_rsa

![ssh-agent](http://clip2net.com/clip/m175449/86224-clip-13kb.png?nocache=1)

9. Теперь добавим публичный ключ в github. Сначала скопируем содержимое ключа в буфер обмена (Можно его открыть любым текстовым редактором и скопировать через него):

`clip < ~/.ssh/id\_rsa.pub`

10. Заходим на [страницу управления ssh-ключами](https://github.com/settings/keys)

11. Нажимаем кнопку \*New ssh key\*

![Add new ssh-key](https://help.github.com/assets/images/help/settings/ssh-add-ssh-key.png)

12. В поле \*Title\* вводим название, например имя компьютера. (Моё — win\_lasto4ka).

13. В поле \*Key\* копируем содержимое ключа.

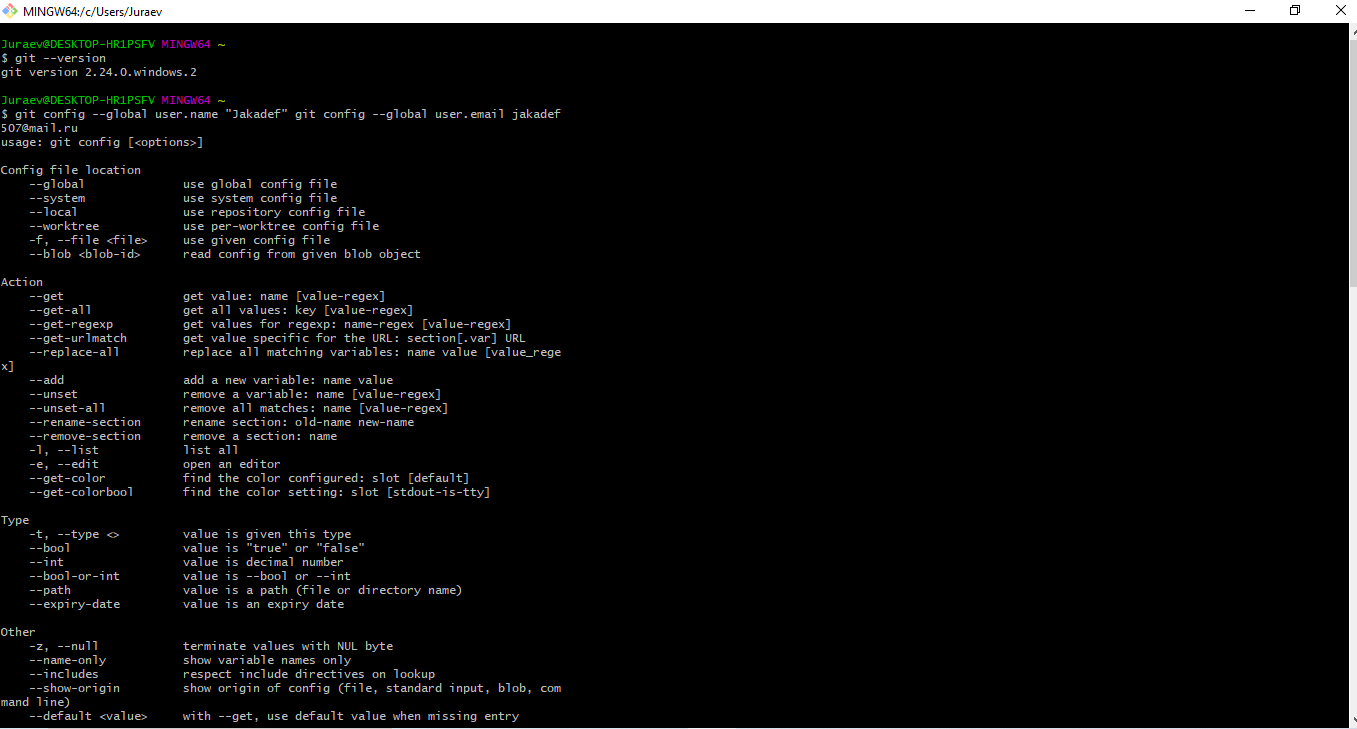
![Paste ssh-key](https://help.github.com/assets/images/help/settings/ssh-key-paste.png)

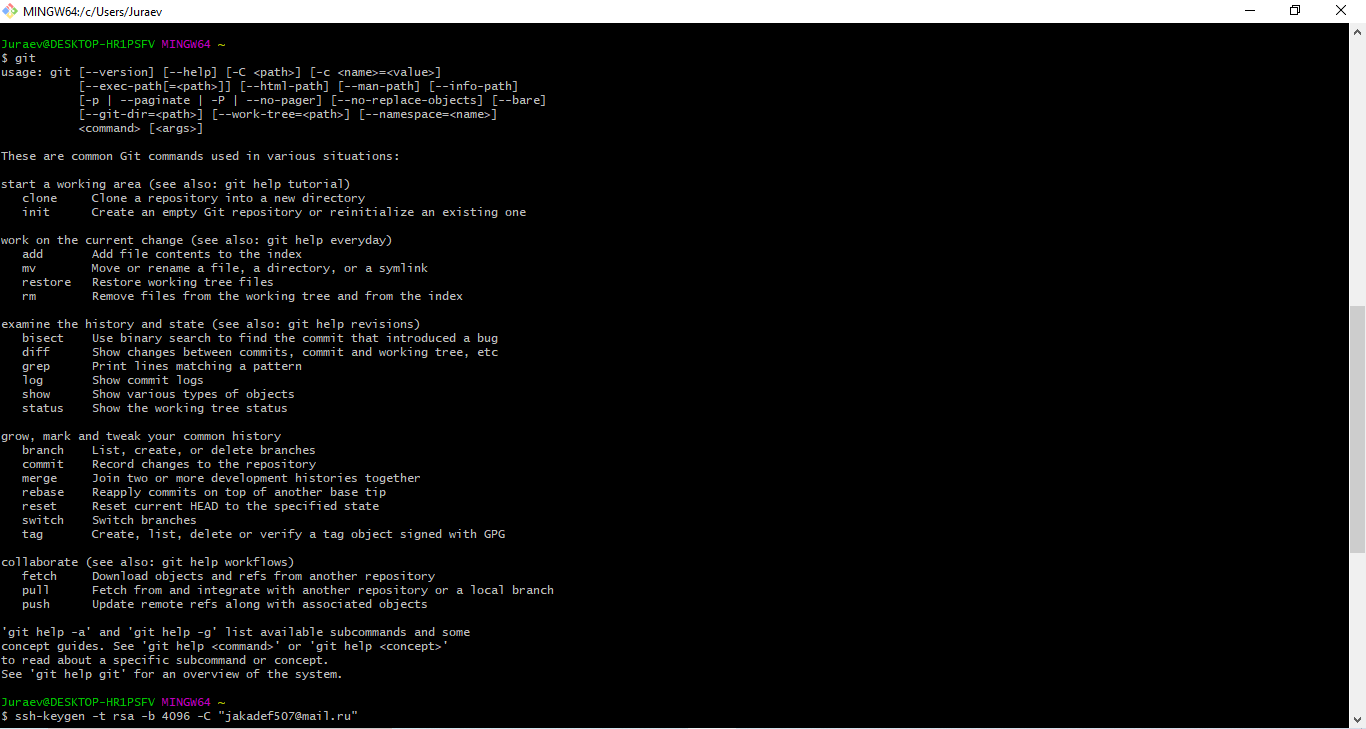
14. Нажимаем кнопку \*Add SSH Key\*

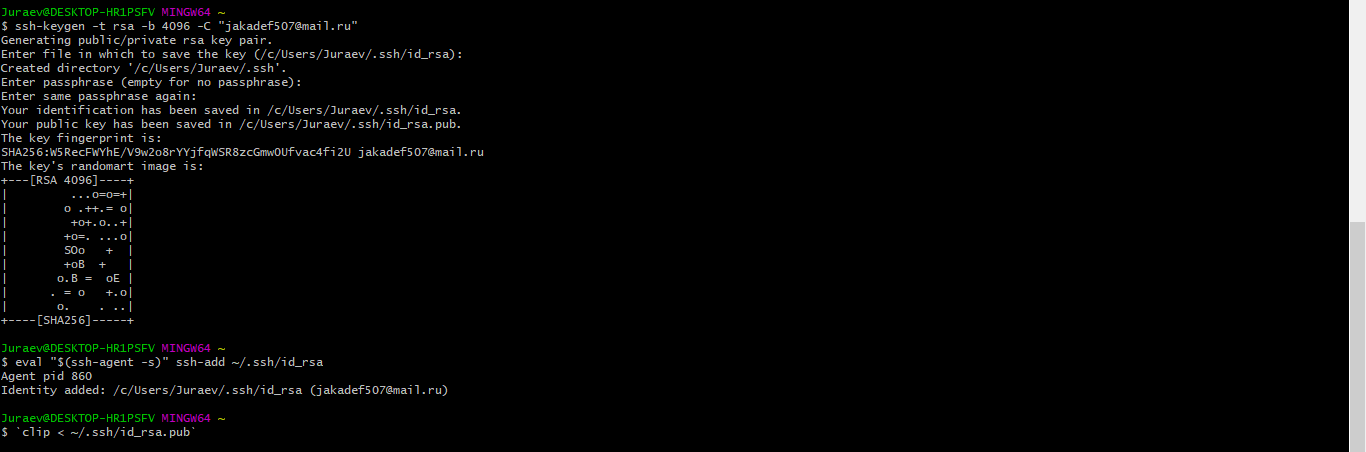
![Add SSH key](https://help.github.com/assets/images/help/settings/ssh-add-key.png)

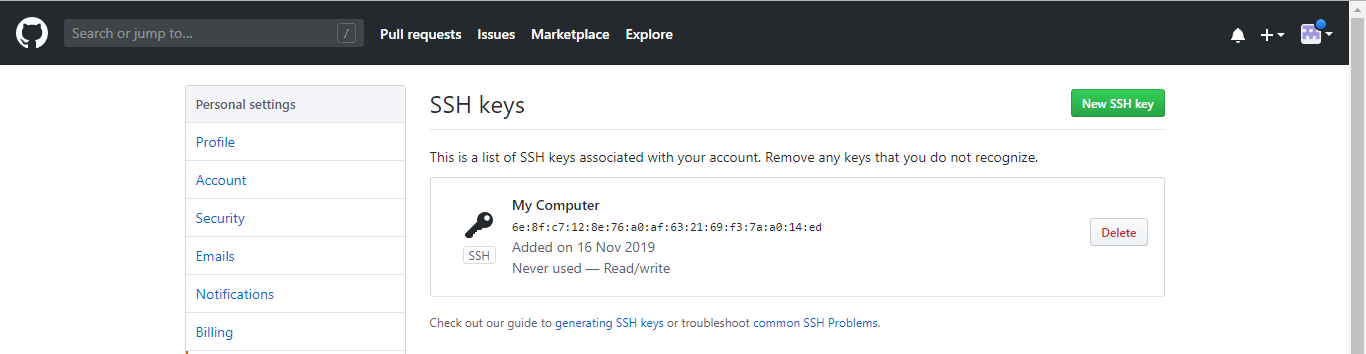
15. Готово!

Мы установили git и настроили его для работы с github. \*\*Молодцы\*\*!!!









# Основы Git - Создание Git-репозитория

## Создание Git-репозитория

Для создания Git-репозитория существуют два основных подхода. Первый подход — импорт в Git уже существующего проекта или каталога. Второй — клонирование уже существующего репозитория с сервера.

### [Создание репозитория в существующем каталоге](https://git-scm.com/book/ru/v1/%D0%9E%D1%81%D0%BD%D0%BE%D0%B2%D1%8B-Git-%D0%A1%D0%BE%D0%B7%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5-Git-%D1%80%D0%B5%D0%BF%D0%BE%D0%B7%D0%B8%D1%82%D0%BE%D1%80%D0%B8%D1%8F#%D0%A1%D0%BE%D0%B7%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5-%D1%80%D0%B5%D0%BF%D0%BE%D0%B7%D0%B8%D1%82%D0%BE%D1%80%D0%B8%D1%8F-%D0%B2-%D1%81%D1%83%D1%89%D0%B5%D1%81%D1%82%D0%B2%D1%83%D1%8E%D1%89%D0%B5%D0%BC-%D0%BA%D0%B0%D1%82%D0%B0%D0%BB%D0%BE%D0%B3%D0%B5)

Если вы собираетесь начать использовать Git для существующего проекта, то вам необходимо перейти в проектный каталог и в командной строке ввести

$ git init

Эта команда создаёт в текущем каталоге новый подкаталог с именем .git содержащий все необходимые файлы репозитория — основу Git-репозитория. На этом этапе ваш проект ещё не находится под версионным контролем. (В главе 9 приведено подробное описание файлов содержащихся в только что созданном вами каталоге .git)

Если вы хотите добавить под версионный контроль существующие файлы (в отличие от пустого каталога), вам стоит проиндексировать эти файлы и осуществить первую фиксацию изменений. Осуществить это вы можете с помощью нескольких команд git add указывающих индексируемые файлы, а затем commit:

$ git add \*.c

$ git add README

$ git commit -m 'initial project version'

Мы разберём, что делают эти команды чуть позже. На данном этапе, у вас есть Git-репозиторий с добавленными файлами и начальным коммитом.

### [Клонирование существующего репозитория](https://git-scm.com/book/ru/v1/%D0%9E%D1%81%D0%BD%D0%BE%D0%B2%D1%8B-Git-%D0%A1%D0%BE%D0%B7%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5-Git-%D1%80%D0%B5%D0%BF%D0%BE%D0%B7%D0%B8%D1%82%D0%BE%D1%80%D0%B8%D1%8F#%D0%9A%D0%BB%D0%BE%D0%BD%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5-%D1%81%D1%83%D1%89%D0%B5%D1%81%D1%82%D0%B2%D1%83%D1%8E%D1%89%D0%B5%D0%B3%D0%BE-%D1%80%D0%B5%D0%BF%D0%BE%D0%B7%D0%B8%D1%82%D0%BE%D1%80%D0%B8%D1%8F)

Если вы желаете получить копию существующего репозитория Git, например, проекта, в котором вы хотите поучаствовать, то вам нужна команда git clone. Если вы знакомы с другими системами контроля версий, такими как Subversion, то заметите, что команда называется clone, а не checkout. Это важное отличие — Git получает копию практически всех данных, что есть на сервере. Каждая версия каждого файла из истории проекта забирается (pulled) с сервера, когда вы выполняете git clone. Фактически, если серверный диск выйдет из строя, вы можете использовать любой из клонов на любом из клиентов, для того чтобы вернуть сервер в то состояние, в котором он находился в момент клонирования (вы можете потерять часть серверных перехватчиков (server-side hooks) и т.п., но все данные, помещённые под версионный контроль, будут сохранены, подробнее см. в главе 4).

Клонирование репозитория осуществляется командой git clone [url]. Например, если вы хотите клонировать библиотеку Ruby Git, известную как Grit, вы можете сделать это следующим образом:

$ git clone git://github.com/schacon/grit.git

Эта команда создаёт каталог с именем grit, инициализирует в нём каталог .git, скачивает все данные для этого репозитория и создаёт (checks out) рабочую копию последней версии. Если вы зайдёте в новый каталог grit, вы увидите в нём проектные файлы, пригодные для работы и использования. Если вы хотите клонировать репозиторий в каталог, отличный от grit, можно это указать в следующем параметре командной строки:

$ git clone git://github.com/schacon/grit.git mygrit

Эта команда делает всё то же самое, что и предыдущая, только результирующий каталог будет назван mygrit.

В Git'е реализовано несколько транспортных протоколов, которые вы можете использовать. В предыдущем примере использовался протокол git://, вы также можете встретить http(s):// или user@server:/path.git, использующий протокол передачи SSH. В главе 4 мы познакомимся со всеми доступными вариантами конфигурации сервера для обеспечения доступа к вашему Git-репозиторию, а также рассмотрим их достоинства и недостатки.