

## Теоретическая часть

**Система контроля версий** — это система, регистрирующая изменения в одном или нескольких файлах с тем, чтобы в дальнейшем была возможность вернуться к определённым старым версиям этих файлов. Чаще всего в системах контроля версий хранятся исходные коды программ, но на самом деле под версионный контроль можно поместить файлы практически любого типа.

Система контроля версий даёт возможность возвращать отдельные файлы к прежнему виду, возвращать к прежнему состоянию весь проект, просматривать происходящие со временем изменения, определять, кто последним вносил изменения во внезапно переставший работать модуль, кто и когда внёс в код какую-то ошибку, и многое другое. Вообще, если, пользуясь системой контроля версий, вы всё испортите или потеряете файлы, всё можно будет легко восстановить. Вдобавок, накладные расходы за всё, что вы получаете, будут очень маленькими.

**Язык разметки** (текста) в компьютерной терминологии — набор символов или последовательностей, вставляемых в текст для передачи информации о его выводе или строении. Принадлежит классу компьютерных **языков**.

**Репозиторий**, хранилище — место, где хранятся и поддерживаются какие-либо данные. Чаще всего данные в репозитории хранятся в виде файлов, доступных для дальнейшего распространения по сети.

### **Git**

Git (произносится «гит») — распределённая система контроля версий. Проект был создан Линусом Торвальдсом для управления разработкой ядра Linux, первая версия выпущена 7 апреля 2005 года. Основные требования к новой системе были следующими:

- ✓ Скорость
- ✓ Простота дизайна
- ✓ Поддержка нелинейной разработки (тысячи параллельных веток)
- ✓ Полная распределённость
- ✓ Возможность эффективной работы с такими большими проектами, как ядро Linux (как по скорости, так и по размеру данных)

С момента рождения Git развивался и эволюционировал, становясь проще и удобнее в использовании, сохраняя при этом свои первоначальные качества. Он невероятно быстр, очень эффективен для больших проектов, а также обладает превосходной системой ветвления для нелинейной разработки

**GitHub** — крупнейший веб-сервис для хостинга IT-проектов и их совместной разработки. Основан на системе контроля версий Git и разработан на Ruby on Rails и Erlang. Сервис абсолютно бесплатен для проектов с открытым исходным кодом и предоставляет им все возможности (включая SSL), а для частных проектов предлагаются различные платные тарифные планы.

Создатели сайта называют GitHub «социальной сетью для разработчиков». Кроме размещения кода, участники могут общаться, комментировать правки друг друга, а также следить за новостями знакомых. С помощью широких возможностей Git программисты могут объединять свои

репозитории — GitHub предлагает удобный интерфейс для этого и может отображать вклад каждого участника в виде дерева.

Первый частный репозиторий был создан 12 января 2008. К концу 2011 года в проекте уже было зарегистрировано более миллиона пользователей и более двух миллионов репозиторий. По состоянию на март 2017 года на сайте существовало более 58 миллионов репозиторий.

**Markdown (маркдаун)** — облегчённый язык разметки созданный с целью написания максимально читабельного и удобного для правки текста, но пригодного для преобразования в языки для продвинутых публикаций. Разработчики Маркдауна ставили своей целью создать язык, код которого будет визуально схож с результатом его выполнения. За основу были взяты принятые правила оформления e-mail сообщений.

**XPath** — это невероятно гибкий, мощный, и вместе с тем сравнительно простой инструмент для навигации по документам XML. Предлагаю перевод руководства по XPath, сделанный на основе руководства консорциума W3C.

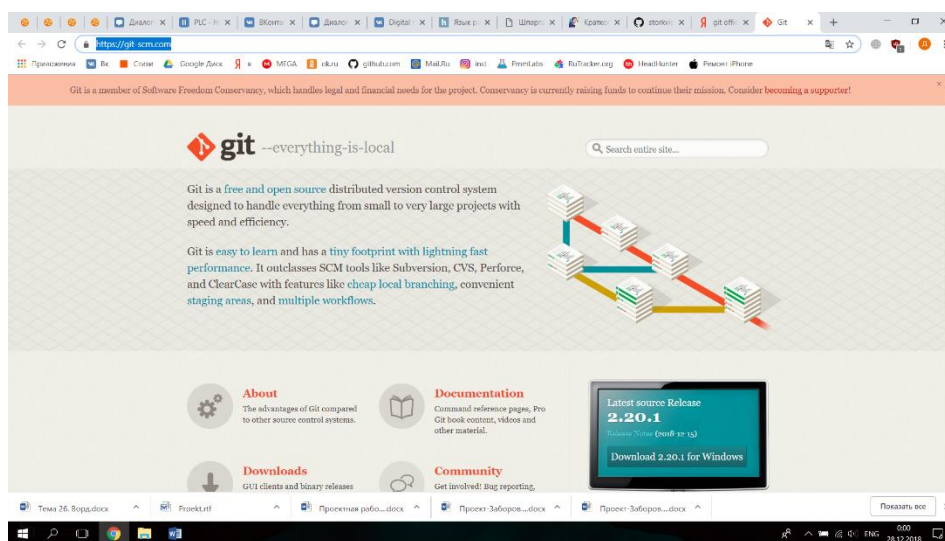
**BaseX** — компактная система управления базами данных XML, разрабатываемая в рамках коллективного проекта на платформе GitHub. Эта система преимущественно применяется для сохранения, запроса и визуализации объёмных XML-документов и XML-коллекций. BaseX может быть использован в различных операционных системах, на основании свободной Open-Source-лицензии.

BaseX предоставляет высоко-стандартную совместимость с W3C-языками XPath и XQuery, а также с обновляющими и полнотекстовыми расширениями. Интегрированный графический интерфейс пользователя позволяет проводить интерактивное обследование и анализ собственных данных, предоставляя возможность выполнения команд языков XPath и XQuery.

## Практика

### **Работа с GitHub**

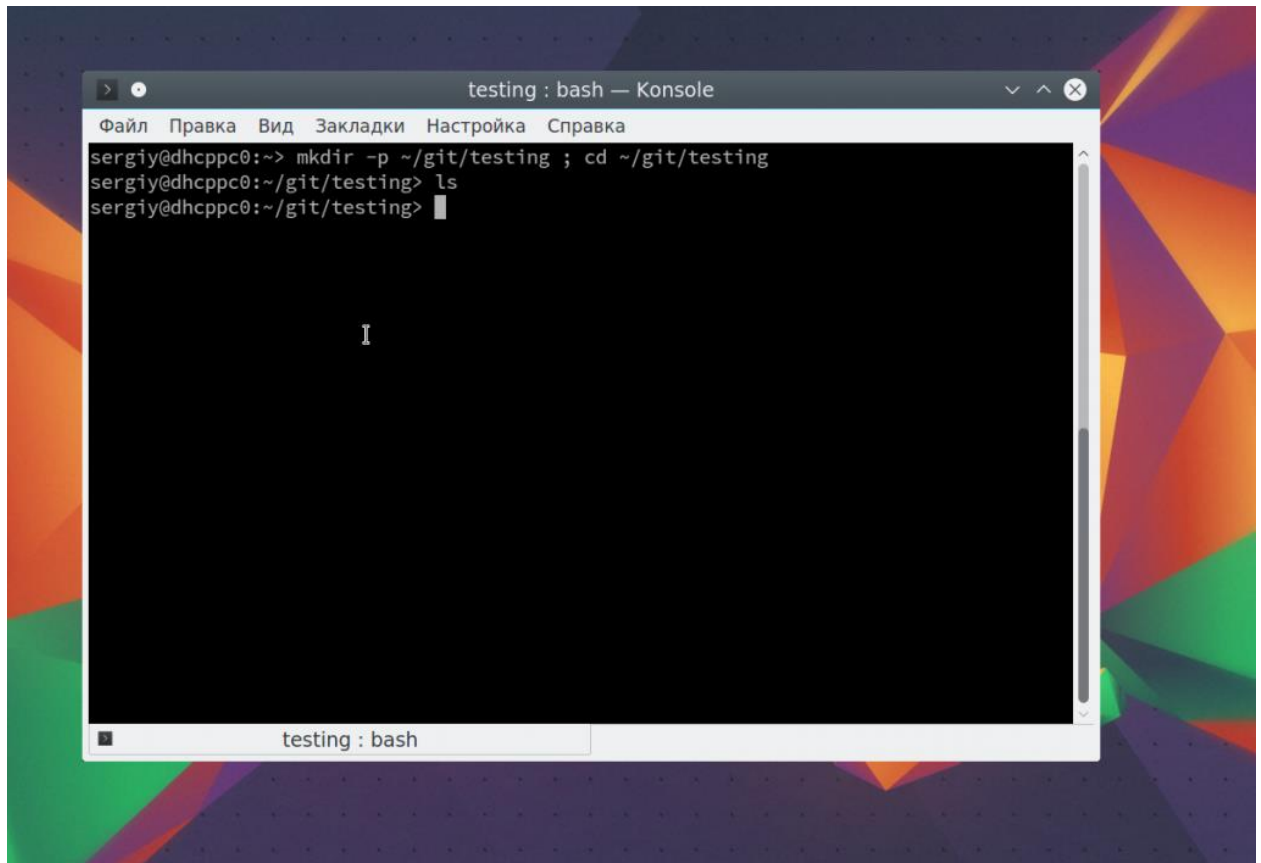
1. Переходим на официальную страницу <https://git-scm.com/> и скачиваем последнюю версию; устанавливаем на диск c/d в папку Git



## СОЗДАНИЕ ПРОЕКТА

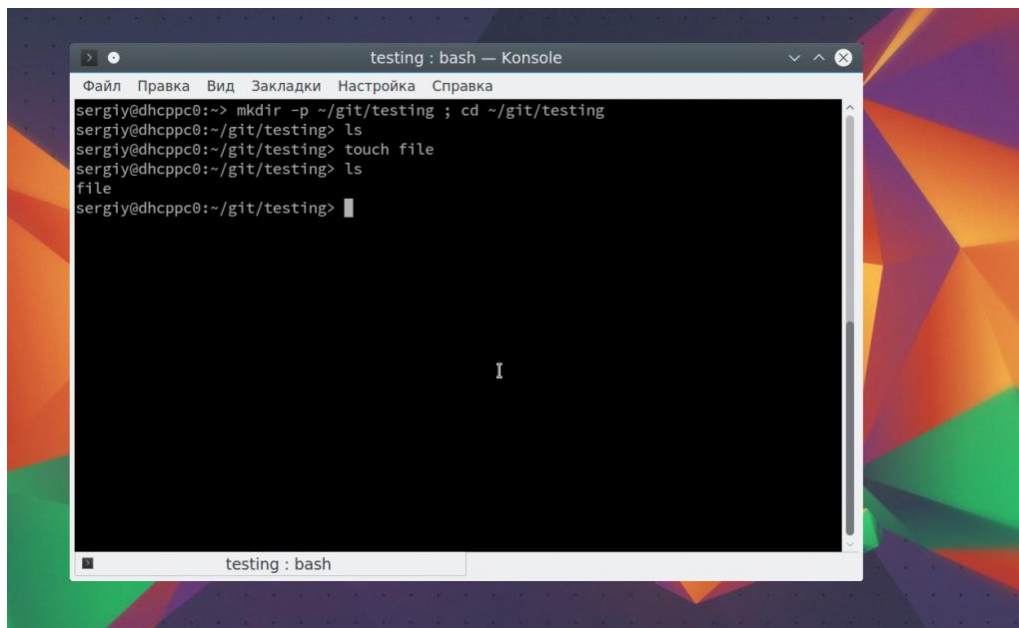
Когда настройка git завершена перейдем к вашему проекту. В самом начале вам достаточно создать папку для файлов проекта. Если вы собираетесь работать над несколькими проектами, создайте папку git в вашем домашнем каталоге, а уже туда поместите папки ваших проектов:

```
mkdir -p ~/git/testing ; cd ~/git/testing
```



Эта команда создаст нужную структуру папок и переводит текущий каталог в только что созданный. Теперь создадим первый файл нашего проекта:

```
touch file
```

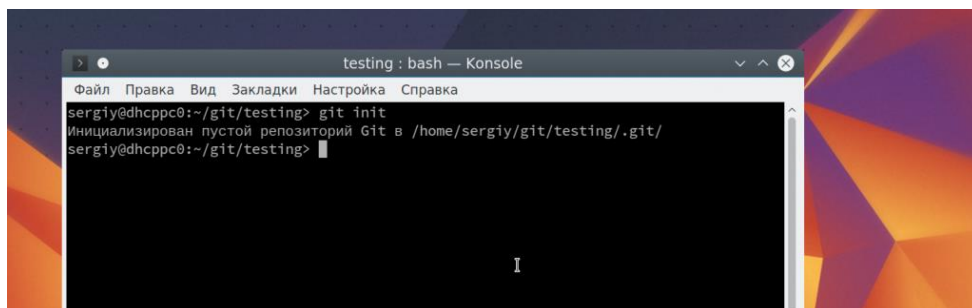
A terminal window titled "testing : bash — Konsole" with a menu bar (Файл, Правка, Вид, Закладки, Настройка, Справка). The terminal shows the following commands and output:

```
sergiy@dhcppc0:~> mkdir -p ~/git/testing ; cd ~/git/testing
sergiy@dhcppc0:~/git/testing> ls
sergiy@dhcppc0:~/git/testing> touch file
sergiy@dhcppc0:~/git/testing> ls
file
sergiy@dhcppc0:~/git/testing>
```

## НАСТРОЙКА ПРОЕКТА В GIT

Перед тем как git начнет отслеживать изменения, нужно подготовить все необходимые конфигурационные файлы. Сначала инициализируем пустой репозиторий в нашей папке:

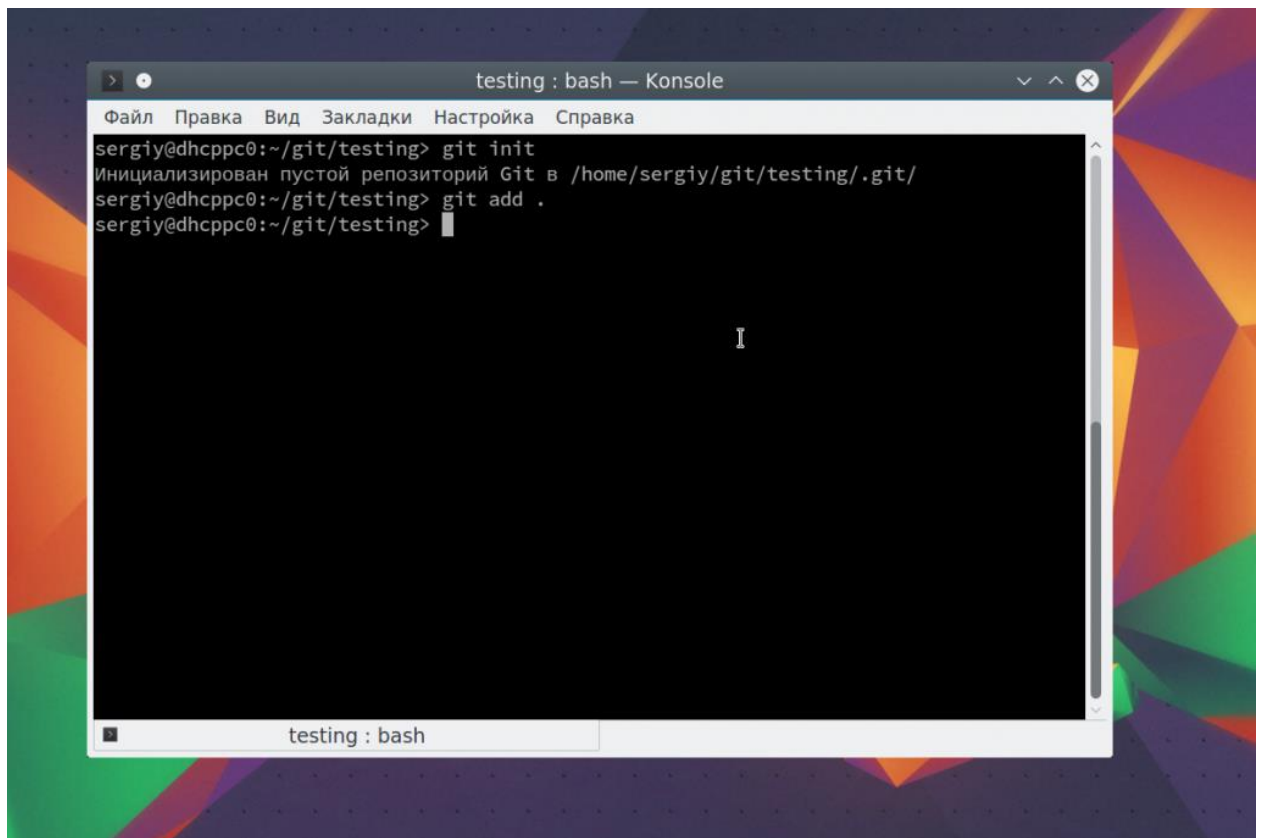
`git init`

A terminal window titled "testing : bash — Konsole" with a menu bar (Файл, Правка, Вид, Закладки, Настройка, Справка). The terminal shows the following commands and output:

```
sergiy@dhcppc0:~/git/testing> git init
Инициализирован пустой репозиторий Git в /home/sergiy/git/testing/.git/
sergiy@dhcppc0:~/git/testing>
```

После того как репозиторий будет создан, вам нужно добавить свои файлы в него. Каждый файл нужно добавлять отдельно или сказать утилите, что необходимо добавить все файлы явно. Пока вы не добавите файл сам он не будет отслеживаться. Новые файлы в будущем тоже нужно добавлять, они не добавляются автоматически. Сначала добавим текущую папку:

`git add .`

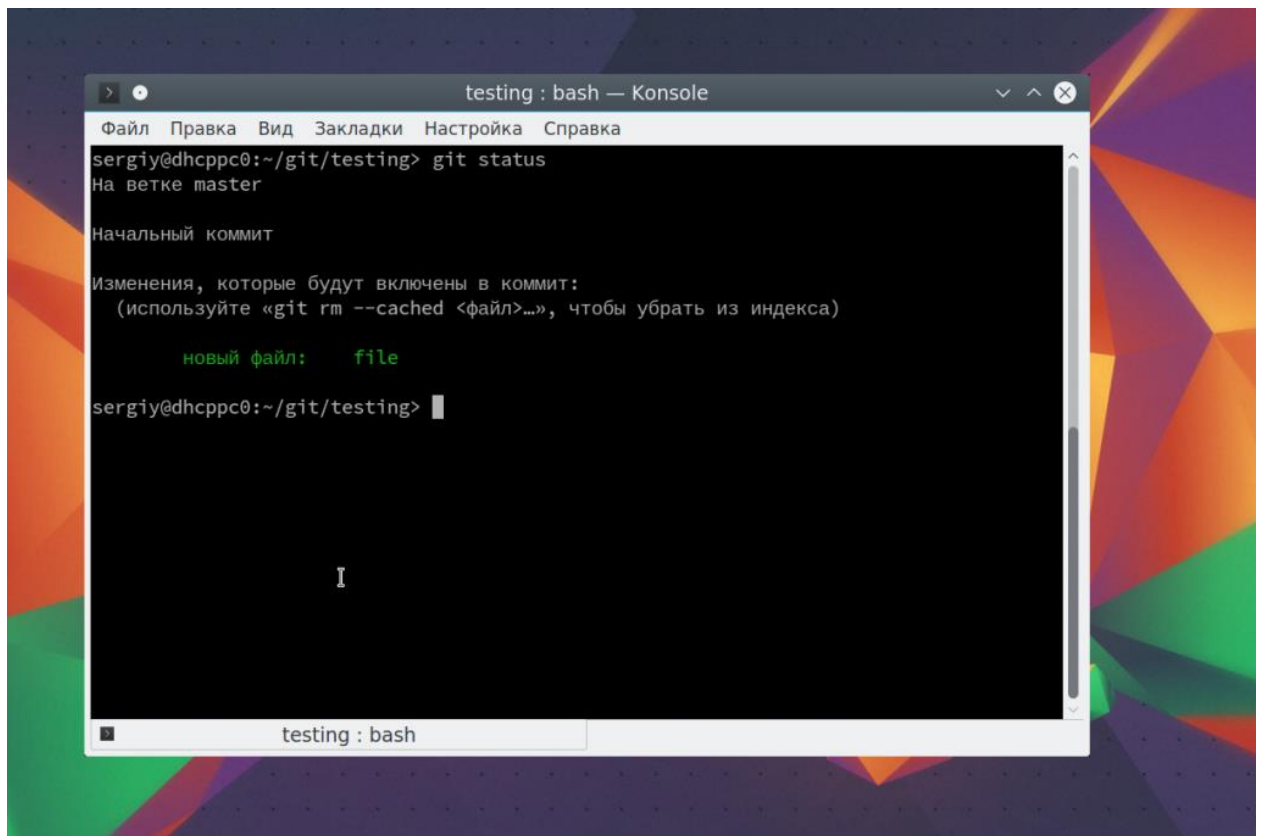


## ФИКСАЦИЯ ИЗМЕНЕНИЙ

Изменения тоже автоматически не отслеживаются. Фиксация изменений выполняется с помощью команды `commit`. Вам нужно указать что было изменено с помощью небольшого комментария, буквально в несколько предложений. Хорошая практика выполнять фиксацию перед каждым серьезным изменением.

Таким образом, вы будете хранить все версии проекта, от самой первой и до текущей, а также сможете знать что, когда и где было изменено. Чтобы создать свой первый коммит выполните:

```
git commit -m "Initial Commit" -a
```

A terminal window titled "testing : bash — Konsole" with a menu bar (Файл, Правка, Вид, Закладки, Настройка, Справка). The prompt is "sergiy@dhcppc0:~/git/testing>". The command "git status" has been executed, showing the following output: "На ветке master", "Начальный коммит", "Изменения, которые будут включены в коммит:", "(используйте «git rm --cached <файл>...», чтобы убрать из индекса)", and "новый файл: file". The prompt "sergiy@dhcppc0:~/git/testing>" is shown again with a cursor.

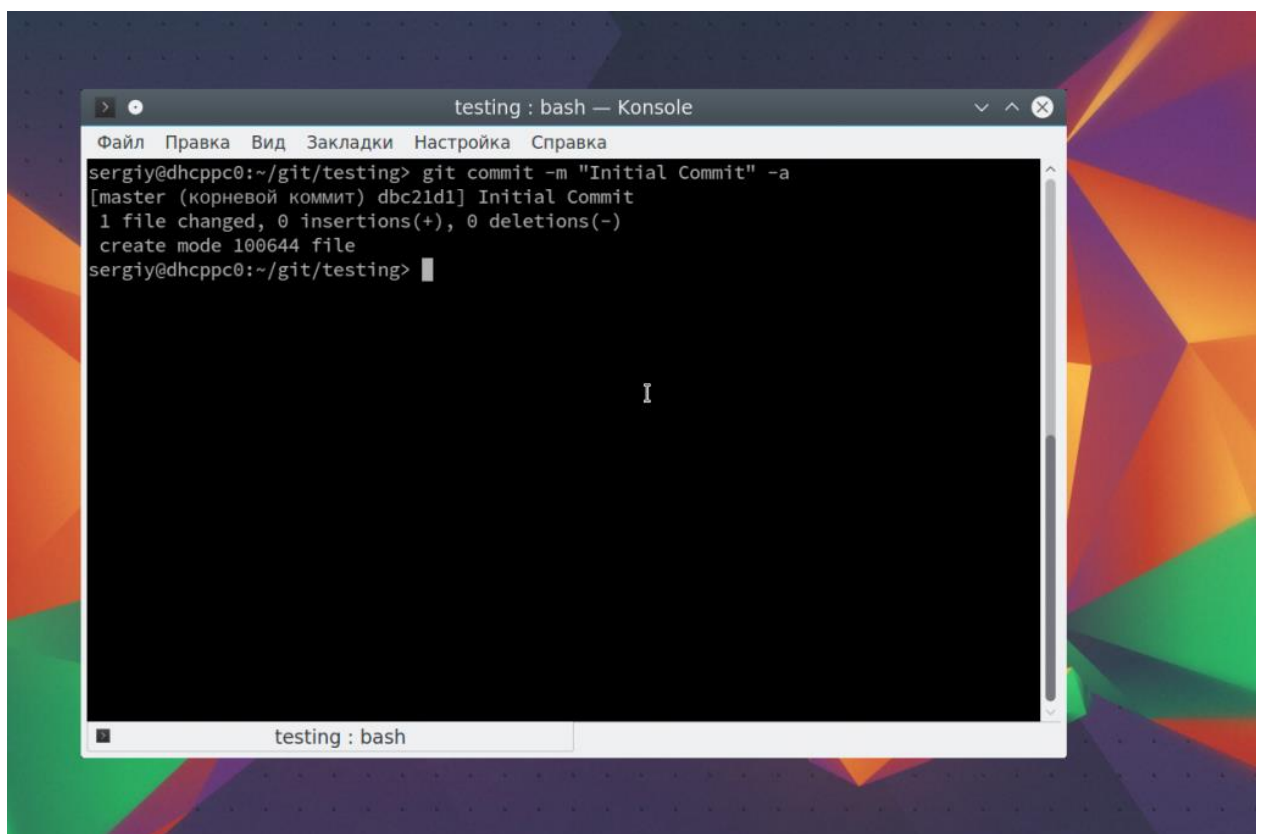
```
testing : bash — Konsole
Файл Правка Вид Закладки Настройка Справка
sergiy@dhcppc0:~/git/testing> git status
На ветке master

Начальный коммит

Изменения, которые будут включены в коммит:
(используйте «git rm --cached <файл>...», чтобы убрать из индекса)

новый файл: file

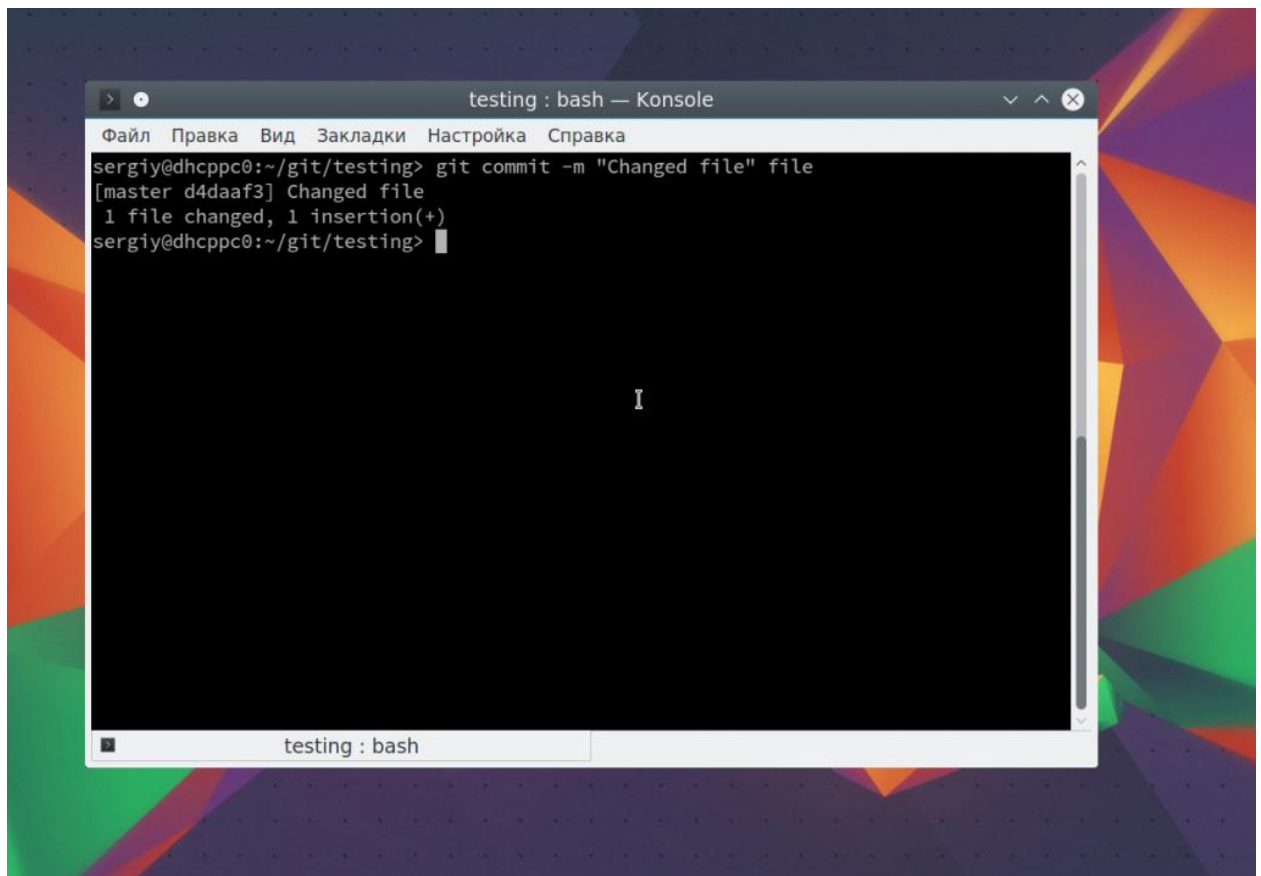
sergiy@dhcppc0:~/git/testing>
```

A terminal window titled "testing : bash — Konsole" with a menu bar (Файл, Правка, Вид, Закладки, Настройка, Справка). The prompt is "sergiy@dhcppc0:~/git/testing>". The command "git commit -m 'Initial Commit' -a" has been executed, showing the following output: "[master (корневой коммит) dbc21d1] Initial Commit", "1 file changed, 0 insertions(+), 0 deletions(-)", and "create mode 100644 file". The prompt "sergiy@dhcppc0:~/git/testing>" is shown again with a cursor.

```
testing : bash — Konsole
Файл Правка Вид Закладки Настройка Справка
sergiy@dhcppc0:~/git/testing> git commit -m "Initial Commit" -a
[master (корневой коммит) dbc21d1] Initial Commit
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 file
sergiy@dhcppc0:~/git/testing>
```

Команде необходимо передать два параметра, первый - это -m, ваш комментарий, второй -a, означает, что нужно применить действие ко всем измененным файлам. Для первого раза используется этот параметр, но обычно вам нужно указать измененные файлы или каталоги. Например, можно делать так:

```
git commit -m "Changed file" file
```



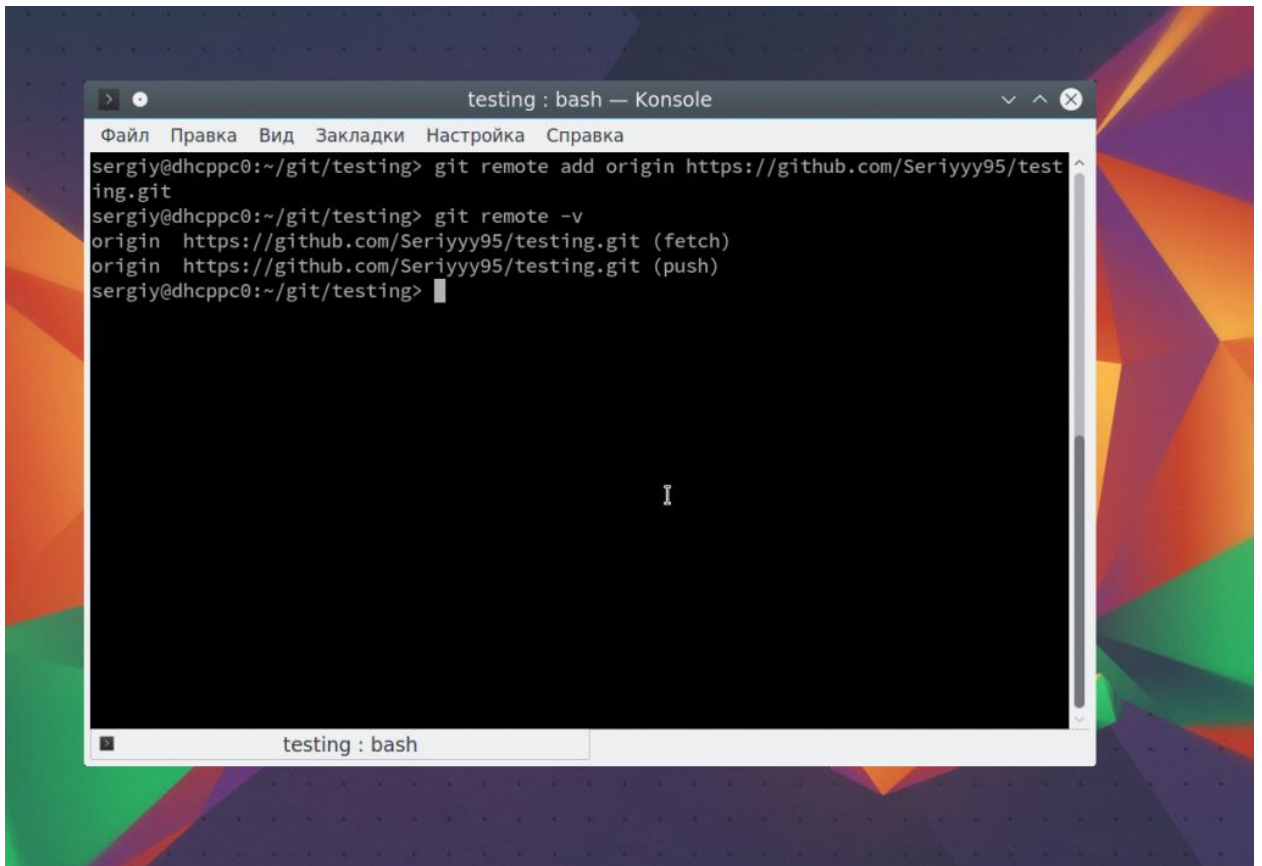
## ОТПРАВКА ИЗМЕНЕНИЙ

До этого момента мы делали все в локальном репозитории. Вы можете использовать git локально, если нужен только контроль версий, но иногда нужно обменяться информацией с другими разработчиками и отправить данные в удаленный репозиторий.

Сначала нужно добавить удаленный репозиторий с помощью команды `remote`. Для этого нужно передать ей URL:

```
git remote add origin https://github.com/Seriyyy95/testing.git
```



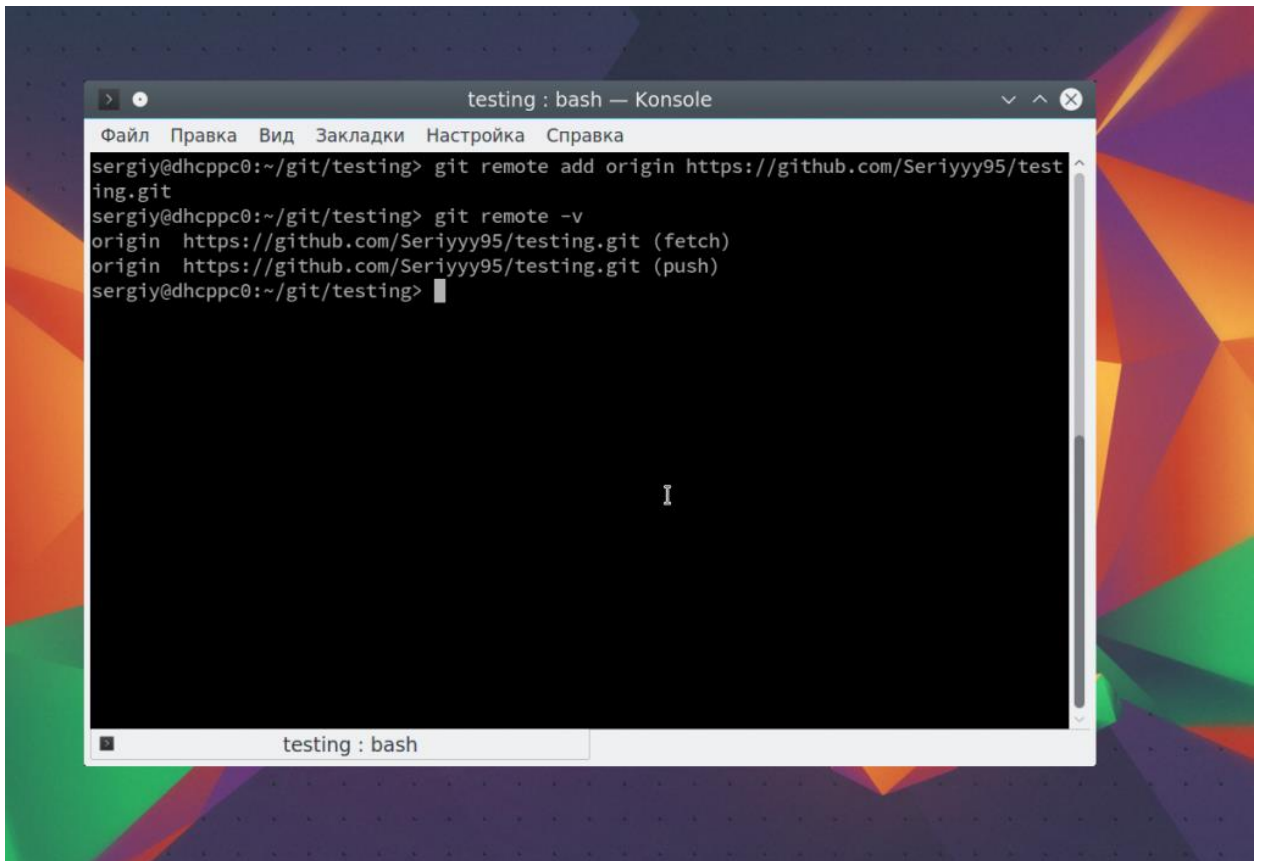
A terminal window titled "testing : bash — Konsole" with a menu bar in Russian (Файл, Правка, Вид, Закладки, Настройка, Справка). The terminal shows the following commands and output:

```
sergiy@dhcppc0:~/git/testing> git remote add origin https://github.com/Seriyyy95/testing.git
sergiy@dhcppc0:~/git/testing> git remote -v
origin https://github.com/Seriyyy95/testing.git (fetch)
origin https://github.com/Seriyyy95/testing.git (push)
sergiy@dhcppc0:~/git/testing>
```

The terminal has a scrollbar on the right and a status bar at the bottom showing "testing : bash".

Затем можно посмотреть список удаленных репозиторий:

`git remote -v`

A terminal window titled "testing : bash — Konsole" with a menu bar in Russian (Файл, Правка, Вид, Закладки, Настройка, Справка). The terminal shows the following commands and output:

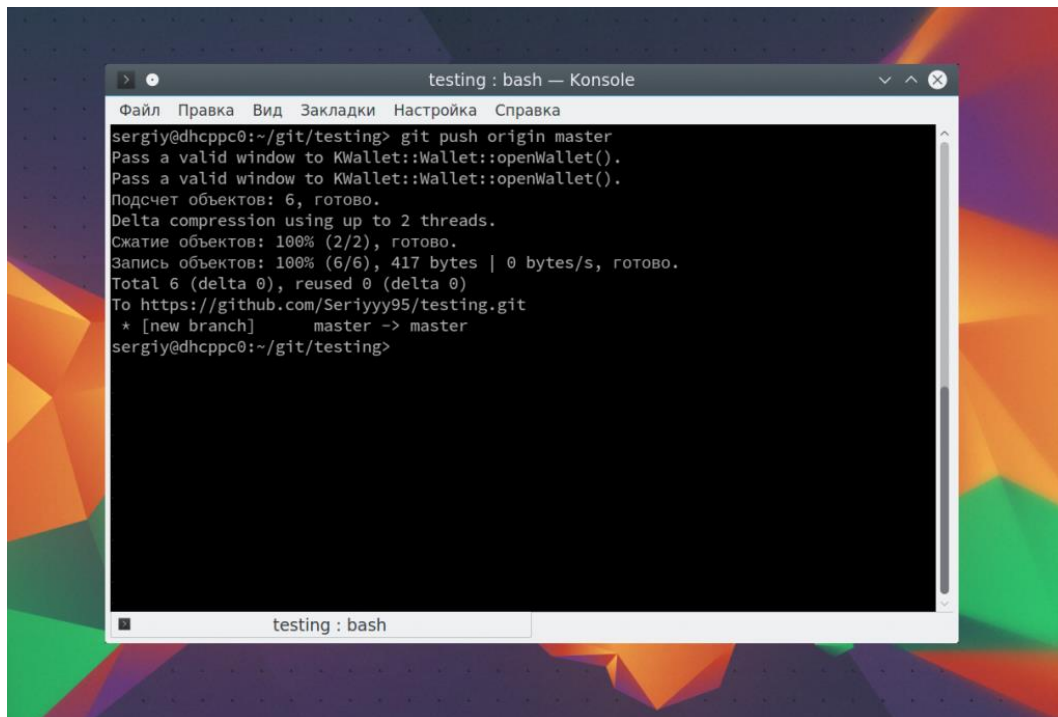
```
sergiy@dhcppc0:~/git/testing> git remote add origin https://github.com/Seriyyy95/testing.git
sergiy@dhcppc0:~/git/testing> git remote -v
origin https://github.com/Seriyyy95/testing.git (fetch)
origin https://github.com/Seriyyy95/testing.git (push)
sergiy@dhcppc0:~/git/testing>
```

The terminal has a scrollbar on the right and a status bar at the bottom showing "testing : bash".



Вы можете использовать не только github сервера, но и любые другие. Теперь для отправки ваших изменений используйте такую команду:

```
git push origin master
```

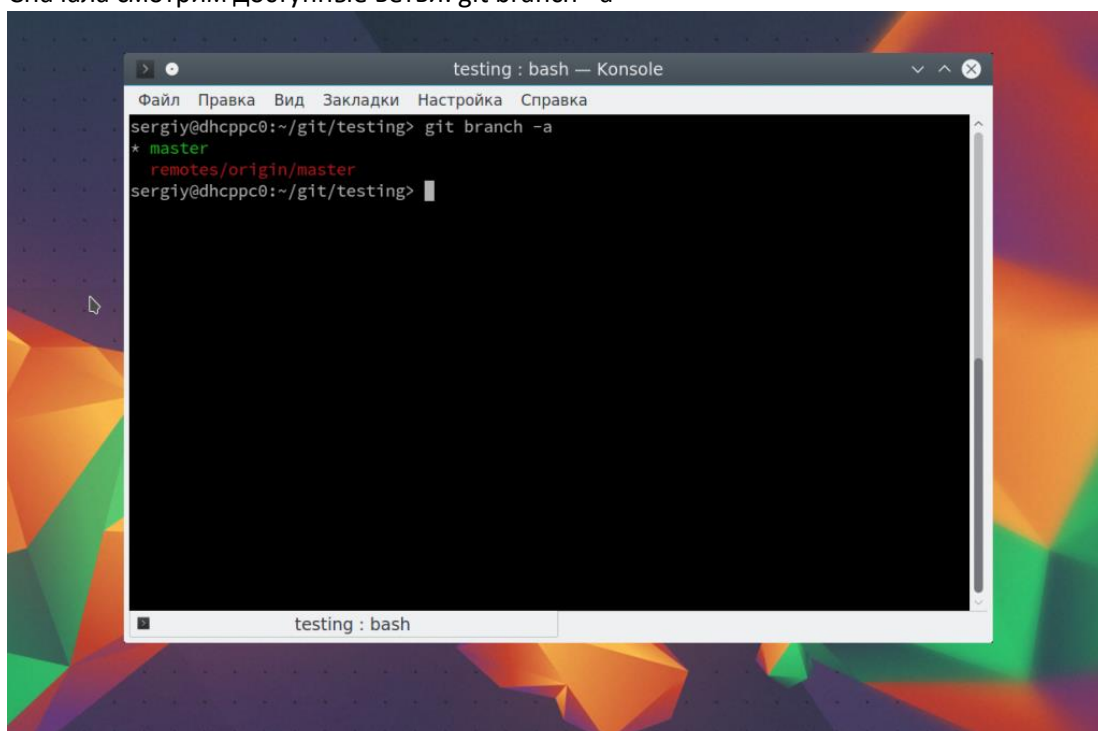
A terminal window titled "testing : bash — Konsole" with a menu bar (Файл, Правка, Вид, Закладки, Настройка, Справка). The terminal shows the execution of the command `git push origin master`. The output includes: "Pass a valid window to KWallet::Wallet::openWallet().", "Подсчет объектов: 6, готово.", "Delta compression using up to 2 threads.", "Сжатие объектов: 100% (2/2), готово.", "Запись объектов: 100% (6/6), 417 bytes | 0 bytes/s, готово.", "Total 6 (delta 0), reused 0 (delta 0)", "To https://github.com/Seriyyy95/testing.git", and "\* [new branch] master -> master". The prompt is `sergiy@dhcppc0:~/git/testing>`.

```
sergiy@dhcppc0:~/git/testing> git push origin master
Pass a valid window to KWallet::Wallet::openWallet().
Pass a valid window to KWallet::Wallet::openWallet().
Подсчет объектов: 6, готово.
Delta compression using up to 2 threads.
Сжатие объектов: 100% (2/2), готово.
Запись объектов: 100% (6/6), 417 bytes | 0 bytes/s, готово.
Total 6 (delta 0), reused 0 (delta 0)
To https://github.com/Seriyyy95/testing.git
* [new branch] master -> master
sergiy@dhcppc0:~/git/testing>
```

Команда push указывает, что нужно отправить данные в удаленный репозиторий, origin - наш настроенный репозиторий, а master - ветвь.

## УПРАВЛЕНИЕ ВЕТВЯМИ

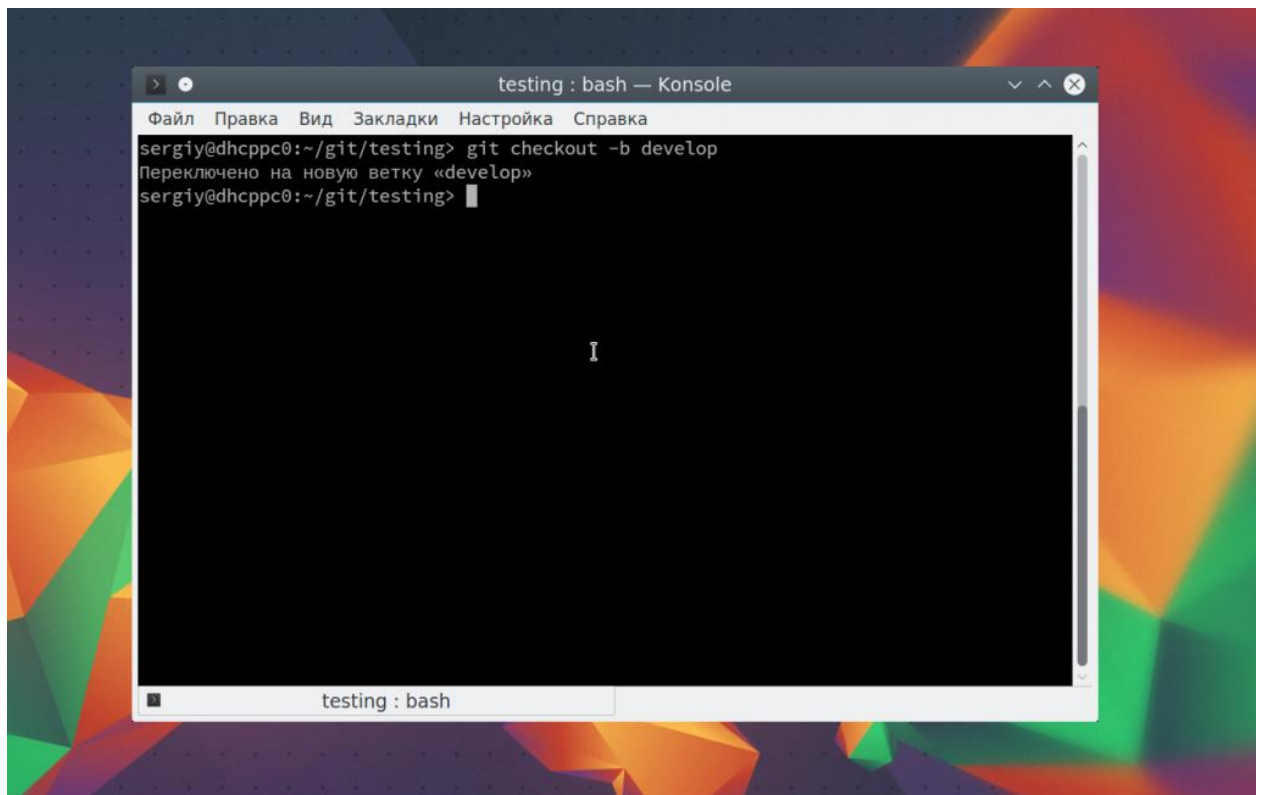
Для простых проектов достаточно одной ветви. Но если проект большой и он имеет несколько версий, в том числе тестовую, то может понадобиться создать для каждой из них отдельную ветвь. Сначала смотрим доступные ветви: `git branch -a`

A terminal window titled "testing : bash — Konsole" with a menu bar (Файл, Правка, Вид, Закладки, Настройка, Справка). The terminal shows the execution of the command `git branch -a`. The output lists two branches: `* master` and `remotes/origin/master`. The prompt is `sergiy@dhcppc0:~/git/testing>`.

```
sergiy@dhcppc0:~/git/testing> git branch -a
* master
remotes/origin/master
sergiy@dhcppc0:~/git/testing>
```

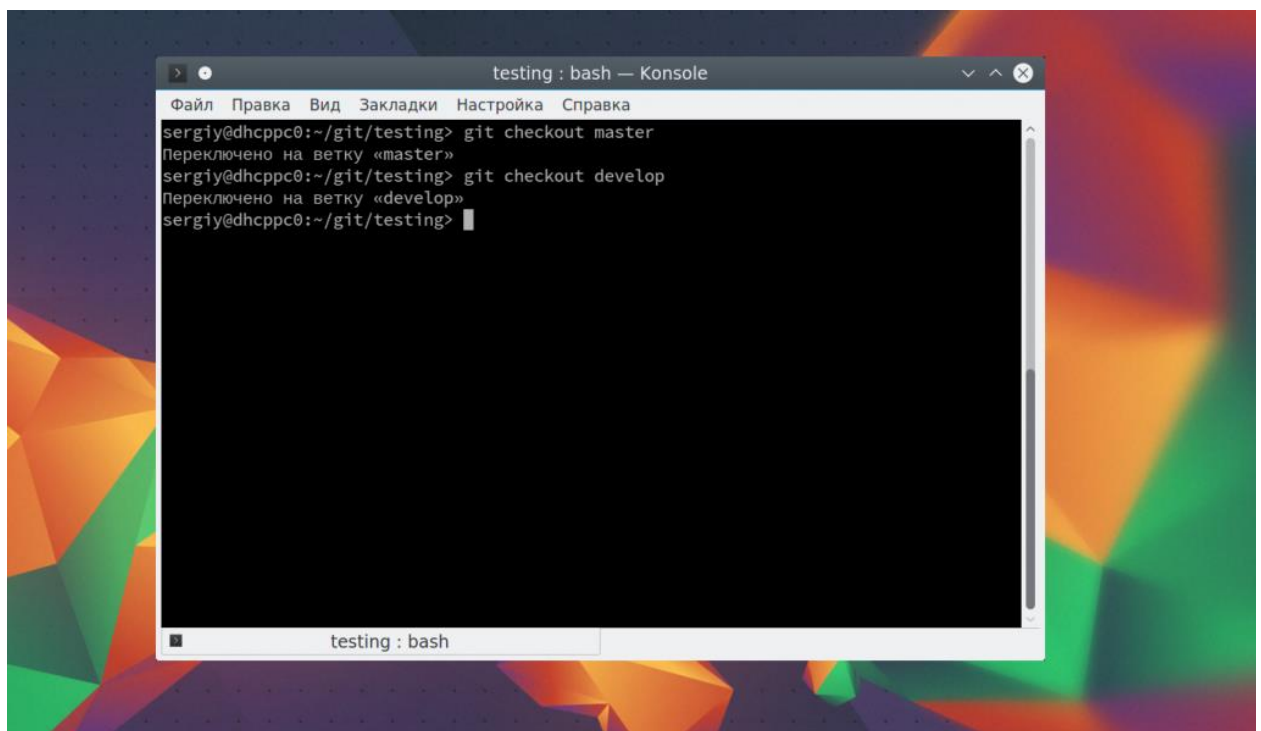
Опция -a указывает что нужно вывести все ветви, даже не синхронизированные. Звездочка указывает на активную ветвь. Теперь создадим ветвь для разработки с помощью команды checkout:

```
git checkout -b develop
```



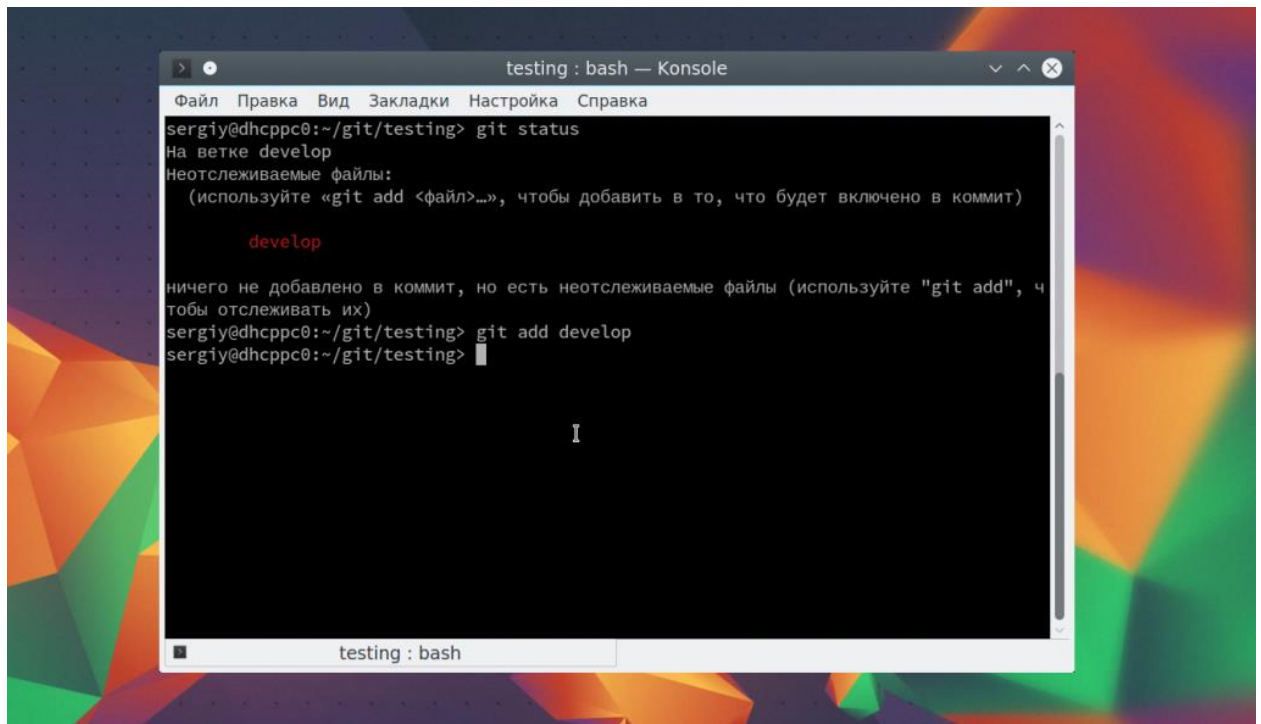
Переключаться между ветвями можно тоже с помощью той же команды:

```
git checkout master  
git checkout develop
```



Теперь создадим еще один файл:

`touch develop`

A screenshot of a terminal window titled "testing : bash — Konsole". The window shows the output of the command `git status`. The output indicates that the user is on the `develop` branch and that there are untracked files. The terminal text is as follows:

```
sergiy@dhcppc0:~/git/testing> git status
На ветке develop
Неотслеживаемые файлы:
  (используйте «git add <файл>...», чтобы добавить в то, что будет включено в коммит)

    develop

ничего не добавлено в коммит, но есть неотслеживаемые файлы (используйте "git add", чтобы отслеживать их)
sergiy@dhcppc0:~/git/testing> git add develop
sergiy@dhcppc0:~/git/testing> 
```

И добавим его в нашу новую ветвь `develop`:

`git add develop`

## Работа с языком разметки Markdown

<http://daringfireball.net/projects/markdown/>

- ✓ В месте вывода содержания укажите тег [TOC]. В результате, вы получите список как сверху
- ✓ Поместите в нужном месте якорь: `<a name="abcd"></a>`, и, там где это необходимо, ссылку на этот якорь: `[Текст ссылки](#abcd)`.
- ✓ Чтобы в маркдауне поставить тэг Читать далее, достаточно в строчке написать `<!--more-->`
- ✓ Для того чтобы написать заголовок в Markdown, необходимо использовать знак # (хэш). Если необходимо несколько уровней заголовков, h1 — h6, нужно изменить количество хэшей (#) перед текстом заголовка.
  - # H1
  - ## H2
  - ### H3
  - #### H4
  - ##### H5
  - ##### H6Альтернативные теги для H1 и H2 (знаки равно или минусы под заголовком + пустая строчка над заголовком)

- ✓ Markdown поддерживает два вида списков: маркерованный и нумерованный. Для организации маркерованного списка используются знаки \*, + и -. От них зависит вид маркеров. Чтоб сделать многоуровневый список, нужно будет сделать отступы (4 или 8 пробелов).

1. Первый пункт списка

2. Второй пункт

...\* Немаркерованный вложенный подпункт.

1. Номер не имеет значения, нужно чтобы это была просто любая цифра

...1. Маркерованный вложенный подпункт.

4. Тут другой пункт.

...Можно вставлять в список отдельный абзац. Но, обратите внимание на пустую строку сверху и пробелы перед строчкой.

...Чтобы вставить абзац без пустой строки сверху, нужно в конце строки поставить два пробела...

...Обратите внимания что эта строчка отдельная, хотя находится в том же абзаце...

...В этой строчке пробелы в конце строки не требуются.

\* Для нумерованных списков можно использовать звёздочки

- Или минусы

+ Или плюсы

1. Первый пункт списка

2. Второй пункт

- Немаркерованный вложенный подпункт.

3. Номер не имеет значения, нужно чтобы это была просто любая цифра

1. Маркерованный вложенный подпункт.

4. Тут другой пункт.

Можно вставлять в список отдельный абзац. Но, обратите внимание на пустую строку сверху и пробелы перед строчкой.

Чтобы вставить абзац без пустой строки сверху, нужно в конце строки поставить два пробела. Обратите внимания что эта строчка отдельная, хотя находится в том же абзаце. В этой строчке пробелы в конце строки не требуются.

- ✓ Для нумерованных списков можно использовать звёздочки
- ✓ Или минусы
- ✓ Или плюсы

Создать горизонтальную линию можно поместив три или больше звездочек \*, минусов — или подчеркиваний \_ на отдельной строке.

```
***
```

но вы можете разделять символы пробелами, чтобы сделать линию более очевидной в процессе редактирования документа.

```
* * * * *
```

```
-----
```

```
_ _ _ _ _
```

Каждая из приведенных выше строк даст одинаковый результат. Вот такой:

---

## Абзацы, параграфы

В маркдаун все блочные элементы (заголовки, списки, абзацы и т.д.) — все, что визуально начинается с новой строки отбивается пустой строкой. Для вставки пустой строки необходимо два раза поставить символ переноса строки (нажать на Enter)

---

## Перенос строки

Перенос строки всегда игнорируется. Это делается для того, чтобы, например, была возможность при наборе держать определенную длину строки, скажем в 80 символов.

Т.к. маркдаун создавался с целью быть удобочитаемым, его нередко используют в текстовых документах и для комфорта читателя лучше не давать тексту расплываться соплёй по экрану.

Для переноса строки внутри абзаца нужно использовать два пробела .. в конце строки.

---

## Курсивное и жирное выделение текста

Для курсива необходимо поставить знаки \* *вокруг текста*. Для жирного начертания обрaмим текст **двумя звездочками**, а для **жирного курсива** — ***тремя***. Альтернативный синтаксис — использование знака \_ по тем же правилам.

```
__жирный курсив__
```

---

## Выделение текст или код

Выделить текст или часть кода можно с помощью символа гравис (обратный апостроф) `

Выделить `текст` или `часть кода` можно с помощью символа гравис (обратный апостроф) `

Для выделения блока кода используют три символа в строчке до и после блока кода:

```
```  
$a = 5;  
$b = 3;  
$c = $a + $b;  
```
```

---

## Отмена форматирования

Если вы хотите, чтобы введенные теги HTML, код, теги маркдауна отображались в точности как вы их написали, то начните каждую строку с четырех пробелов.

....тут должен быть **\*\*жирный текст\*\***, но я поставил 4 пробела в начале стрки

---

## Цитаты

Для обозначения цитат достаточно поставить знак > в начале строки

```
> Это простая цитата..  
Состоящая из двух строк
```

*Это простая цитата*  
*Состоящая из двух строк*

(напомню, для переноса строки я использую два пробела в конце строчки)  
Цитаты можно вкладывать друг в друга:

```
> Это просто цитата  
  
>  
  
> > А это уже вложенная цитата
```

*Это просто цитата*

*А это уже вложенная цитата*

---

## Ссылки

Существует два варианта оформления ссылок. Первый — обычная вставка в текст:

```
[Текст ссылки](адрес "Описание")
```

#### Текст ссылки

и второй вариант — оформление ссылки в виде сноски. Когда в текст вставляется конструкция вида:

```
[Текст ссылки][Тег1]
```

... Указывающая, что именно в этом место будет располагаться ссылка, а где-нибудь ниже её описание:

```
[Тег1][Адрес ссылки]
```

Результат выполнения будет аналогичен первому варианту, но такое оформление удобнее с точки зрения дальнейшей поддержки и редактирования.

```
I get 10 times more traffic from [Google][1] than from  
[Yahoo][2] or [MSN][3].
```

```
[1]: http://google.com/      "Google"
```

```
[2]: http://search.yahoo.com/ "Yahoo Search"
```

```
[3]: http://search.msn.com/   "MSN Search"
```

I get 10 times more traffic from Google than from  
Yahoo or MSN.

---

## Сноски

Сноски и примечания<sup>1</sup> задаются так<sup>2</sup>:

```
Сноски и примечания[^1] задаются так[^2]:
```

```
// пустая строка обязательно
```

```
[^1]: Все сноски отображаются в конце страницы
```

```
[^2]: Просто не так ли?)
```

---

## Изображения

Изображения помещаются на страницу также, как и ссылки, с одним отличием: в начале записи используется знак !

```
![Alt-текст](Путь к файлу "Подпись")
```



Вставка реального изображения может выглядеть как-то так:

```
![mountains](/img/mountain.png "Пейзаж с горами")
```

Создадим картинку, которая является еще и ссылкой на какую-нибудь страницу в сети:

```
[Поисковая система Google][google]
```

```
![Логотип Google][logo]
```

Выше приведены обычная ссылка и обычное изображение. Вы можете поместите картинку туда, где указан текст ссылки, например:

```
[![Логотип Google][logo]][google]
```

Нужно помнить, что приведенные выше ссылки должны быть определены где-либо в документе:

```
[logo]: http://www.google.com/images/logo.gif
```

```
[google]: http://www.google.com/ "щелкните, чтобы посетить Google.com"
```

Такое совмещение создает картинку, щелчок по которой переместит вас на указанную ссылку:



---

## Таблицы (работает не со всеми плагинами wordpress)

Создание таблиц с Markdown намного нагляднее, чем в HTML. Форматирование интуитивно понятно, добавлю только что для выравнивания текста внутри ячеек используются знаки : в строке, отделяющей заголовок от основной таблицы.

Item	Value	Quantity
:----- :----- :-----:		
Computer	1600	3
Phone	12	2
Pipe	1	1

```
Item | Value | Quantity
:---|:---|:---
Computer | 1600 | 3
```

## Спецсимволы

Если вы хотите отобразить любой из специальных символов Markdown вместо того, чтобы использовать его для форматирования, просто поставьте перед ним символ обратной косой черты (\). Сама косая черта не отображается, однако следующий за ней символ будет показан как есть: \\*

---

## Списки определений

Списки определений содержат термины и их описания. Это выглядит подобно словарю. Ниже простой пример:

**Moodle**

: Хорошо известная платформа для онлайн-обучения

**PHP**

: Язык программирования.

Часто используется для разработки интерактивных веб-приложений.

**Moodle**

Хорошо известная платформа для онлайн-обучения

**PHP**

Язык программирования.

Часто используется для разработки интерактивных веб-приложений.

---

## Спойлер (работает не со всеми плагинами wordpress)

Скрываем большие объемы текстовой и графической информации с помощью Advanced Spoiler, который уже 2 года не обновляется и работает не везде.

[spoiler]тут текст[/spoiler]

[spoiler]тут текст[/spoiler]

---

## Использование HTML

Если вы являетесь специалистом в HTML, то можете обнаружить, что Markdown не дает вам всех возможностей, которые вы бы хотели. К счастью Markdown разрабатывался с учетом этого и позволяет вставлять теги HTML непосредственно в формируемый текст.

\* `<small>мелкий текст</small>`

\* `<big>крупный текст</big>`

- мелкий текст
- крупный текст

Имейте в виду, что HTML-разметка сосуществует с разметкой Markdown. Это освобождает вас от необходимости использовать HTML для основных элементов оформления, таких как параграфы, списки и т.п., однако в необходимых случаях позволяет использовать все возможности HTML.

---

## Рекомендуемый мной онлайн редактор Markdown

<http://dillinger.io/>

---

## Рекомендуемый плагин для wordpress

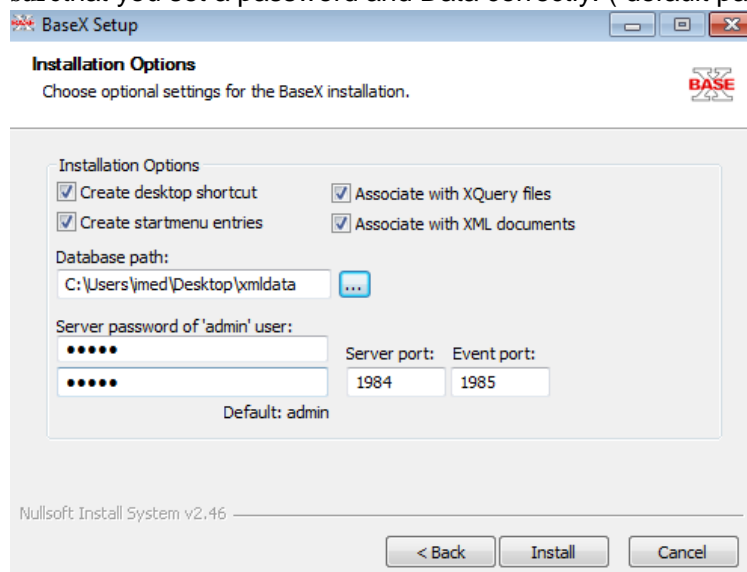
- WP — Markdown on Save Improved
- Markdown Shortcode — позволяет вставить синтаксис маркдауна в части статьи. Код обрамляется тэгами:

**via shortcode**

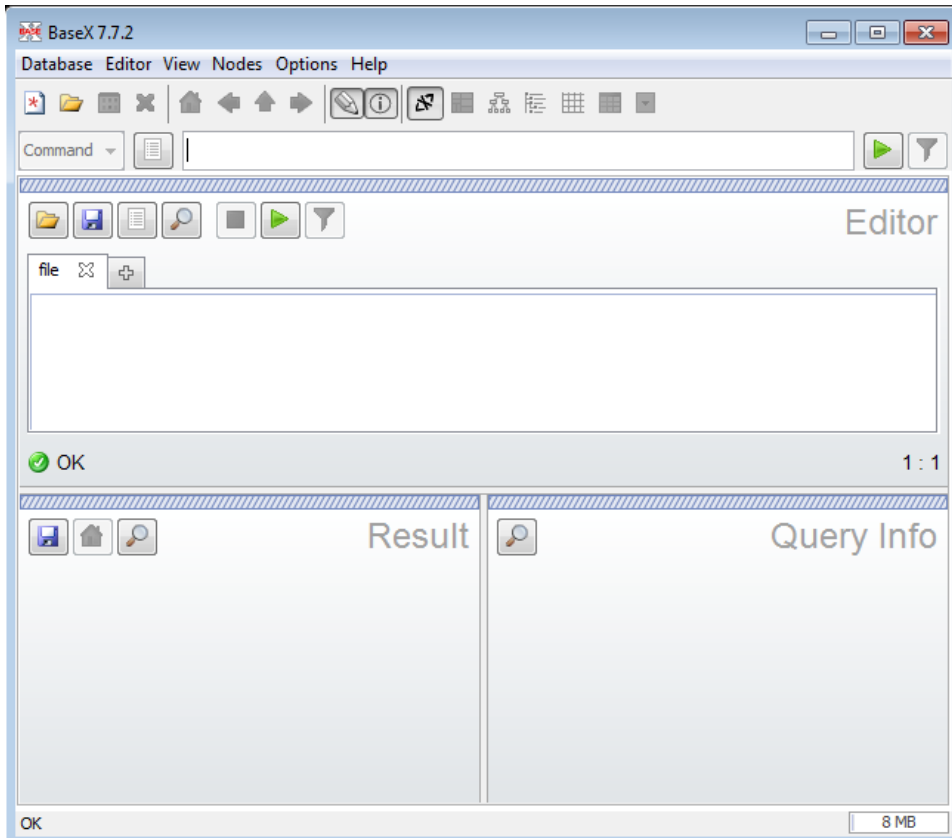
### Работа с BaseX

Скачать можно здесь <http://files.basex.org/releases/7.7.2/BaseX772.exe>

- ✓ Double Click to Install, Follow on screen instructions to complete the installation. **Make sure** that you set a password and Data correctly. ( default password is admin ).



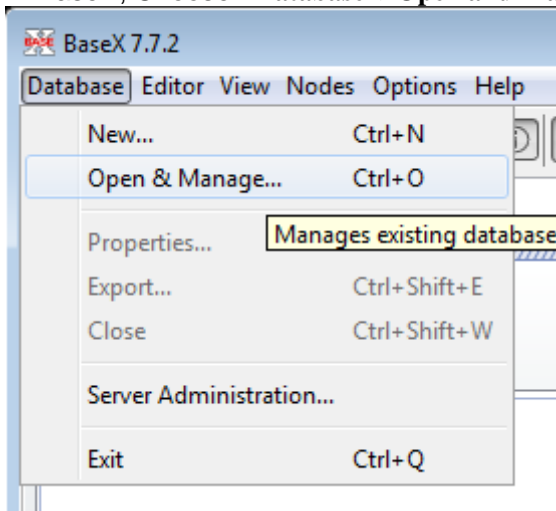
- ✓ Once completed the installation, Double click on the icon for BaseX to launch it. BaseX should look as below:



- ✓ Download the following XML file (**factbook.xml**) that can be used for testing. Save the on your PC, preferably with the XML data folder you have created or chosen within step 2.

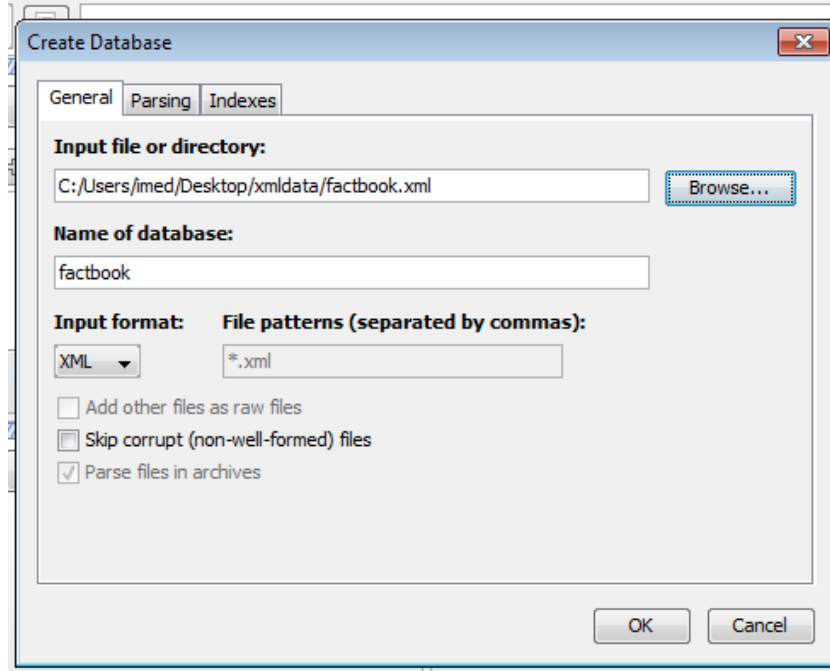
#### **Download factbook.xml**

- ✓ In BaseX, Choose : **Database→Open and Manage ....** as shown below,

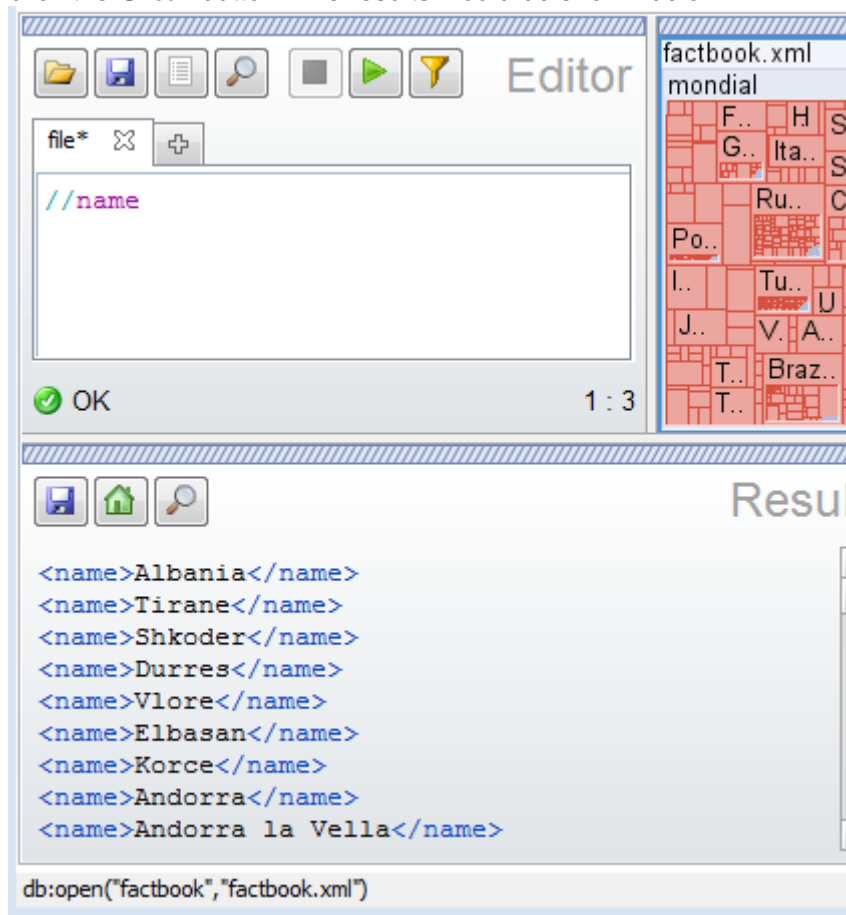


- ✓ Choose **Yes** if asked that no database exists

- ✓ Browse for the **factbook.xml** file that you have downloaded. Click Ok once done.



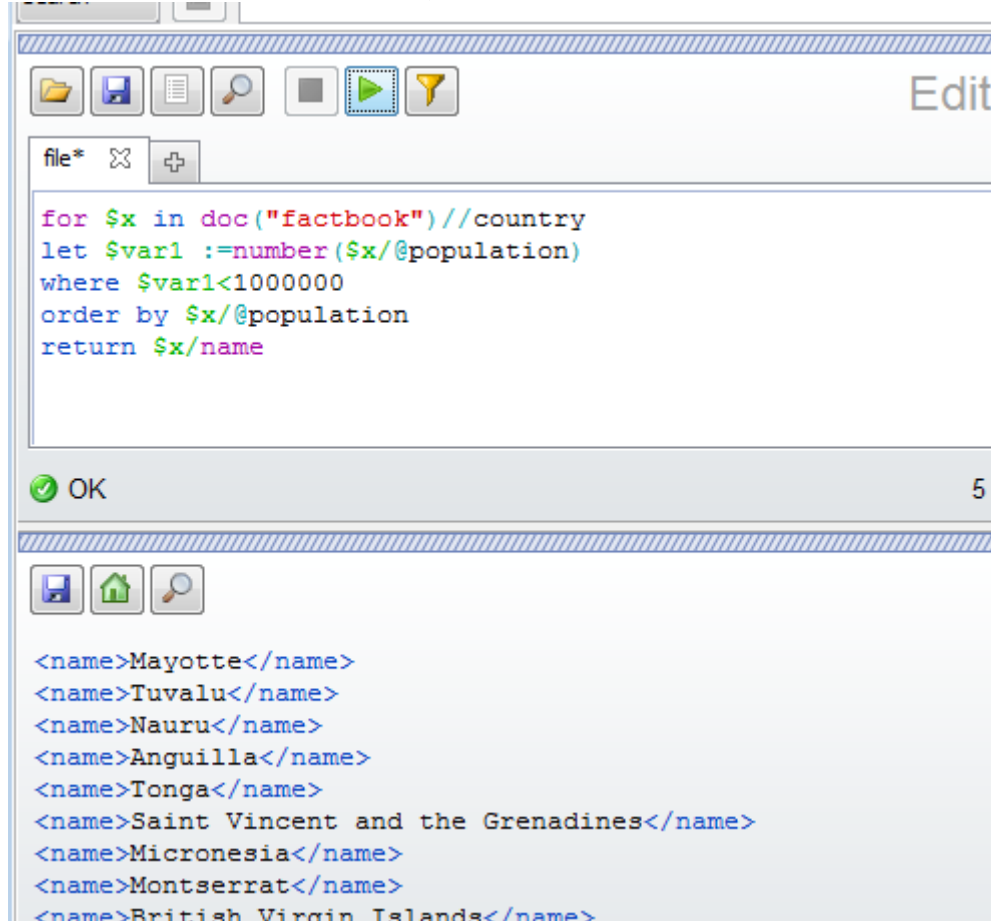
- ✓ To use xPath, type in the following xPath Expression within the Editor window: **//name** Then click the **Green** button. The results would be shown below:



- ✓ To use or practice xQuery, Type in the following statements instead inside the editor:

```
1. for $x in doc("factbook")//country
2. let $var1 :=number($x/@population)
3. where $var1<1000000
4. order by $x/@population
5. return $x/name
```

- ✓ Click the Green Button to execute, the results would be shown below:



## Работа с XPath

Это наши тестовые данные XML-документа.

1. <magazine>
2. <book>
3. <title>Learning PHP, MySQL and JavaScript With jQuery, CSS and HTML5</title>
4. <author>Робин Никсон</author>
5. <price>1158 руб.</price>
6. <content>
7. <rating>4.3</rating>
8. <description>The fully revised, updated and extended 4th edition of the hugely popular web development book - includes CSS, HTML5, jQuery and the mysqli extension.</description>
9. </content>
10. </book>
11. <journal>
12. <title>PHP World</title>
13. <author>Anton Bekon</author>
14. <price>412 руб.</price>
15. </journal>
16. <book>
17. <title>PHP. Быстрый старт</title>
18. <author>Каллум Хопкинс</author>
19. <price>348 руб.</price>

```
20. <content>
21. <rating>3.5</rating>
22. <description>Все, что нужно знать о PHP в одной книге!</description>
23. </content>
24. </book>
25. <book>
26. <title>Modern PHP: New Features and Good Practices</title>
27. <author>Джош Локхарт</author>
28. <price>834 руб.</price>
29. <content>
30. <rating>5</rating>
31. <description>PHP is experiencing a renaissance, though it may be difficult to tell with all of the outdated
    PHP tutorials online.</description>
32. </content>
33. </book>
34. </magazine>
```

## Запрос: //book

Результат:

```
1. <book>
2. <title>Learning PHP, MySQL and JavaScript With jQuery, CSS and HTML5</title>
3. <author>Робин Никсон</author>
4. <price>1158 руб.</price>
5. <content>
6. <rating>4.3</rating>
7. <description>The fully revised, updated and extended 4th edition of the hugely popular web development
    book : includes CSS, HTML5, jQuery and the mysqli extension.</description>
8. </content>
9. </book>
10. <book>
11. <title>PHP. Быстрый старт</title>
12. <author>Каллум Хопкинс</author>
13. <price>348 руб.</price>
14. <content>
15. <rating>3.5</rating>
16. <description>Все, что нужно знать о PHP в одной книге!</description>
17. </content>
18. </book>
19. <book>
20. <title>Modern PHP: New Features and Good Practices</title>
21. <author>Джош Локхарт</author>
22. <price>834 руб.</price>
23. <content>
24. <rating>5</rating>
25. <description>PHP is experiencing a renaissance, though it may be difficult to tell with all of the outdated
    PHP tutorials online.</description>
26. </content>
27. </book>
```

Здесь получены все узлы с именем <book>.

## Запрос: //book[2]



```
1. <book>
2. <title>PHP. Быстрый старт</title>
3. <author>Каллум Хопкинс</author>
4. <price>348 руб.</price>
5. <content>
6. <rating>3.5</rating>
7. <description>Все, что нужно знать о PHP в одной книге!</description>
8. </content>
9. </book>
```

Здесь получен только второй узел `<book>`.

Если выполнить запрос как `//book[last()]`, то вы получите только последний узел. В данном случае такой запрос аналогичен `//book[3]`.

## Запрос: `//title/..`

Результат:

```
1. <book>
2. <title>Learning PHP, MySQL and JavaScript With jQuery, CSS and HTML5</title>
3. <author>Робин Никсон</author>
4. <price>1158 руб.</price>
5. <content>
6. <rating>4.3</rating>
7. <description>The fully revised, updated and extended 4th edition of the hugely popular web development
   book : includes CSS, HTML5, jQuery and the mysqli extension.</description>
8. </content>
9. </book>
10. <journal>
11. <title>PHP World</title>
12. <author>Anton Bekon</author>
13. <price>412 руб.</price>
14. </journal>
15. <book>
16. <title>PHP. Быстрый старт</title>
17. <author>Каллум Хопкинс</author>
18. <price>348 руб.</price>
19. <content>
20. <rating>3.5</rating>
21. <description>Все, что нужно знать о PHP в одной книге!</description>
22. </content>
23. </book>
24. <book>
25. <title>Modern PHP: New Features and Good Practices</title>
26. <author>Джош Локхарт</author>
27. <price>834 руб.</price>
28. <content>
29. <rating>5</rating>
30. <description>PHP is experiencing a renaissance, though it may be difficult to tell with all of the outdated
   PHP tutorials online.</description>
31. </content>
32. </book>
```

При данном запросе мы получаем родителя элемента `<title>`, т.е. все узлы с именем `<book>` и `<journal>`.

Запрос: `//book/author[text() = «Каллум Хопкинс»]`

Результат:

```
<author>Каллум Хопкинс</author>
```

Возвращается элемент `<author>` с содержимым «Каллум Хопкинс».

Запрос: `//book/content/rating[text()<5]`

Результат:

1. `<rating>4.3</rating>`
2. `<rating>3.5</rating>`

Как видите, были возвращены все элементы `<rating>` со значением меньше 5.

Запрос: `//book[position()<3]`

Результат:

1. `<book>`
2. `<title>Learning PHP, MySQL and JavaScript With jQuery, CSS and HTML5</title>`
3. `<author>Робин Никсон</author>`
4. `<price>1158 руб.</price>`
5. `<content>`
6. `<rating>4.3</rating>`
7. `<description>The fully revised, updated and extended 4th edition of the hugely popular web development book : includes CSS, HTML5, jQuery and the mysqli extension.</description>`
8. `</content>`
9. `</book>`
10. `<book>`
11. `<title>PHP. Быстрый старт</title>`
12. `<author>Каллум Хопкинс</author>`
13. `<price>348 руб.</price>`
14. `<content>`
15. `<rating>3.5</rating>`
16. `<description>Все, что нужно знать о PHP в одной книге!</description>`
17. `</content>`
18. `</book>`

Здесь происходит выбор первых двух узлов `<book>`.

Запрос: `magazine/*/price`

Результат:

1. `<price>1158 руб.</price>`
2. `<price>412 руб.</price>`
3. `<price>348 руб.</price>`
4. `<price>834 руб.</price>`

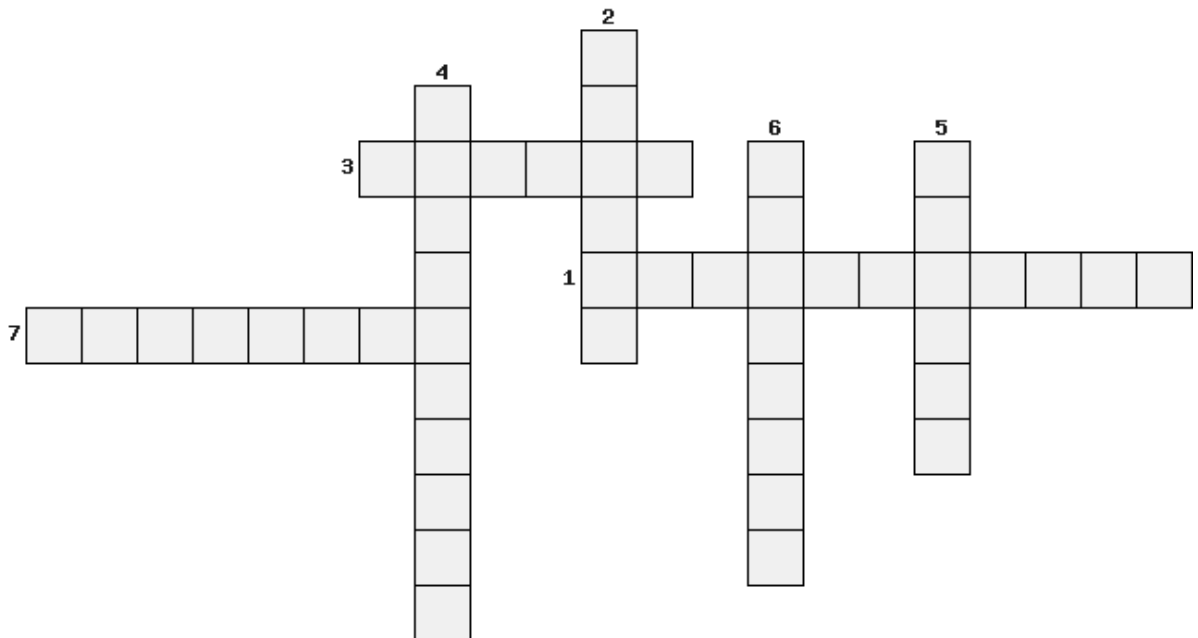
Таким образом мы вернули все элементы `<price>`, который были найдены в корневом узле `<magazine>`.

Примеров можно написать огромное множество. Я лишь показал часть возможностей по работе с xPath. В самом конце статьи прикрепляю полезные ссылки на ресурсы от Microsoft.

### Задание 1.

1. Назовите основные требования к системе контроля версий Git.
2. Для чего нужна система контроля версий?
3. Как называется компактная система управления базами данных XML?
4. Что означает «язык разметки»?
5. Что подразумевается под «репозиторием»?

### Задание 2



По горизонтали:

- 1 Что программисты могут объединить на Git?
- 3 В виде чего GitHub может изображать вклад участников?
- 7 Название облегченного языка разметки

По вертикали:

- 2 В каком месяце был создан первый частный репозиторий?
- 4 В список основных требований к Git входила поддержка какой разработки?
- 5 крупнейший веб-сервис для хостинга IT-проектов
- 6 Также входит в список требований к Git

### Задание 3

Git

1. Как перейти из ветки master в ветку dev?

- a git checkout dev
- b git move dev
- c git change master dev
- d git branch master dev

2. Как посмотреть список меток?

- a)git label
- b)git labels
- c)git show --labels
- d)git tag

3. При помощи какой команды можно посмотреть историю всех коммитов с сокращённым SHA-1 хэшем?

- a)git log --short-hash
- b)git log --abbrev-commit
- c)git log --short
- d)git log --tiny-commit

4. Какова минимальная длина SHA-1 хэша должна быть, чтобы можно было посмотреть информацию о коммите?

- a 3 символа
- b 5 символов
- c 4 символа
- d 6 символов

5. Какую команду необходимо выполнить, чтобы запустить графический инструмент разрешения конфликтов при merge?

- a git mergegui
- b git mergetool
- c git mergemaster
- d git mergemodify
- e git mergecheck

6. Как посмотреть последний коммит у каждой ветки?

- a git branch -v
- b git branch --last-commit
- c git checkout --last-commit
- d git commit --branch --last-commit

### Задание 4

Git

1. Какой порт используется в git протоколе?
2. Как получить список всех веток?
3. Как «спрятать» данные в git?

### Задание 5

Дополните определение:

1. Система контроля версий-это ...
2. Язык разметки-это ...
3. Репозиторий-это...

Используемые ресурсы:

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

<https://github.com/>

[Примеры XPath](#)

[Функции XPath](#)

<http://attacklab.net/showdown/>

<http://daringfireball.net/projects/markdown/syntax>