1.1

1. Настройка параметров Git

Егор@DESKTOP-FAKBSA0 MINGW64 ~ (master)

$ git config --global user.name "Egore\_Streltsov"

Егор@DESKTOP-FAKBSA0 MINGW64 ~ (master)

$ git config --global user.email "hivir2222@gmail.com"

Егор@DESKTOP-FAKBSA0 MINGW64 ~ (master)

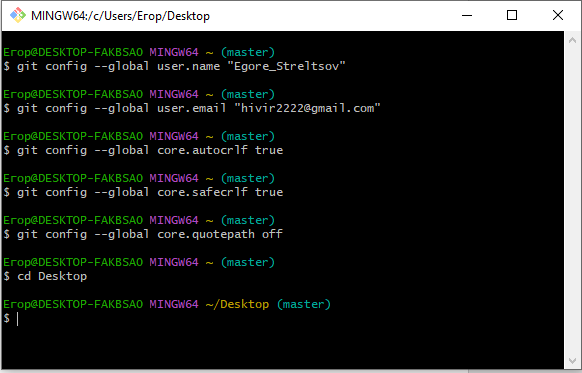
$ git config --global core.autocrlf true

Егор@DESKTOP-FAKBSA0 MINGW64 ~ (master)

$ git config --global core.safecrlf true

Егор@DESKTOP-FAKBSA0 MINGW64 ~ (master)

$ git config --global core.quotepath off



2. Создание проекта и файла hello.html

Егор@DESKTOP-FAKBSA0 MINGW64 ~ (master)

$ cd Desktop

Егор@DESKTOP-FAKBSA0 MINGW64 ~/Desktop (master)

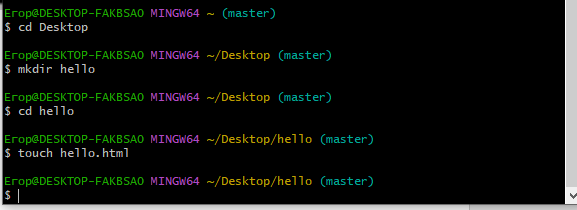
$ mkdir hello

Егор@DESKTOP-FAKBSA0 MINGW64 ~/Desktop (master)

$ cd hello

Егор@DESKTOP-FAKBSA0 MINGW64 ~/Desktop/hello (master)

$ touch hello.html



3. Инициализация Git, добавление страницы в репозиторий, добавление коммита

Егор@DESKTOP-FAKBSA0 MINGW64 ~/Desktop/hello (master)

$ git init

Initialized empty Git repository in C:/Users/Егор/Desktop/hello/.git/

Егор@DESKTOP-FAKBSA0 MINGW64 ~/Desktop/hello (master)

$ git add hello.html

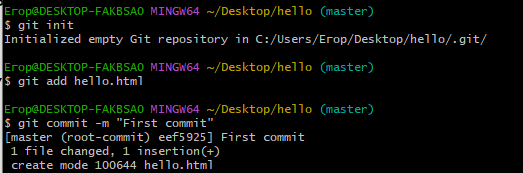
Егор@DESKTOP-FAKBSA0 MINGW64 ~/Desktop/hello (master)

$ git commit -m "First commit"

[master (root-commit) eef5925] First commit

1 file changed, 1 insertion(+)

create mode 100644 hello.html



4. Проверка состояний до изменений и после внесения изменений

Егор@DESKTOP-FAKBSA0 MINGW64 ~/Desktop/hello (master)

$ git status

On branch master

nothing to commit, working tree clean

Егор@DESKTOP-FAKBSA0 MINGW64 ~/Desktop/hello (master)

$ git status

On branch master

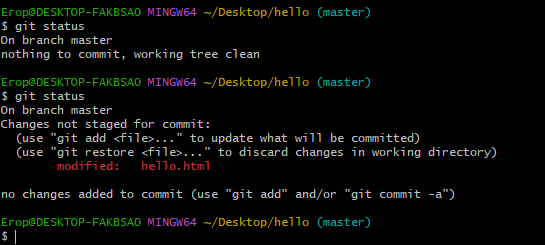
Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working directory)

modified: hello.html

no changes added to commit (use "git add" and/or "git commit -a")



5. Индексация изменений

Егор@DESKTOP-FAKBSA0 MINGW64 ~/Desktop/hello (master)

$ git add hello.html

Егор@DESKTOP-FAKBSA0 MINGW64 ~/Desktop/hello (master)

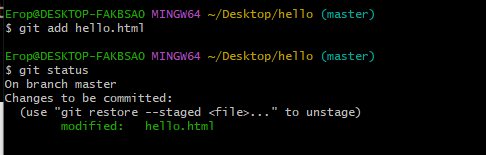
$ git status

On branch master

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

modified: hello.html



6. Индексация и коммит

Егор@DESKTOP-FAKBSA0 MINGW64 ~/Desktop/hello (master)

$ git add a.html

fatal: pathspec 'a.html' did not match any files

Егор@DESKTOP-FAKBSA0 MINGW64 ~/Desktop/hello (master)

$ git add a.html

Егор@DESKTOP-FAKBSA0 MINGW64 ~/Desktop/hello (master)

$ git b.html

git: 'b.html' is not a git command. See 'git --help'.

Егор@DESKTOP-FAKBSA0 MINGW64 ~/Desktop/hello (master)

$ git add b.html

Егор@DESKTOP-FAKBSA0 MINGW64 ~/Desktop/hello (master)

$ git commit -m "Changes for a and B"

[master d6feb6b] Changes for a and B

3 files changed, 1 insertion(+), 1 deletion(-)

create mode 100644 a.html

create mode 100644 b.html

Егор@DESKTOP-FAKBSA0 MINGW64 ~/Desktop/hello (master)

$ git add v.html

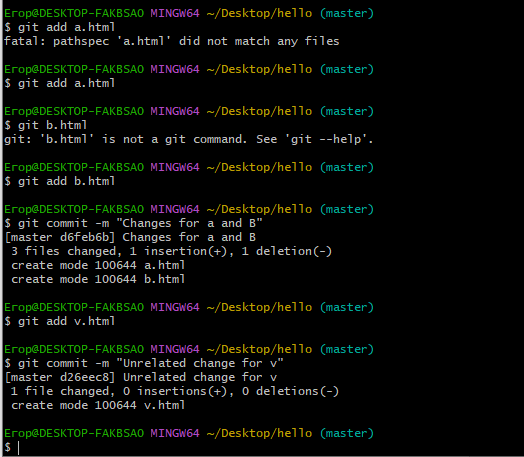
Егор@DESKTOP-FAKBSA0 MINGW64 ~/Desktop/hello (master)

$ git commit -m "Unrelated change for v"

[master d26eec8] Unrelated change for v

1 file changed, 0 insertions(+), 0 deletions(-)

create mode 100644 v.html



7. Коммит изменений

Егор@DESKTOP-FAKBSA0 MINGW64 ~/Desktop/hello (master)

$ git add hello.html

Егор@DESKTOP-FAKBSA0 MINGW64 ~/Desktop/hello (master)

$ git commit

[master 3a0e481] added h2 tag

1 file changed, 1 insertion(+)

Егор@DESKTOP-FAKBSA0 MINGW64 ~/Desktop/hello (master)

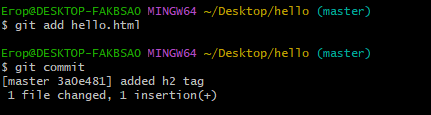
$ git status

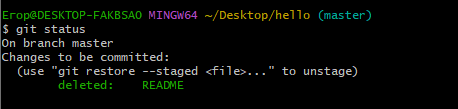
On branch master

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

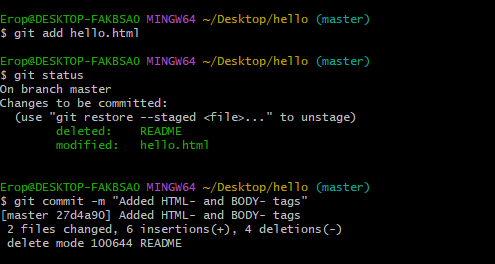
deleted: README



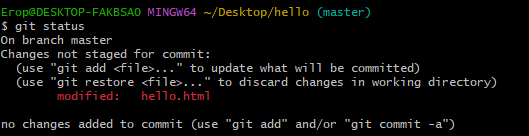


8. Изменения

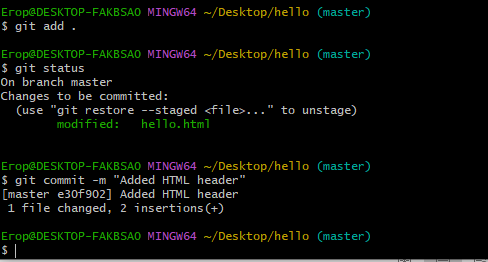
Сначала нужно в файл hello.html добавить теги html и body, а затем проиндексировать изменения.



Затем еще раз нужно добавить изменения в файл (добавить тег head), но изменения не индексировать .

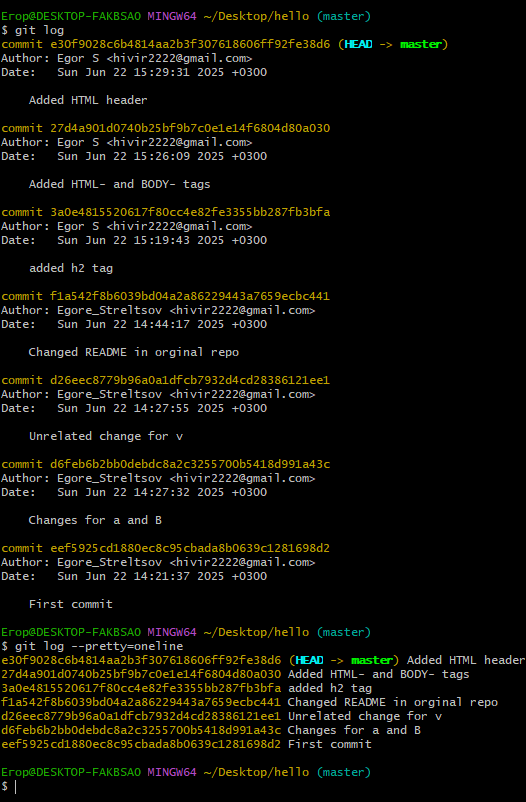


Добавление второго изменения в индекс, коммит второго изменения

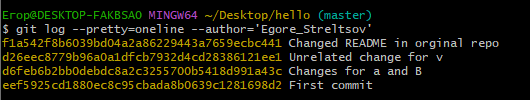


1.2 Git

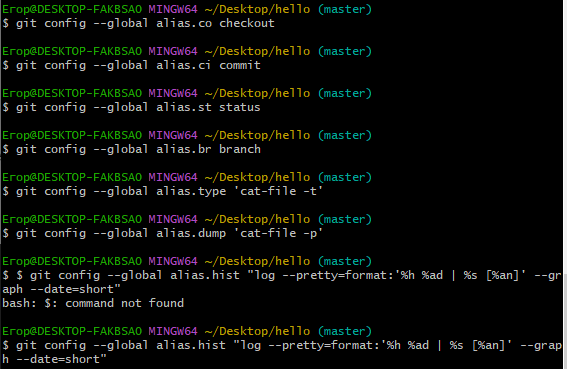
Вывод списка произведенных изменений с полной информацией и в однострочном формате:



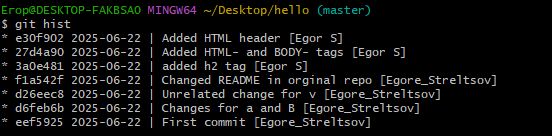
Вывод изменений с автором :



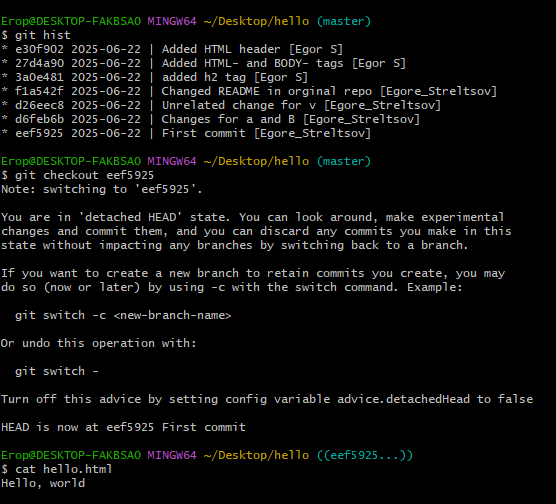
Настройка алиасов



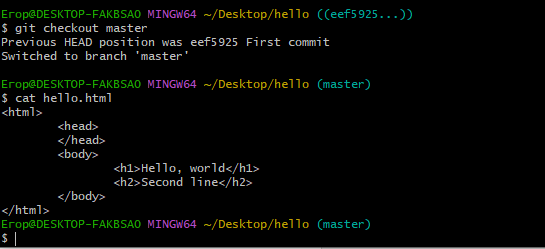
Примеры выводов с использованием алиасов:



Далее нужно выполнить команду git checkout с номером нужного хэша



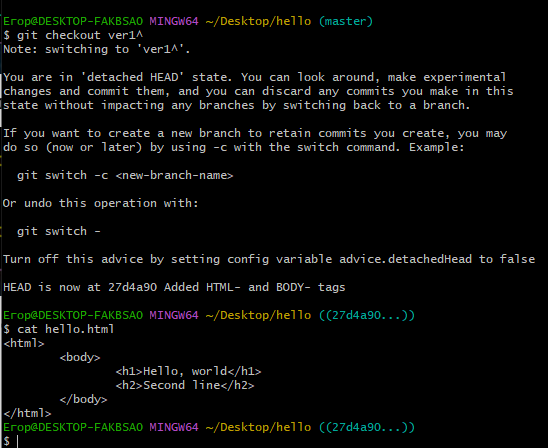
Возвращение к последней версии в ветке master и вывод содержимого



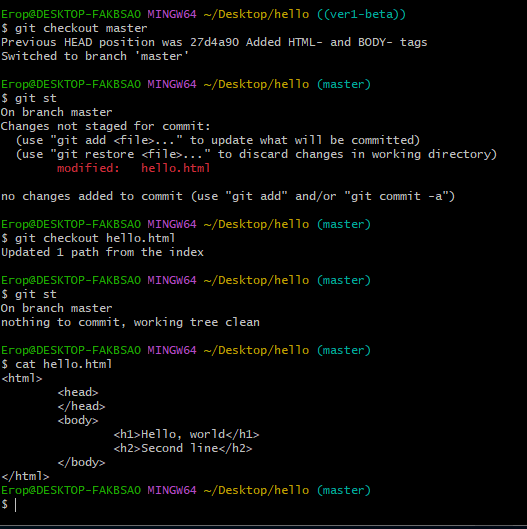
Создание тегов версий

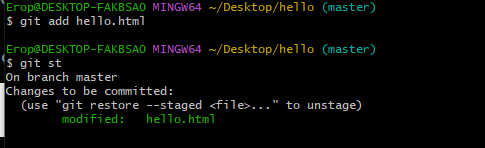


Чтобы перейти к предыдущей версии, можно использовать символ «^», который означает «родитель».

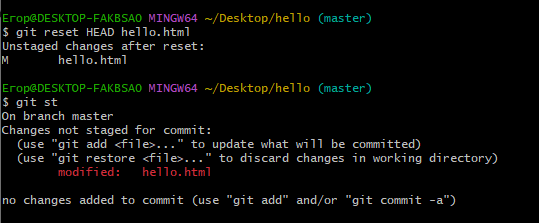


Отмена локальных изменений до индексации

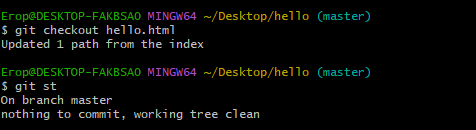


Отмена проиндексированных изменений 

Для отмены индексация изменения используется команда git reset HEAD hello.html



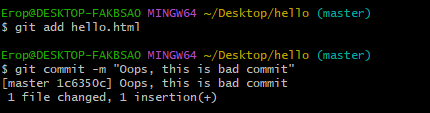
Команда reset сбрасывает буферную зону к HEAD и очищает ее от проиндексированных изменений. Но для удаления ненужного по-прежнему используется команда git checkout



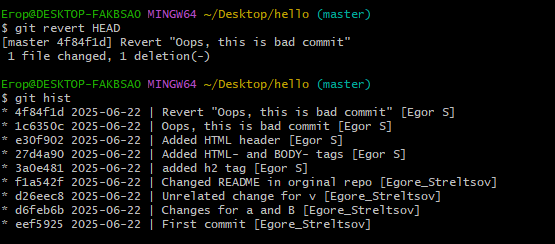
Отмена коммитов

Для отмены коммита можно использовать способ создания нового коммита, отменяющего изменения.

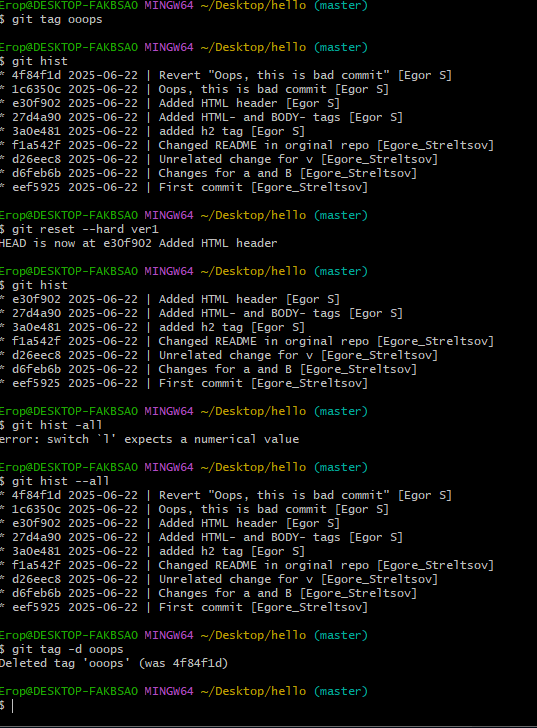
Для начала надо внести изменение, проиндексировать его и записать коммит



Для создания коммита, который удалит ненужные изменения, используется команда git revert HEAD После этого будет открыт редактор, в котором можно отредактировать коммит сообщение затем надо сохранить файл и закрыть редактор.



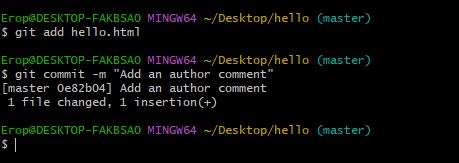
Создание, а затем удаление тега “ooops”



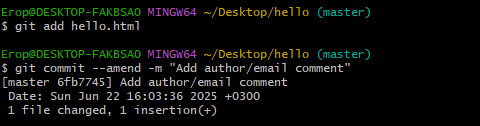
Git 1.3

Внесение изменений в коммиты

Для начала будет создан коммит, в который позже будут внесены изменения. Добавление комментария в файл hello.html и его индексация и коммит.

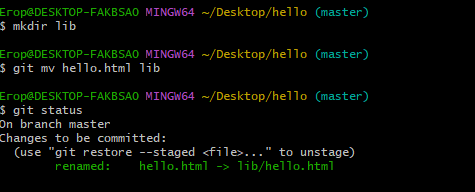


Добавление почты в коммент, и изменения коммита с указанием добавления почты.

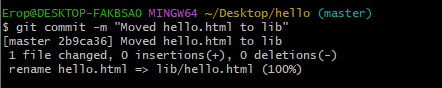


Перемещение файлов

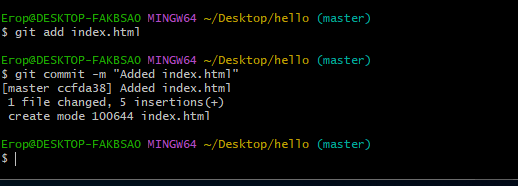
Для перемещения файлов в пределах репозитория используются команды. После выполнения данных команды git индексирует эти изменения (удаление файла hello.html и создание файла lib/hello.html).

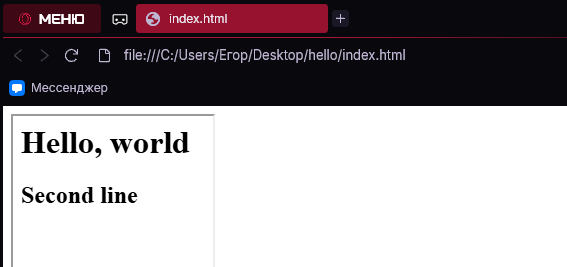


Коммит данного действия



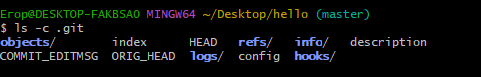
Создание нового файла – index.html, с внесением в него изменений, индексация и коммит



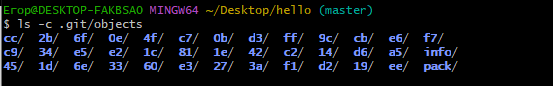


Каталог .git

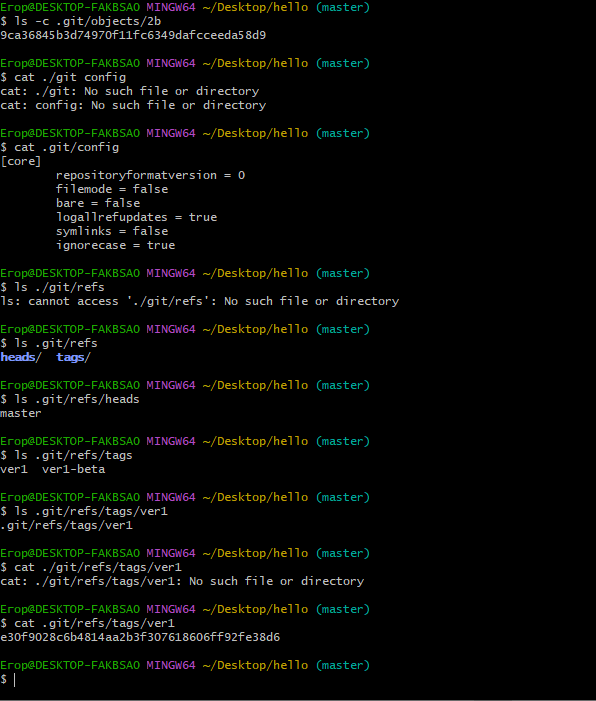
Для просмотра каталога .git необходимо выполнить след. Команду



Просмотр каталога objects



Просмотр различных файлов в подкаталоге tags, веток в каталоге heads.

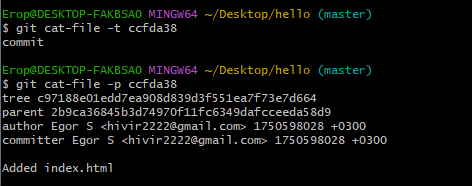


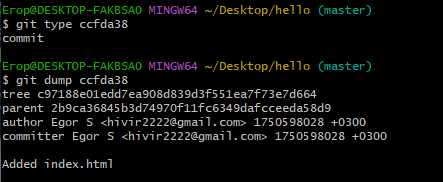
Работа с обьектами git

Просмотр последнего коммита

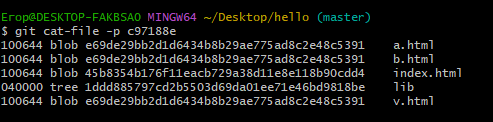


Далее надо использовать хэш последнего коммита, используя команды cat-file -p и cat-file -t для просмотра объекта коммита. Также вместо длинных команд можно использовать сокращенные type и dump, если данные команды были заданы как алиасы





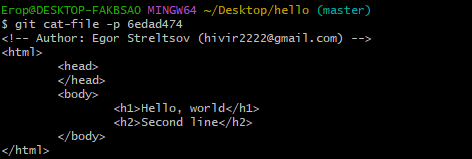
Просмотр дерева каталогов с помощью его хэша



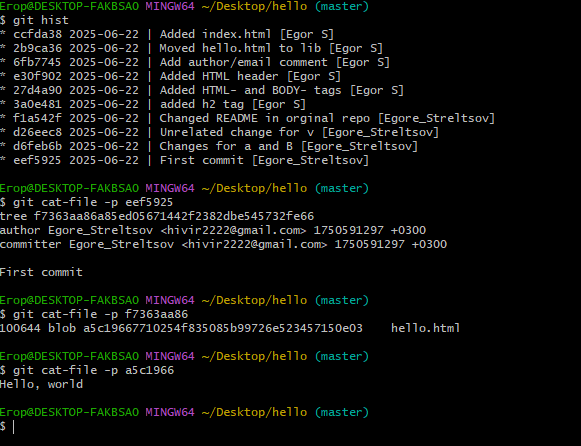
Просмотр каталога lib



Просмотр содержимого файла hello.html

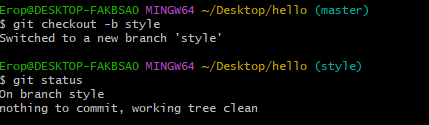


Аналогичным образом можно просмотреть содержимое файла, каким оно было в самом первом коммите. Для этого требуется использовать лишь нужные хэши.

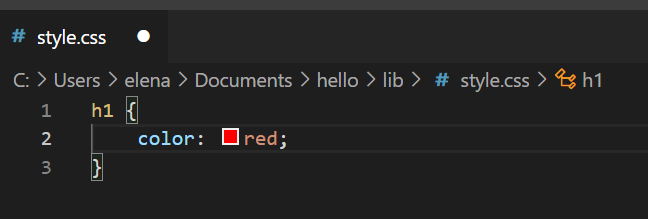
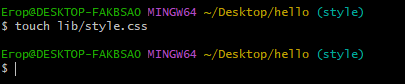


Git 1.4

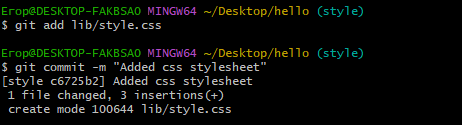
Создание ветки style



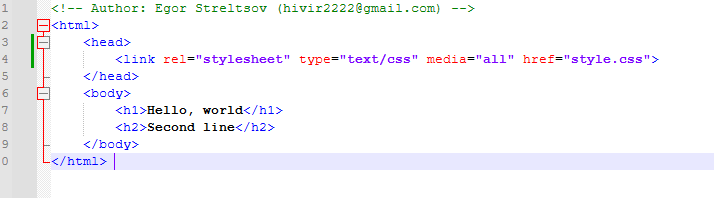
Затем нужно создать файл стилей и внести в него код

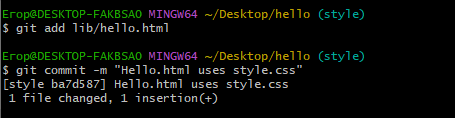


После этого надо произвести индексацию и коммит

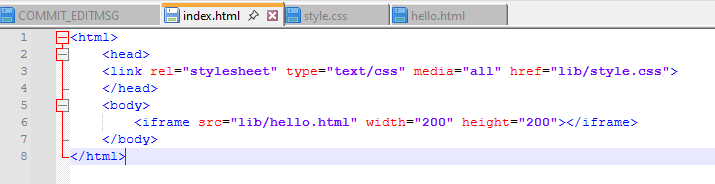


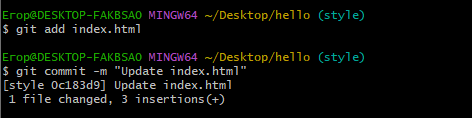
Далее требуется изменить основную страницу hello.html и закоммитить изменения





Далее аналогичные действия нужно осуществить с файлом index.html



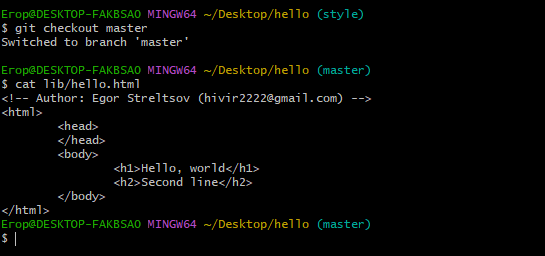


Навигация по веткам

При просмотре истории можно увидеть, что теперь в проекте 2 ветки.



Для переключения на ветку master используется команда git checkout master. После переключения на нужную ветку при выводе файла hello.html можно увидеть, что изменения отсутствуют (по причине того, что они закоммичены в другой ветке).



При переключении на ветку style файл hello.html будет иметь другое содержание



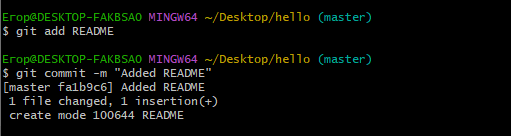
Изменение в ветке master

Необходимо переключиться на ветку master и добавить файл README

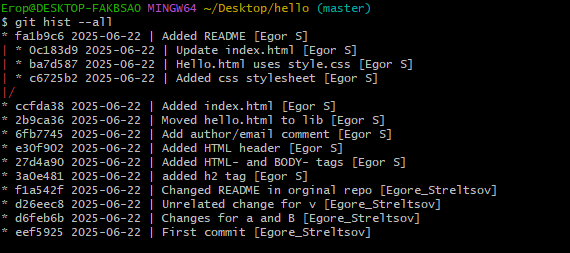




Индексация и коммит

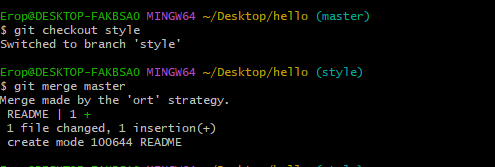


Просмотр отличающихся веток

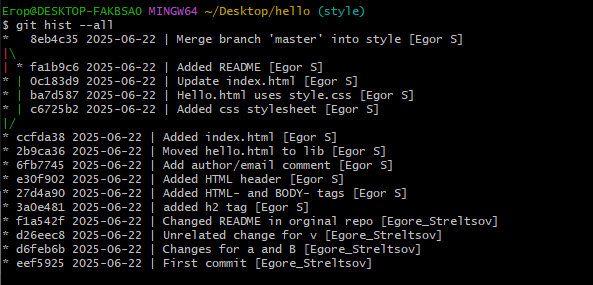


Слияние

Слияние переносит изменения из двух веток в одну. Для слияния нужно перейти на ветку style и с помощью команды git merge master совместить ветки

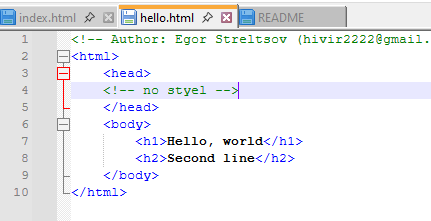


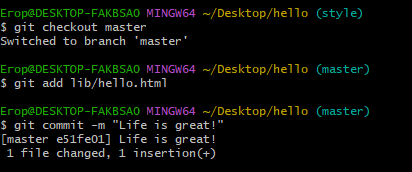
Просмотр истории



Создание конфликта

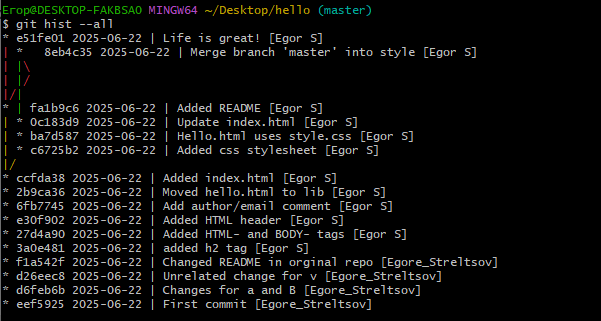
Для того, чтобы создать конфликт необходимо перейти в ветку master и внести изменения в файл hello.html





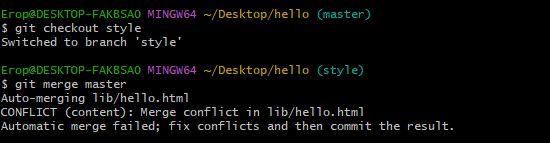
Индексация и коммит

После выполнения предыдущих действий при просмотре веток можно будет увидеть конфликт в виде:



Разрешение конфликтов

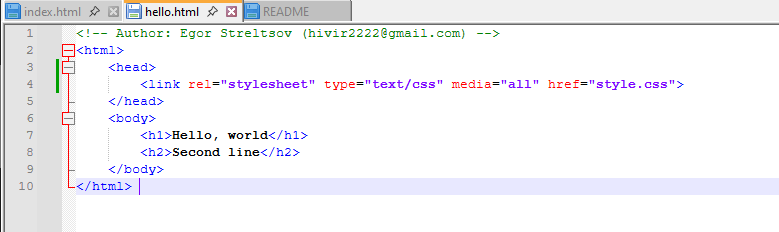
При попытке объединить ветку style с master будет показана ошибка из-за конфликта

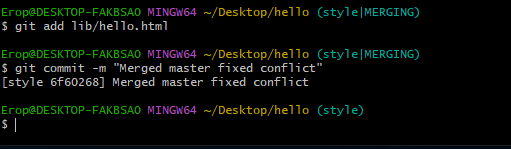


При открытии файла hello.html конфликт будет показан



Решение конфликта вручную

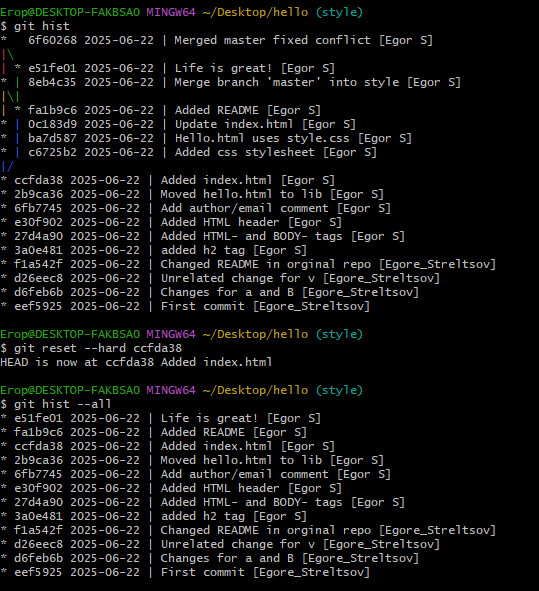




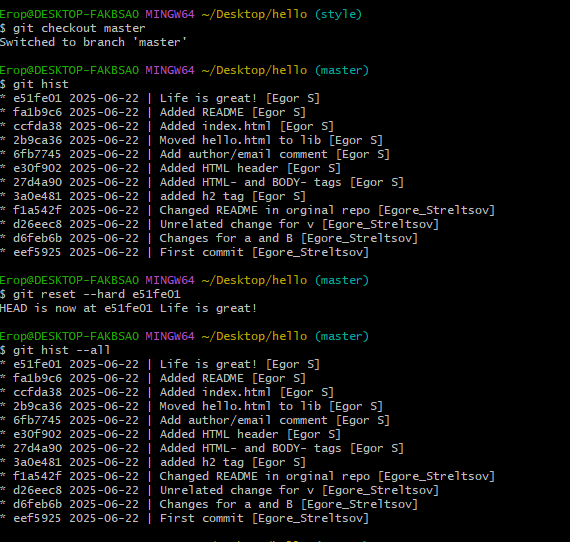
Git 1.5

Сброс ветки style

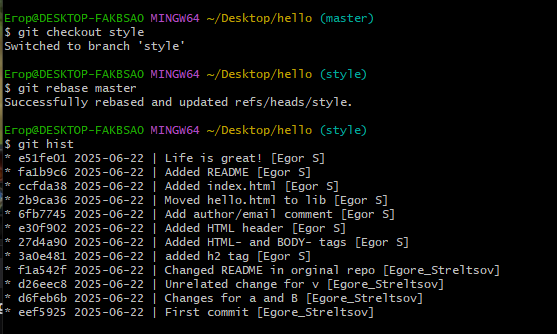
Для сброса ветки необходимо применить команду reset --hard до требуемой точки

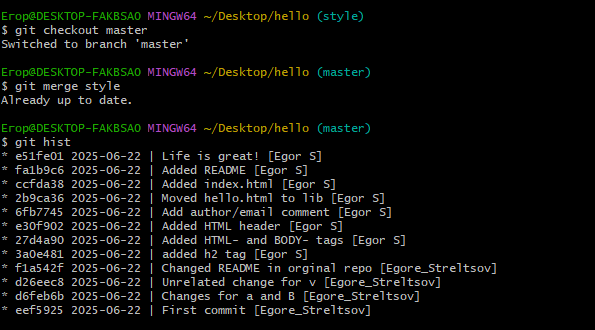


Сброс ветки master



Перебазирование

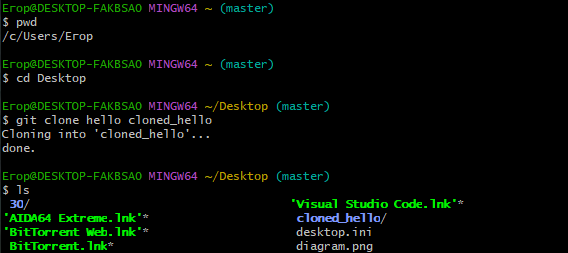


Слияние в ветку master 

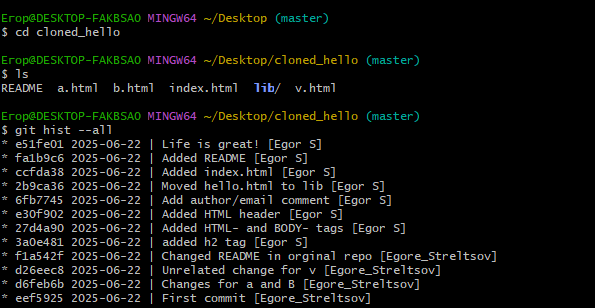
Клонирование репозиториев

Далее требуется научиться делать копии репозиториев.

Для этого необходимо перейти в рабочий каталог и затем использовать команду git clone.

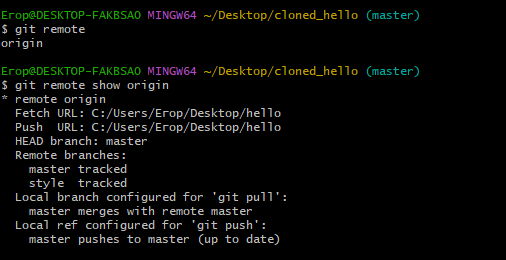


Просмотр клонированного репозитория



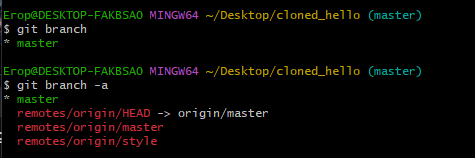
Origin

Origin – имя по умолчанию. Просмотр данных о нем возможен с помощью команд



Удаление ветки

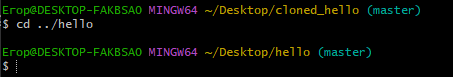
Для просмотра удаленных веток используется команда git branch –a



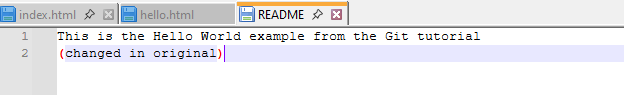
Git 1.6

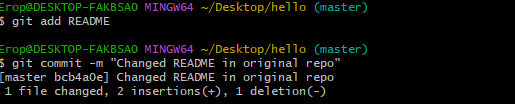
Работа с Git

Сначала необходимо внести изменения в оригинальный репозиторий. Для этого нужно перейти в данный репозиторий

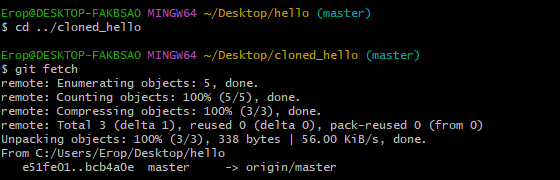


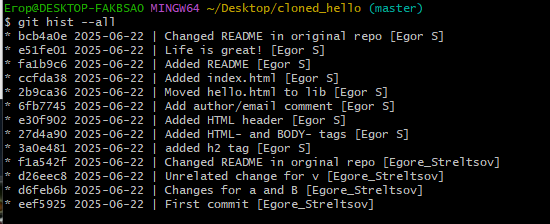
Далее надо внести изменения в файл README (рисунок 2) и затем произвести индексацию и коммит





Далее требуется перейти в клонированный репозиторий и извлечь изменения с помощью команды git fetch и просмотреть историю .

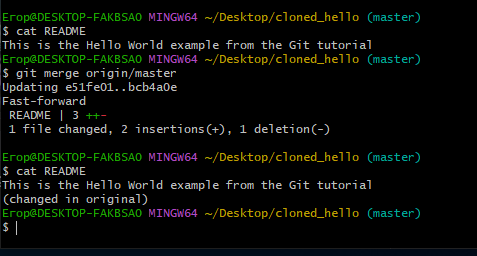


Извлечение изменений, просмотр истории 

При попытке вывести содержимое файла README можно увидеть, что изменения не были внесены

Далее нужно слить извлеченные изменения в ветку master

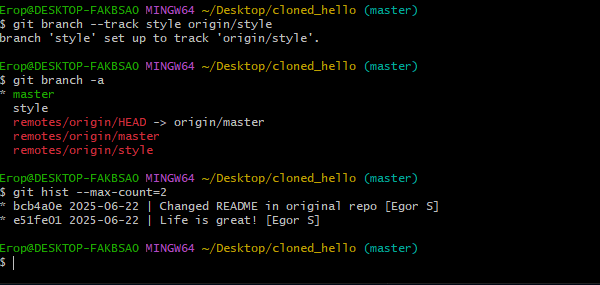
И после выполнения предыдущего действия при выводе README можно будет увидеть последние изменения



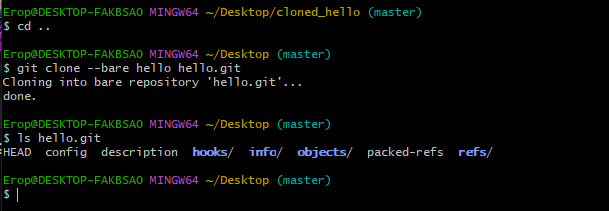
Также существует команда, объединяющая функции git fetch и git merge



Далее требуется добавить локальную ветку, которая будет отслеживать удаленную ветку



Далее необходимо создать чистый репозиторий

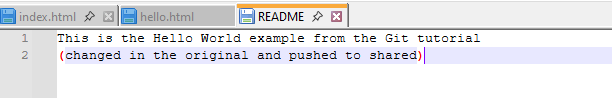


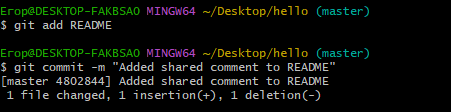
Для добавления удаленного репозитория используется команда git remote add shared



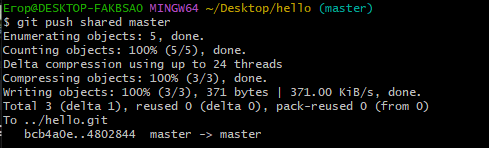
Затем требуется научиться отправлять изменения в удаленный репозиторий.

Для этого сначала надо внести изменения, проиндексировать и произвести коммит.



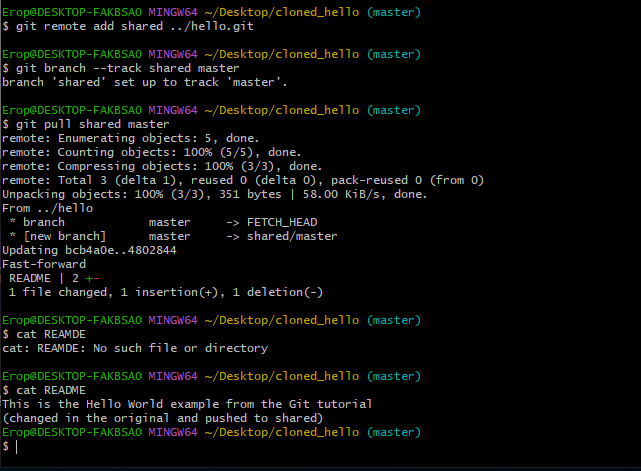


Далее надо отправить изменения в общий репозиторий, используя команду git push shared master



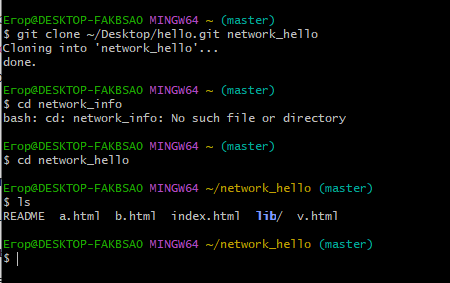
Для извлечения общих изменений нужно перейти в клонированный каталог и выполнить перечень команд





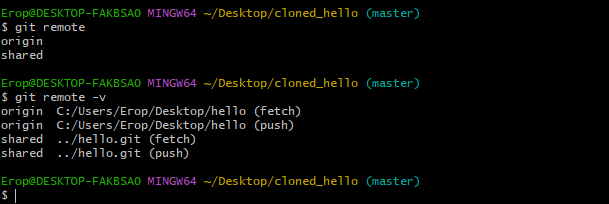
Для настройки git сервера нужно выполнить команду daemon –verbose –export-all –base-path= Затем в другом окне можно проверить работу сервера, сделав копию проекта hello



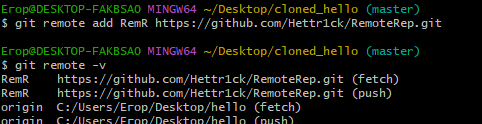


Git 1.7

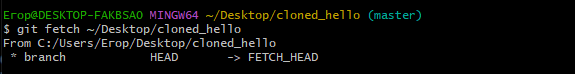
Для того, чтобы просмотреть список настроенных удалённых репозиториев, необходимо запустить команду git remote



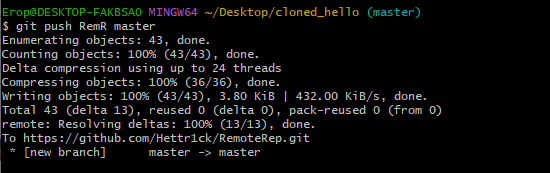
Для добавления удаленного репозитория с новым именем используется команда git remote add



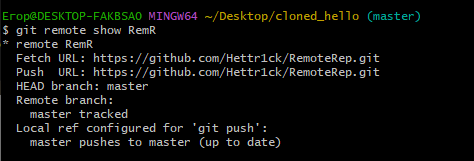
Для получения данных из удалённых проектов используется команда git fetch



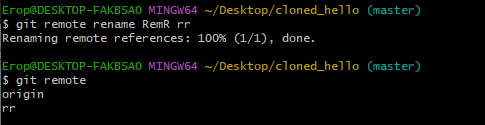
Для отправки изменений в удаленный репозиторий используется команда git push



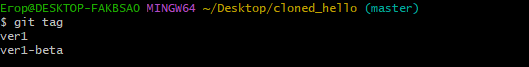
Для получения информации об одном из удалённых репозиториев, можно использовать команду git remote show



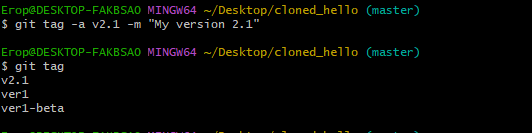
Для переименования удаленных репозиториев используется команда git remote rename



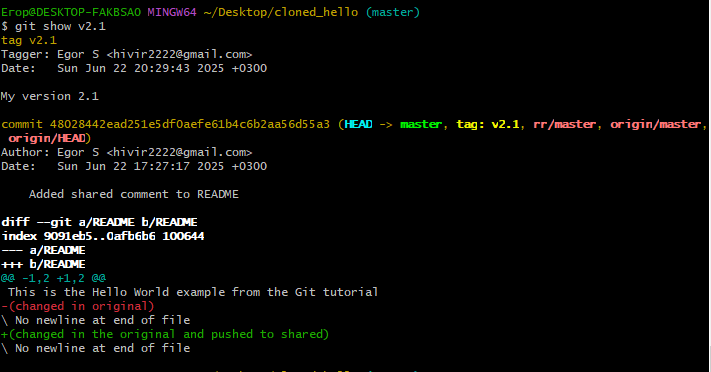
Просмотреть существующие теги можно с помощью команды git tag



Для создания аннотированной метки нужно выполнить команду

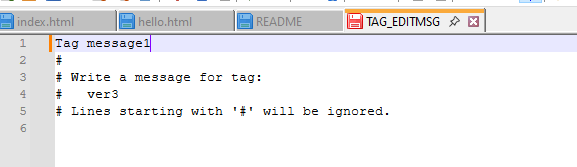


Команда git show осуществляет просмотр данных тегов вместе с коммитом

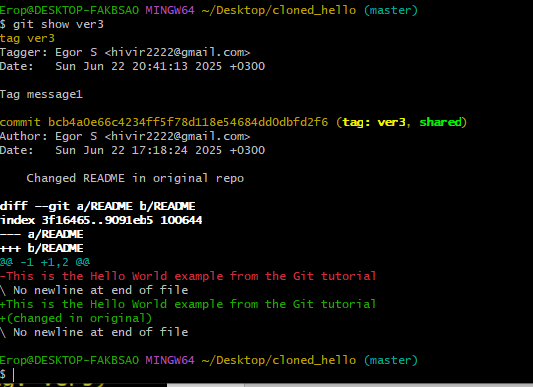


Для отметки определенного коммита тегом надо указать его хэш

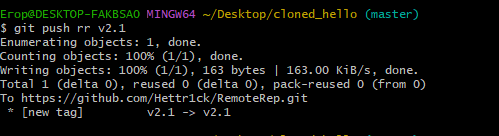
Создание тега определенному коммиту



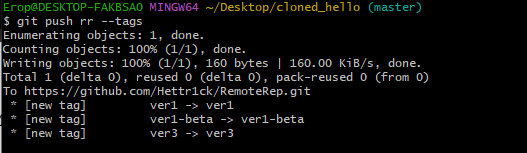
Данные этого тега можно просмотреть аналогичным образом



По умолчанию, команда git push не отправляет теги на удалённые сервера. Нужно выполнить команду git push rr



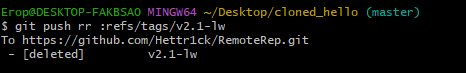
Можно использовать опцию --tags для команды git push. В таком случае все теги отправятся на удалённый сервер



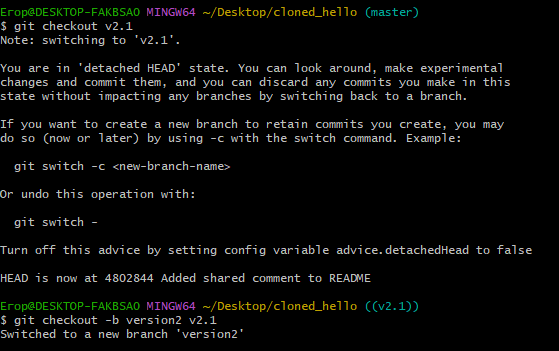
Для того, чтобы удалить тег, надо использовать команду git tag с параметром –d



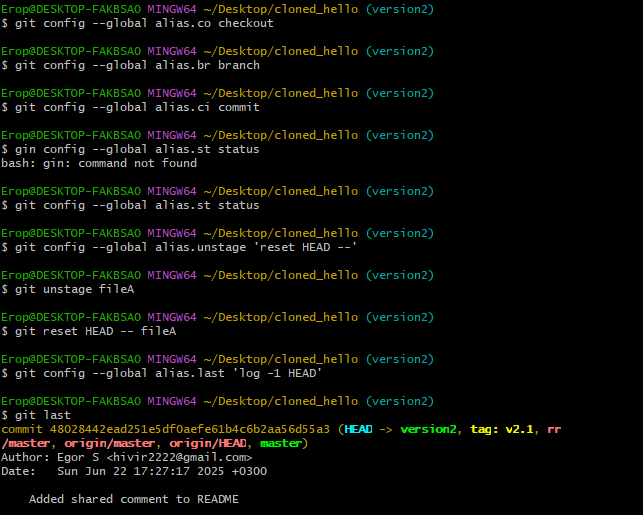
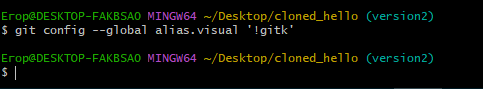
Для удаления тега с сервера используется команда :refs/tags



Для того, чтобы получить версии файлов, на которые указывает тег, можно выполнить git checkout для тега. Однако, это переведёт репозиторий в состояние «detached HEAD». Если в состоянии «detached HEAD» внести изменения и сделать коммит, то тег не изменится, при этом новый коммит не будет относиться ни к какой из веток, а доступ к нему можно будет получить только по его хэшу. Поэтому в таком случае следует создать новую ветку

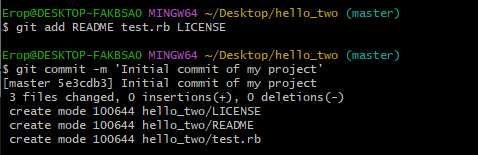


Создание алиасов:

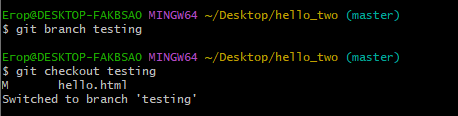
 

Git 1.8

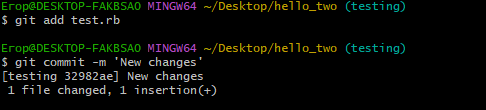
Для начала следует создать репозиторий, создать 3 файла и добавить их в коммит



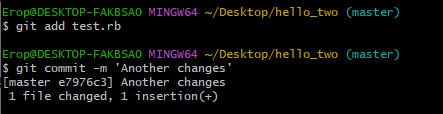
Затем надо создать ветку testing и переключиться на нее



Далее надо внести изменения в файл test.rb и создать коммит



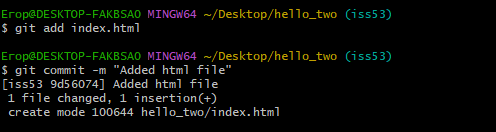
Затем необходимо переключиться на ветку master и внести изменения в файл test.rb на этой ветке



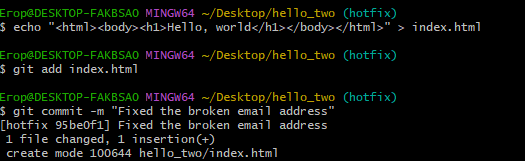
Команда git checkout -b позволяет сразу создать и переключиться на ветку

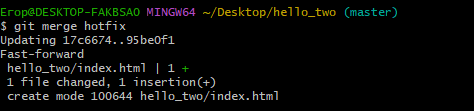


В новой ветке нужно внести в файл изменения

Затем надо на ветке hotfix добавить изменения в файл index.html, а затем слить эту ветку и master

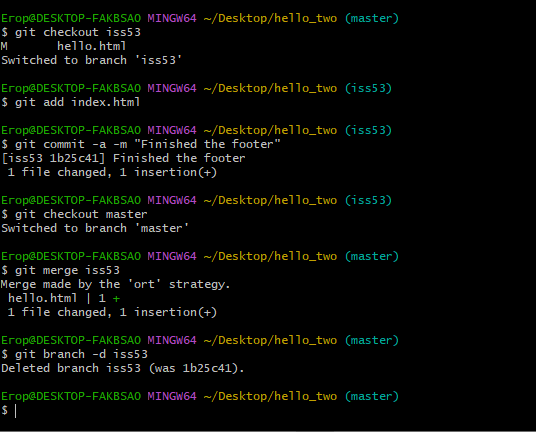




Удаление ветки hotfix

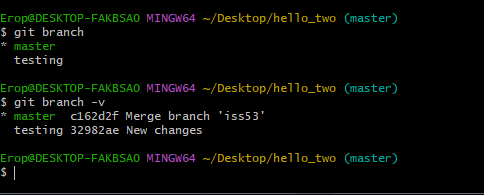


Слияние веток master и iss53 с удалением последней

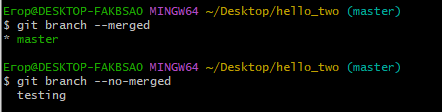


Git 1.9

Просмотр всех веток и последние коммиты в каждой:



Опции --merged и --no-merged могут отфильтровать этот список для вывода только тех веток, которые слиты или ещё не слиты в текущую ветку



Затем следует удалить ветку testing

