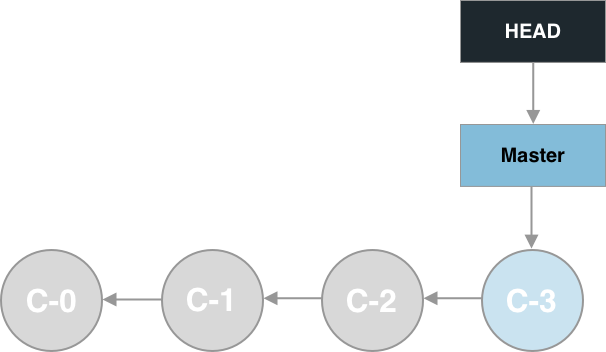
**Ветвление и коммиты слияния**

Ниже мы рассмотрим пример ветвления на гипотетическом проекте. Пока делать в терминале ничего не нужно — на этом этапе вам важно усвоить механику в теории. Практиковаться будем чуть позже.

Допустим, мы работаем над веб-сайтом. Предположим, что скоро Чёрная пятница и нам надо создать для работы над изменениями специальную ветку. Непосредственно перед днём распродаж надо будет сделать так, чтобы наши изменения попали к пользователям.

Представим, что работа над проектом ведётся уже какое-то время и вы уже имеете несколько коммитов:



Чтобы приступить к работе над изменениями для Чёрной пятницы, создадим специальную ветку:

git branch black-friday

\* Если вы ошиблись в названии, например, допустили опечатку, вы можете изменить название ветки с помощью команды: git branch -m старое-имя-ветки новое-имя-ветки.

И сразу переключимся на эту ветку:

git checkout black-friday

После выполнения этих команд состояние репозитория изменится.

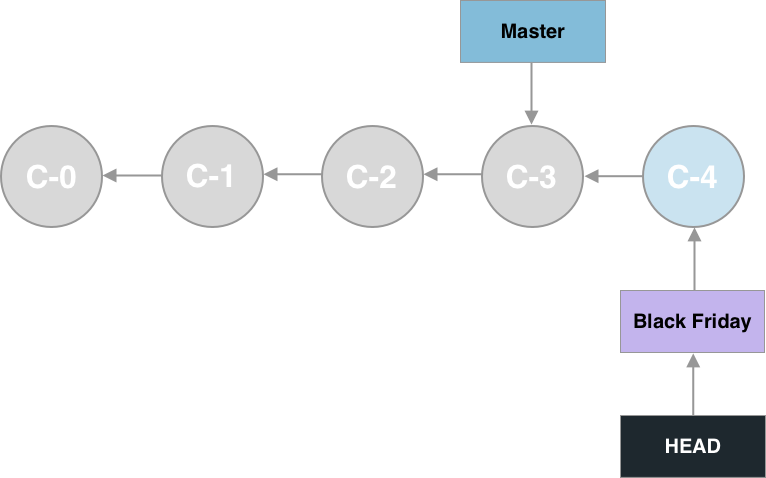
После этого мы можем менять файлы проекта и не бояться, что наш сайт как-то пострадает, ведь мы делаем это в изолированной ветке.

Итак, допустим, мы изменили файл index.html. Теперь мы хотим зафиксировать эти изменения.

git add index.html

git commit -m ''basic changes for Black Friday''

Что изменилось на этот раз?



После того как внесены все необходимые изменения, протестированы и согласованы, приходит время обновлять главную ветку *master*. Для простоты будем считать, что именно ветка master отвечает за версию проекта, которую все пользователи видят на сайте, хотя это не всегда так.

Чтобы внести изменения из одной ветки в другую, нужно выполнить команду **git merge**. По-русски эти команды часто называют «вмержить», «смержить» или «сделать мерж», хотя, конечно, это всё производные от английских терминов. Правильно на русский язык *merge* можно перевести как «слияние», поэтому говорят также «слияние веток».

**Слияние** — это способ соединить обратно раздвоенную историю. Команда git merge позволяет взять независимые части разработки и интегрировать их в одну ветку.

Если бы мы хотели влить новые изменения в текущую ветку (*Black Friday*), то надо было бы выполнить команду: git merge master, что означало бы, что в текущую ветку мы хотим влить ветку с именем *master*. Это очень полезно, когда программист долго разрабатывает какой-то новый функционал, а за это время в основной ветке происходят какие-то изменения, с которыми ему необходимо считаться.

В нашем же случае нам надо наоборот в ветку *master* влить *black-friday*. Как это можно сделать?

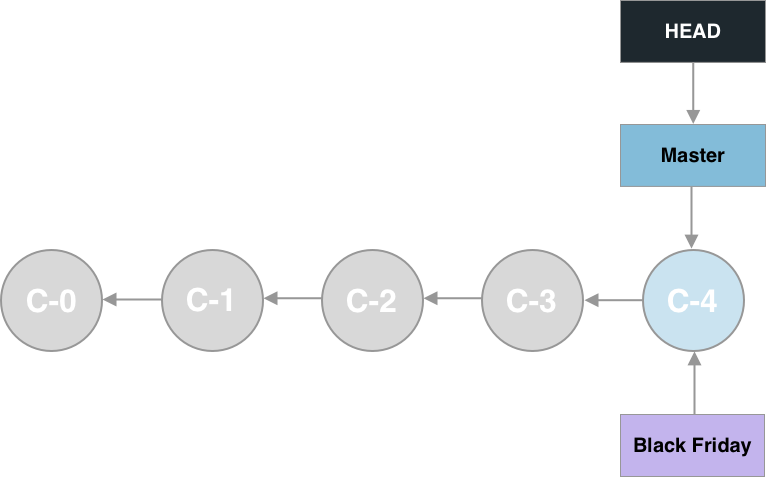
Для этого нам надо сначала переключиться на ту ветку, в которую мы хотим вливать, а затем уже «мержить».

git checkout master

git merge black-friday

Ура! Изменения для Чёрной пятницы теперь доступны всем!

Посмотрим, как выглядит репозиторий теперь:



Конечно, это очень простой пример. В реальной жизни веток бывает очень много, они разрабатываются параллельно и при необходимости сливаются (не всегда именно с главной).

Если все изменения уже находятся в главной ветке и Чёрная пятница прошла, нужна ли отдельная ветка, которая указывает на последний коммит этих изменений?

Чаще всего нет: если в ветке ничего не разрабатывается, то она продолжает указывать на один и тот же коммит, который с течением разработки остаётся где-то далеко в истории коммитов. Если вам когда-нибудь понадобится вернуться к этой версии или посмотреть, какие изменения были тогда внесены, для этого будет достаточно воспользоваться механизмом коммитов.

Значит, её нужно удалять, потому что незачем хранить лишнее. Чтобы удалить ветку, достаточно выполнить команду:

git branch -d black-friday

Как только вы перестали работать с веткой и слили все изменения с основной, можете удалять ветку без потери истории изменений. Обычно ветку оставляют до тех пор, пока не удостоверятся, что новый функционал работает как ожидалось, а затем сразу удаляют.

Даже если вы пользуетесь приложением с графическим интерфейсом и планируете делать так всегда, необходимо понимать, как работают основные команды в *Git*, ведь когда вы нажимаете ту или иную кнопку графического интерфейса, работает тот же механизм, что и при работе с командами *Git* в консоли.

**Проделаем аналогичные действия**

Пример с Чёрной пятницей пока может выглядеть для вас достаточно пространно, так как это упрощённый пример работы над реальным проектом (который у вас ещё впереди).

Создадим ветку в локальном репозитории. Откройте *Git CMD*. Перейдите в каталог локального репозитория (ранее мы назвали его *Skillfactory*). В настоящий момент вы находитесь в ветке *master*. Запомните, какие есть файлы в рабочем каталоге. Файлы, которые там есть, соответствуют ветке *master*.

Создадим новую ветку в *Git* с помощью известной нам команды git branch, после которой следует имя создаваемой ветки. Назовём её следующим образом:

git branch testbranch00

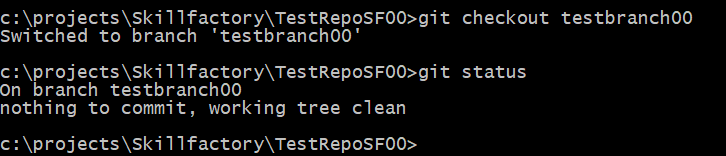
\* Вы можете использовать своё имя ветки. Далее по тексту будет использоваться имя ветки *testbranch00*.

Выполните следующие команды:

* 1. Для проверки состояния выполните команды git status и git log. Последняя показывает историю изменений: перечисляет коммиты, сделанные в репозитории, в обратном хронологическом порядке. Последние коммиты находятся вверху.
  2. Команду git branch --list. Она выведет список веток, которые существуют в вашем локальном репозитории. Текущая ветка *master* будет выделена зелёным цветом.
  3. Команду git branch --list -v . Вы увидите информацию о последних коммитах каждой из веток. Так как коммитов в ветку *testbranch00* ещё не было, то она указывает на коммит в *master*, от которого была создана.

Выполнение команд *git branch --list*

* 1. Команда git checkout переключит нас на созданную ветку. Выполняя команду git checkout, укажите после неё имя ветки, на которую вы хотите переключиться. Например, git checkout testbranch00. Вы получите сообщение о переключении на ветку *testbranch00*.
  2. Выполните команду git status. Вы увидите сообщение о том, что вы находитесь на ветке *testbranch00*.

Выполнение команды *git checkout*

**Задание 23.9**

Добавьте какой-нибудь (лучше текстовый) файл в репозиторий, добавьте его в индекс и зафиксируйте изменения. Синхронизировать с удалённым сервером пока не нужно.

Для проверки выполните команду git status после каждого шага.

Подсказка

Вот что получается на Mac

**Задание 23.10**

1 point possible (graded)

Переключитесь на ветку *master* и выполните команду dir если вы работаете в *Windows* или ls для других ОС. Видите ли вы только что созданный вами файл в списке файлов?

Да

Нет