

# Git

- master
- Слияние веток (merge/rebase/fast-forward)
- Смотрим/сравниваем (git show/git diff), навигация
- Работа с ветками

# master

Хороший курс: <https://smartiqa.ru/courses/git/lesson-1>

# Слияние веток (merge/rebase/fast-forward)

SRC: <https://selectel.ru/blog/tutorials/how-to-rebase-commits-and-branches/>

Объединить две ветки в гите можно по разному:

- rebase - создание новых, но таких же коммитов
- merge - перенос коммитов в ветку как есть, без пересоздания

Результат один и тот же - (условно) ветка master будет в нашей ветке где мы работаем. У нас будут актуальные изменения из мастера и мы не потеряем наши наработки.

**Rebase:** В этом случае **наши коммиты** будут удалены и добавлены **в конец к мастер ветке**, в отличие от merge, когда коммиты объединяются между собой. Т.е. в итоге коммиты в нашей ветке будут такие:

1. Наше изменение
2. Наше изменение
3. Наше изменение
4. Изменения из мастера
5. ...

В случае с **merge** мы получим такое:

1. Наши изменения
2. Мастер
3. Мастер
4. Наши изменения
5. Мастер
6. ...
7. Наши изменения
8. ...

Очевидно, что использовать rebase гораздо удобнее, т.к. **наши изменения будут в конце и их будет проще рассматривать**, после мержа в мастер не надо будет искать, какие изменения были внесены, они будут идти друг за другом и все будет понятно.

Также при rebase коммиты применяются к мастеру один за другим, поэтому будет **проще разобаться в конфликтах**, т.к. будут видны изменения на момент самого старшего

коммита нашей ветки и будет проще понять, в какой момент нам помешали.

Однако у rebase есть и минусы.

- Теряется история коммитов, а именно их дата и хеши.
- Может быть переписан текст коммитов (что с одной стороны может быть полезно, в некоторых ситуациях)
- Коммиты могут быть объединены (что также может быть полезно в некоторых ситуациях). **При git merge невозможно объединить коммиты**

# Смотрим/сравниваем (git show/git diff), навигация

Небольшой ликбез.

Навигация по гиту:

`HEAD` - наша текущая `workdir`, т.е. где мы сейчас находимся. Используется если ветка/коммит не указана явно (например при `git diff`, см пример ниже)

`origin/` (например `origin/master`) - указание на то что надо смотреть в наш ремоут (удаленную репу, гитлаб например). Если у нас несколько ремоутов, то они так и указываются, например `gitlab/master`, `github/master`. Если мы сделаем из нашей ветки `git pull`, то ветка `master` не обновится! Поэтому если будешь сравнивать с мастером, сравнивай с `origin/master`.

`HEAD~1` - на один коммит старше нашего `HEAD`'а. Можно указать соотв-но `HEAD~3`, `HEAD~10`, `HEAD~100`, чтобы указать на 3/10/100 коммитов назад.

Иногда нам надо сравнить или какой-то файл или ветку с другой веткой. Это довольно интуитивно, тут даже особо нечего описывать, поэтому вот команды:

## git log

Показывает историю коммитов

```
# посмотреть историю коммитов вместе с измененными файлами и сами изменения:  
git log -p
```

## git diff

Показывает diff между коммитами и (опционально) файлами

```
# сравнить HEAD с master:
```

```
git diff ..master
```

```
# Сравнить файл из одного коммита с файлом из другого коммита:
```

```
git diff 0abcdea01:file.sls..329a4b3:file.sls
```

# git show

Просто выводит файл из коммита

```
git show 0abcdea01:file.sls
```

# Работа с ветками

Создать ветку:

```
git checkout my_cool_branch
```

Переключиться между ветками:

```
git branch
```

Удалить ветку:

```
git branch -d
```