При слиянии веток могут происходить конфликты. Их, естественно, стоит избегать, но как их разрешать, и что вообще делать, если вдруг вы с ними столкнулись?

Рассмотрим гипотетический проект с двумя ветками: *master* и *feature*. В обеих ветках был изменён файл file.txt. Допустим, мы находимся на ветке *feature* и хотим влить изменения из *master*. Выполняем уже известную команду:

git merge master

Но вместо сообщения об успешном слиянии вдруг видим следующее:

Merge conflict in file.txt

Automatic merge failed

Fix conflicts and then commit the result

Если вы откроете указанный файл, то его структура будет выглядеть следующим образом:

Какая-то одинаковая часть

<<<<<<< HEAD

часть, которая была изменена и сейчас так выглядит в текущей ветке

=======

часть, которая была изменена и сейчас так выглядит в ветке master

>>>>>>> master

Какая-то одинаковая часть

Конфликт возник между изменениями, которые находятся между “<<<<<<< HEAD” и “=======”, и изменениями между “=======” и “>>>>>>> master”. В данный момент файл содержит обе версии.

Если проверить сейчас git status, то вы увидите примерно следующее сообщение:

На ветке feature

У вас есть не слитые пути.

  (разрешите конфликты, затем запустите «git commit»)

  (используйте «git merge --abort», чтобы остановить операцию слияния)

Не слитые пути:

  (используйте «git add <файл>…», чтобы пометить разрешение конфликта)

оба изменены: file.txt

нет изменений, добавленных для коммита

(используйте «git add» и/или «git commit -a»)

Сейчас слияние находится в подвешенном состоянии. Его надо либо продолжить, либо прервать.

Если выполнить **git merge --abort**, как указано в инструкции, то всё вернётся в такое состояние, каким было на ветке *feature* до попытки слияния.

Чтобы продолжить слияние, исправьте в текстовом редакторе файл так, чтобы получить нужное состояние. В зависимости от ситуации вам понадобится использовать версию из ветки *feature* или из ветки *master*, или объединить изменения. В любом случае в финальной версии файла не должны содержаться специальные вставки **git merge** “<<<<<<< HEAD”, “=======” и “>>>>>>> master”.

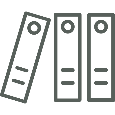
После того как вы исправили **file.txt**, его стоит добавить как перед обычным коммитом с помощью **git add**.

После этого запустите **git commit**, чтобы завершить слияние.

В ходе разработки настоящих проектов у программистов всё время возникают конфликты. Разрешение конфликтов — неотъемлемая часть работы. Поэтому не бойтесь, экспериментируйте сейчас, нарабатывайте опыт и преодолевайте страх перед *Git*, терминалом и конфликтами в частности.

К работе с *Git* нужно привыкнуть. Даже у опытных специалистов время от времени возникают проблемы при взаимодействии с системой контроля версий. *Git* имеет очень много возможностей, что добавляет гибкости этому инструменту, но в то же время в них легко запутаться.

Чтобы вам было проще ориентироваться в его возможностях, вот небольшой **список ресурсов**, который может в этом помочь.

**   Дополнительно:**

* + - [документация Git](https://git-scm.com/docs); https://git-scm.com/docs
    - [книга Pro Git (на русском)](https://git-scm.com/book/ru/v2); https://git-scm.com/book/ru/v2
    - [гид по изучению Git от Atlassian](https://www.atlassian.com/git/tutorials/learn-git-with-bitbucket-cloud); https://www.atlassian.com/git/tutorials/learn-git-with-bitbucket-cloud
    - [шпаргалка](https://github.github.com/training-kit/downloads/ru/github-git-cheat-sheet/);
    - [Git Handbook](https://guides.github.com/introduction/git-handbook/); https://github.github.com/training-kit/downloads/ru/github-git-cheat-sheet/
    - поиграть с визуализацией можно [здесь](http://git-school.github.io/visualizing-git/); http://git-school.github.io/visualizing-git/
    - обязательно поупражняйтесь в [этом тренажёре](https://learngitbranching.js.org/). https://learngitbranching.js.org/