Дорожная карта

0. Приступаем
1. Введение в Git
2. Начало работы с Git
3. Просмотр истории
4. Основы ветвления 👈
5. Слияние
6. Управление ветками
7. Отмена изменений
8. Рабочий процесс

Дорожная карта (продолжение)

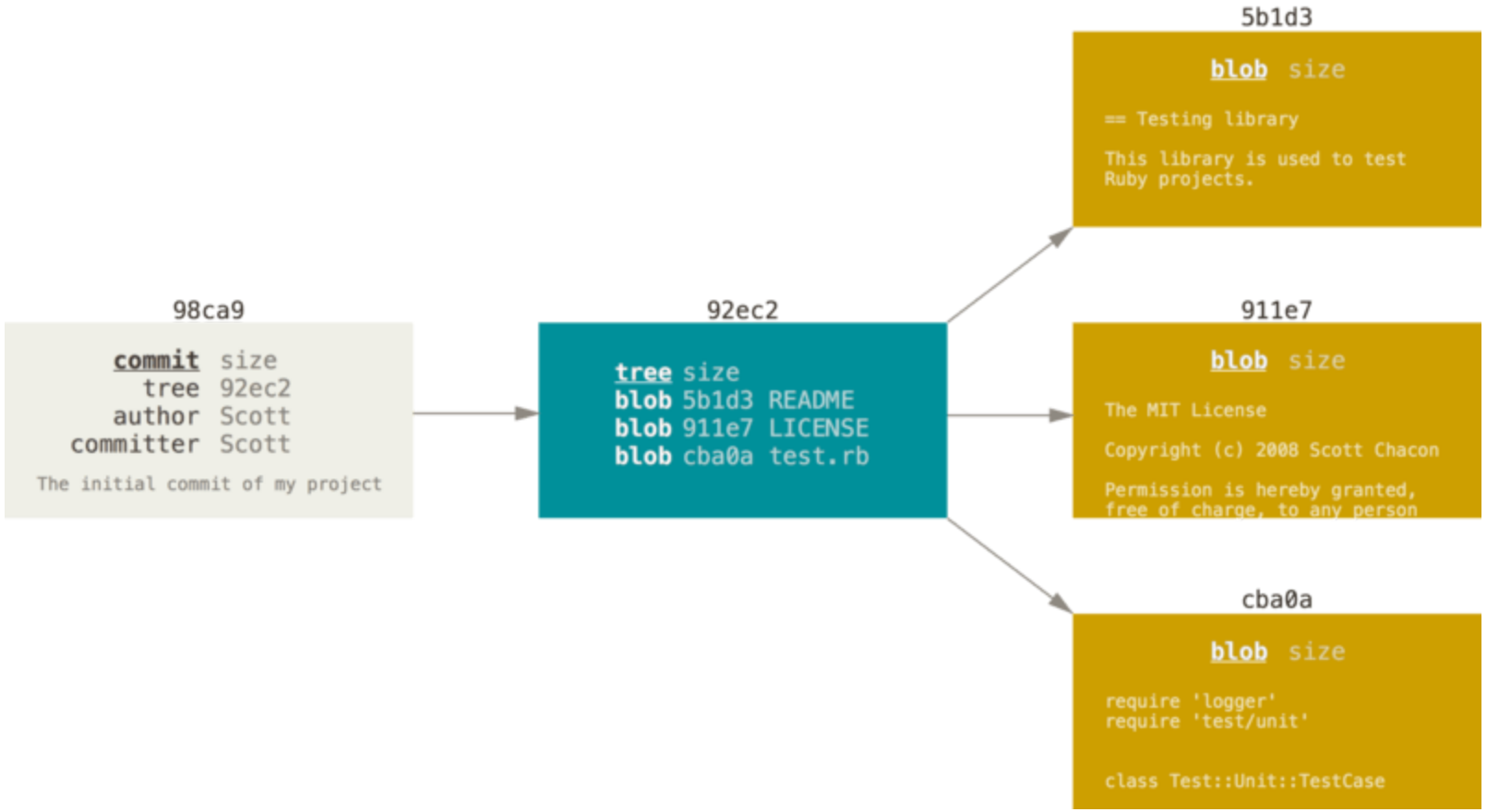
9. Работа в команде
10. Метки
11. Технические тонкости
12. Последние штрихи

Основы ветвления

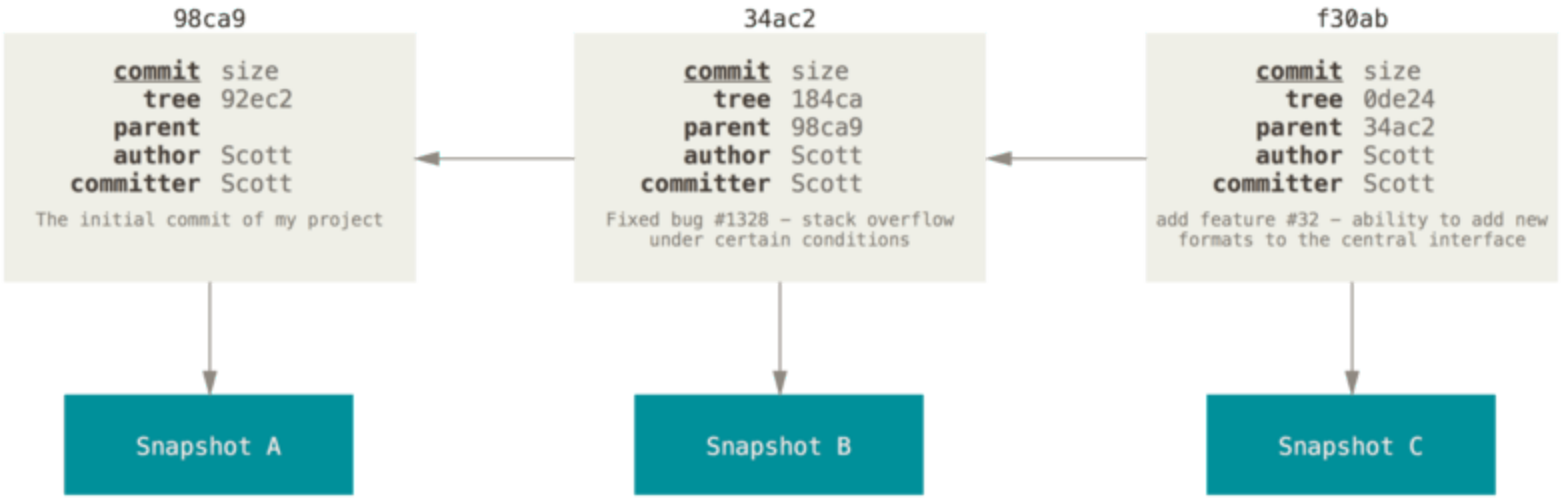
- Почти каждая система контроля версий (СКВ) в какой-то форме поддерживает ветвление.
- Во многих СКВ создание веток - это очень затратный процесс.
- В Git ветвление очень легковесно.
- Git поощряет процесс работы, при котором ветвление и слияние выполняется часто.

Как Git хранит данные

```
$ git add README test.rb LICENSE
$ git commit -m 'initial commit of my project'
```

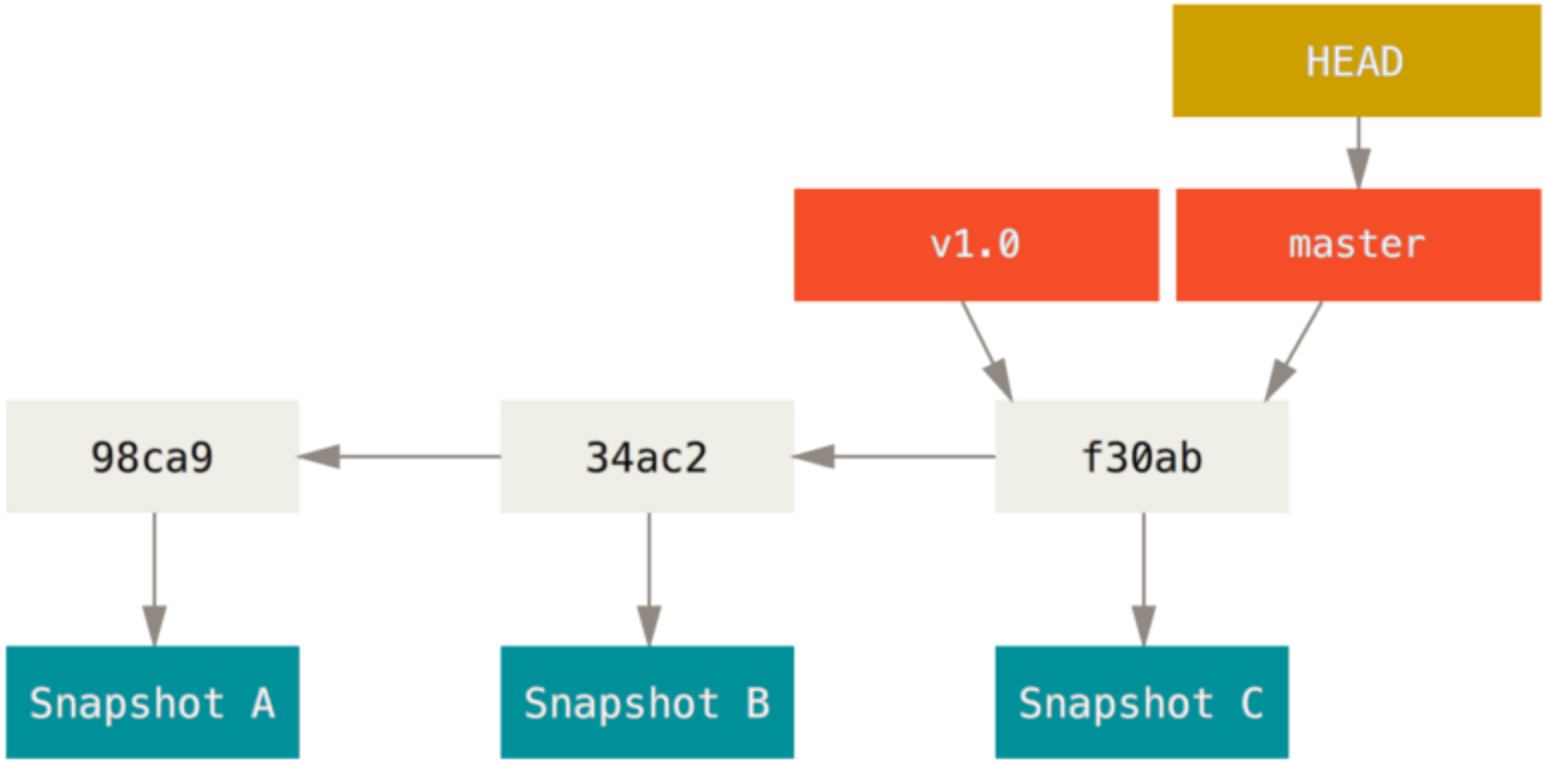


История коммитов



Ветка в Git

Ветка (branch) в Git - это легко перемещаемый указатель на один из этих коммитов. Имя основной ветки по умолчанию в Git - `master`.



Пример рабочего процесса

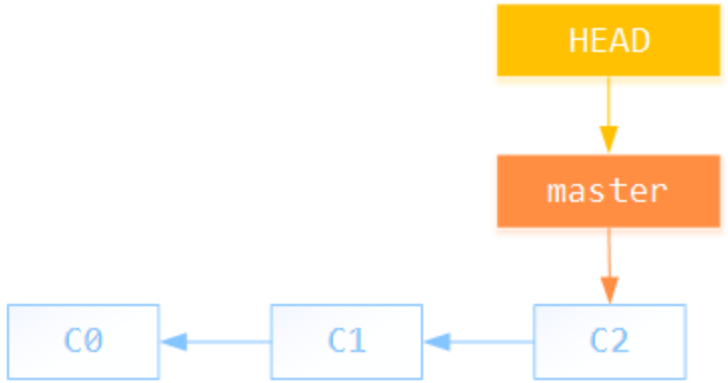
Ваша работа построена так:

1. Вы работаете над проектом.
2. Вы создаете ветку для тестирования.
3. Вы работаете в этой ветке.

Приходит сообщение о критической ошибке, требующей немедленного исправления. Ваши действия:

1. Переключиться на основную ветку.
2. Создать ветку для добавления исправления.
3. После тестирования слить ветку содержащую исправление с основной веткой.
4. Переключиться назад на ветку тестирования, и закончить тесты.

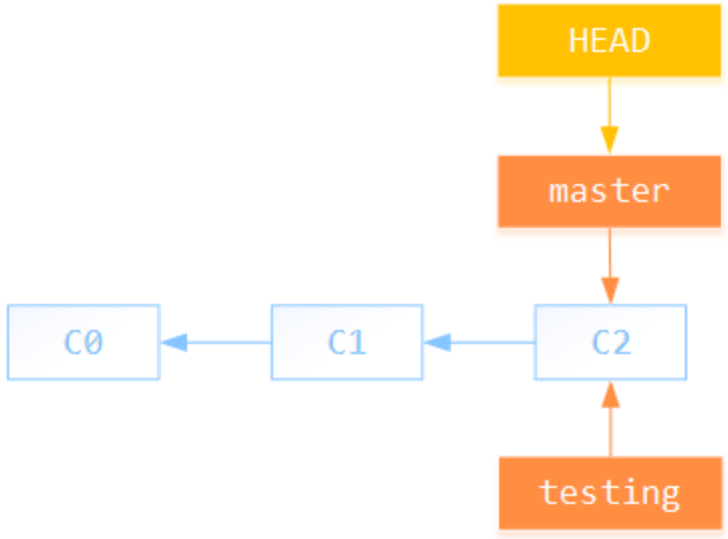
Пример → Текущая ветка




```
$ git log --oneline --decorate
f30ab (HEAD, master) add feature #32 - ability to add new
34ac2 fixed bug #1328 - stack overflow under certain conditions
98ca9 initial commit of my project
```

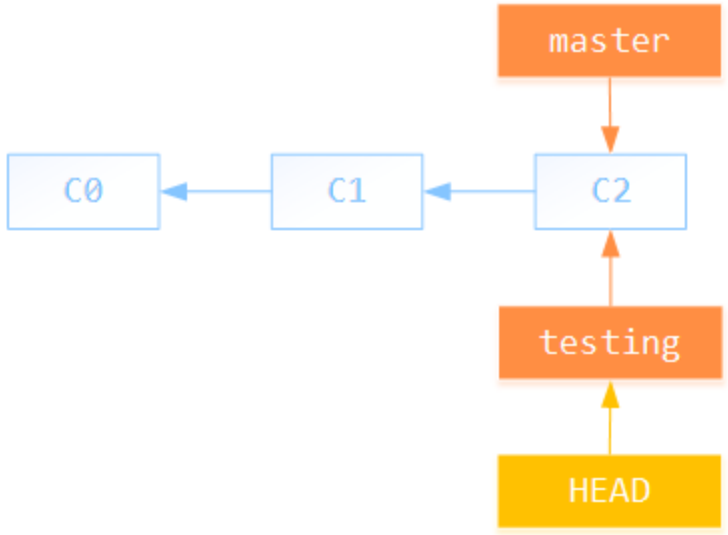
Пример → Создание нового указателя ветки

```
$ git branch testing
```



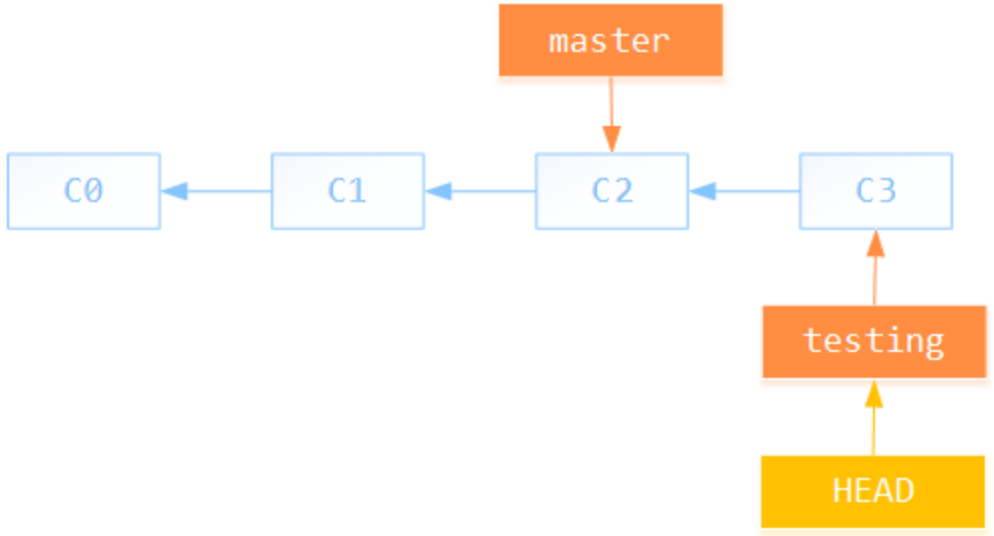
Пример → Переключение ветки

```
$ git checkout testing
Switched to branch 'testing'
```



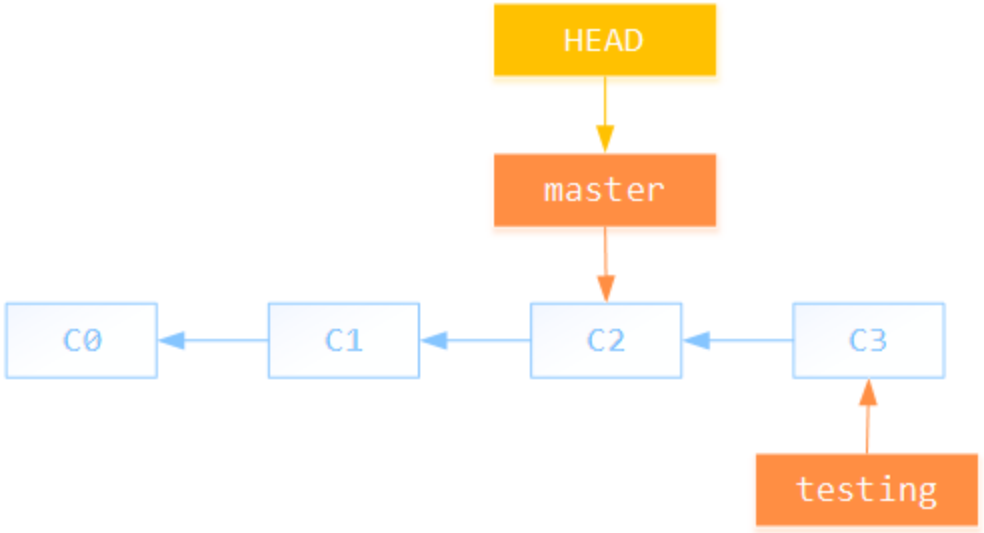
Пример → Указатель на ветку `HEAD` переместился вперёд после коммита

```
$ vim test.rb
$ git commit -a -m 'add unittests'
```



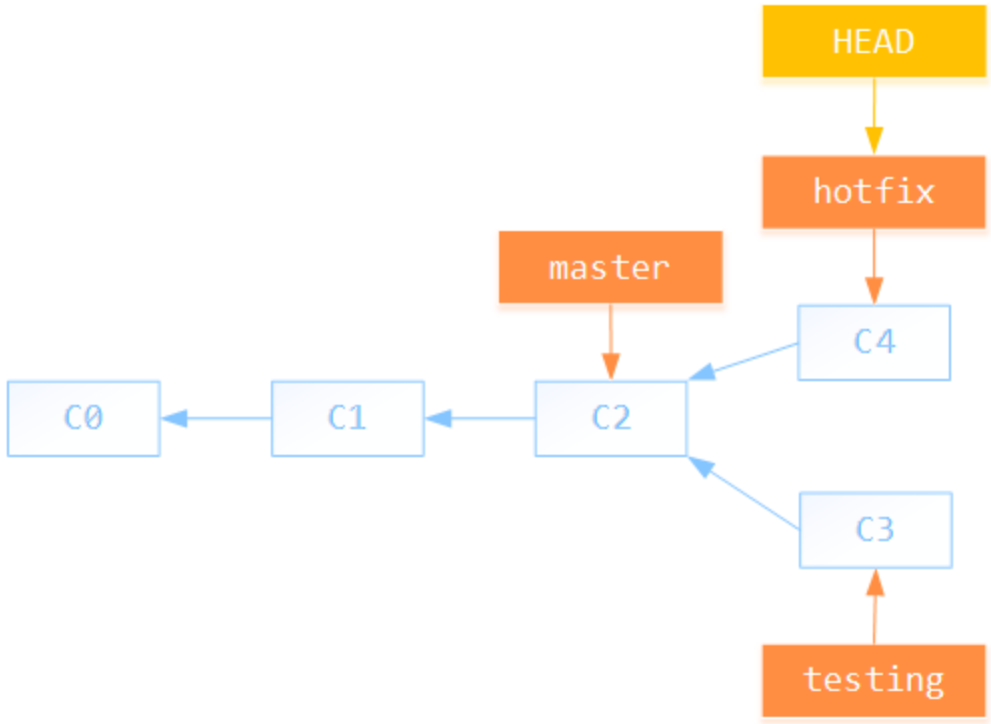
Пример → `HEAD` перемещается когда вы делаете checkout

```
$ git checkout master
Switched to branch 'master'
```



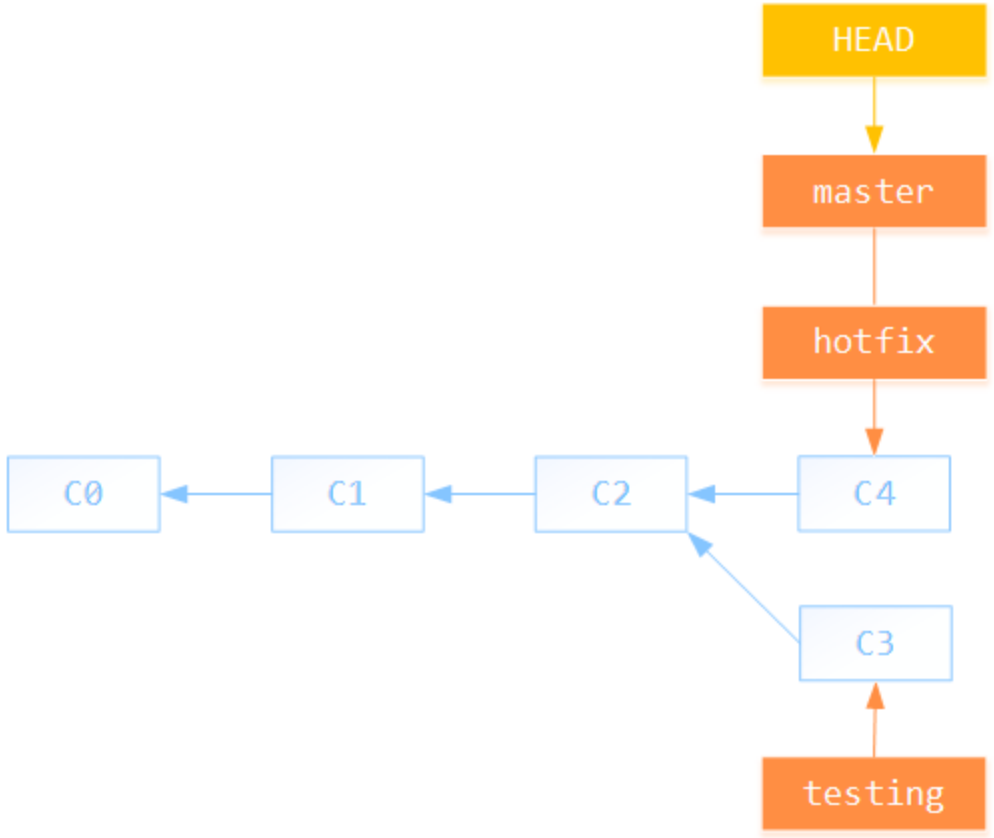
Пример → Ветка hotfix основана на ветке master

```
$ git checkout -b hotfix
Switched to a new branch 'hotfix'
$ vim index.html
$ git commit -a -m 'fixed the broken email address'
[hotfix 1fb7853] fixed the broken email address
 1 file changed, 2 insertions(+)
```



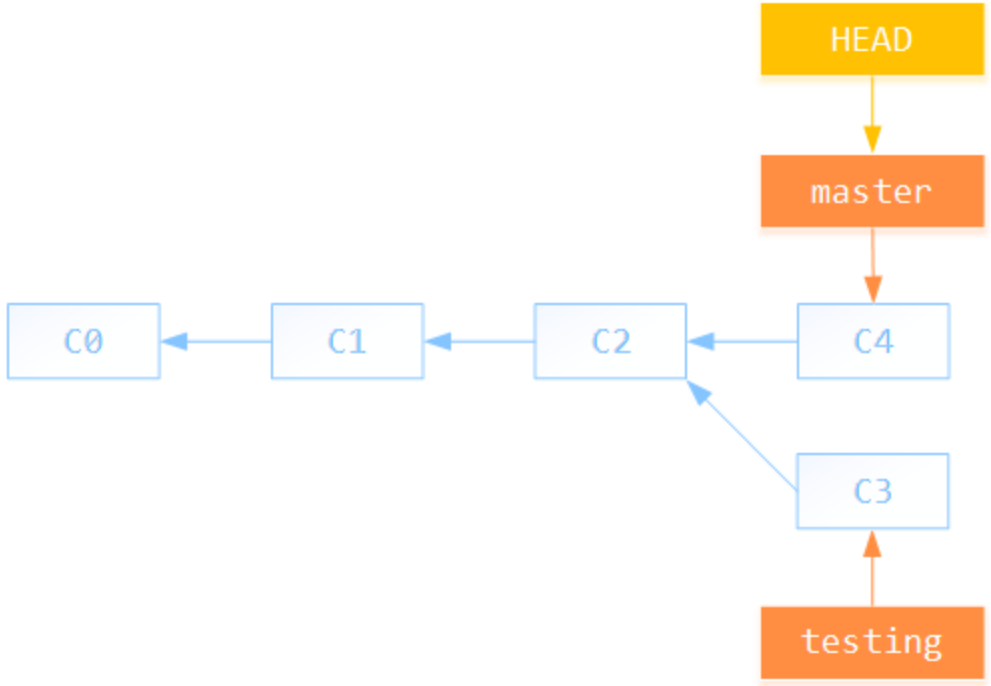
Пример → `master` перемотан до `hotfix`

```
$ git checkout master
$ git merge hotfix
Updating f42c576..3a0874c
Fast-forward
 index.html | 2 ++
 1 file changed, 2 insertions(+)
```



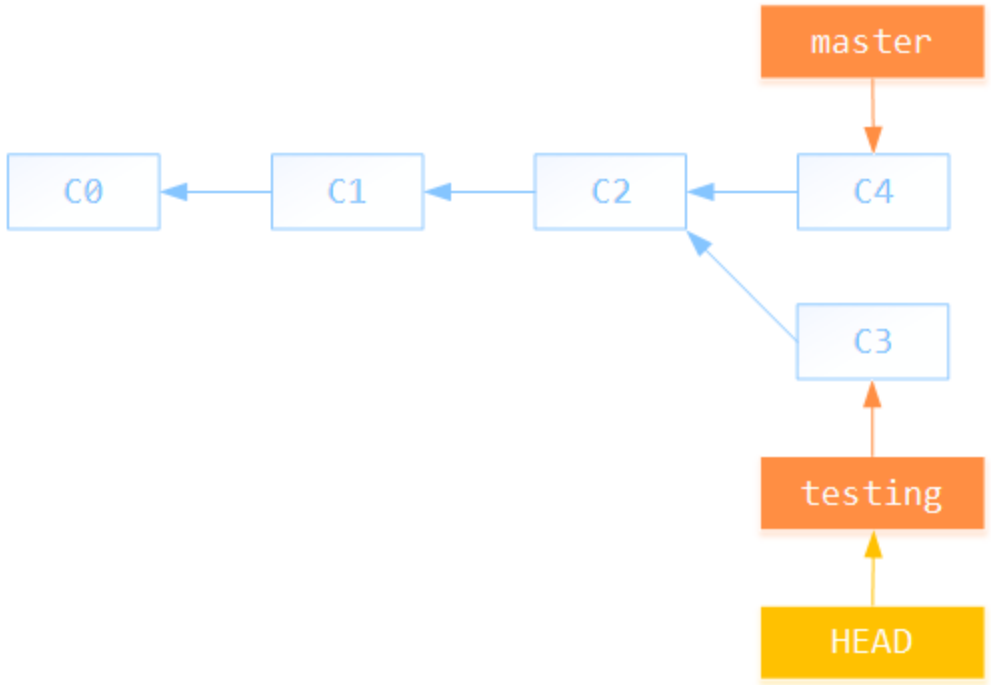
Пример → Удаление ветки `hotfix`

```
$ git branch -d hotfix
Deleted branch hotfix (3a0874c).
```



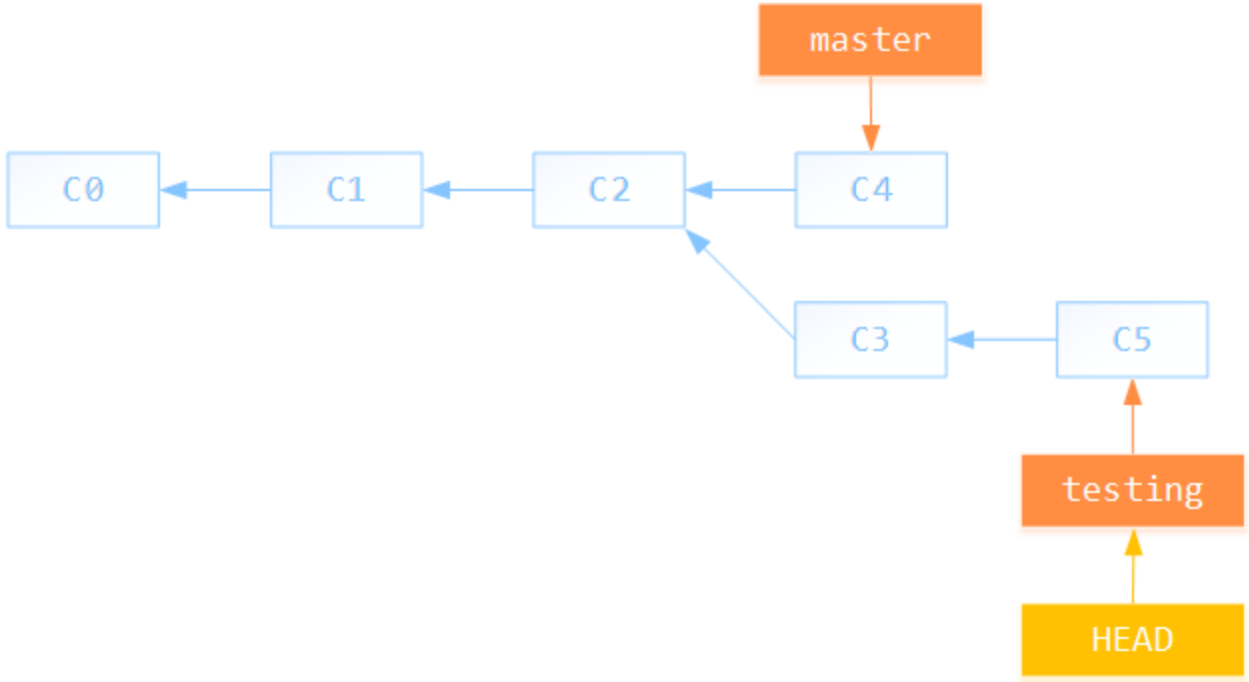
Пример → Переключение на `testing`

```
$ git checkout testing
Switched to branch "testing"
```

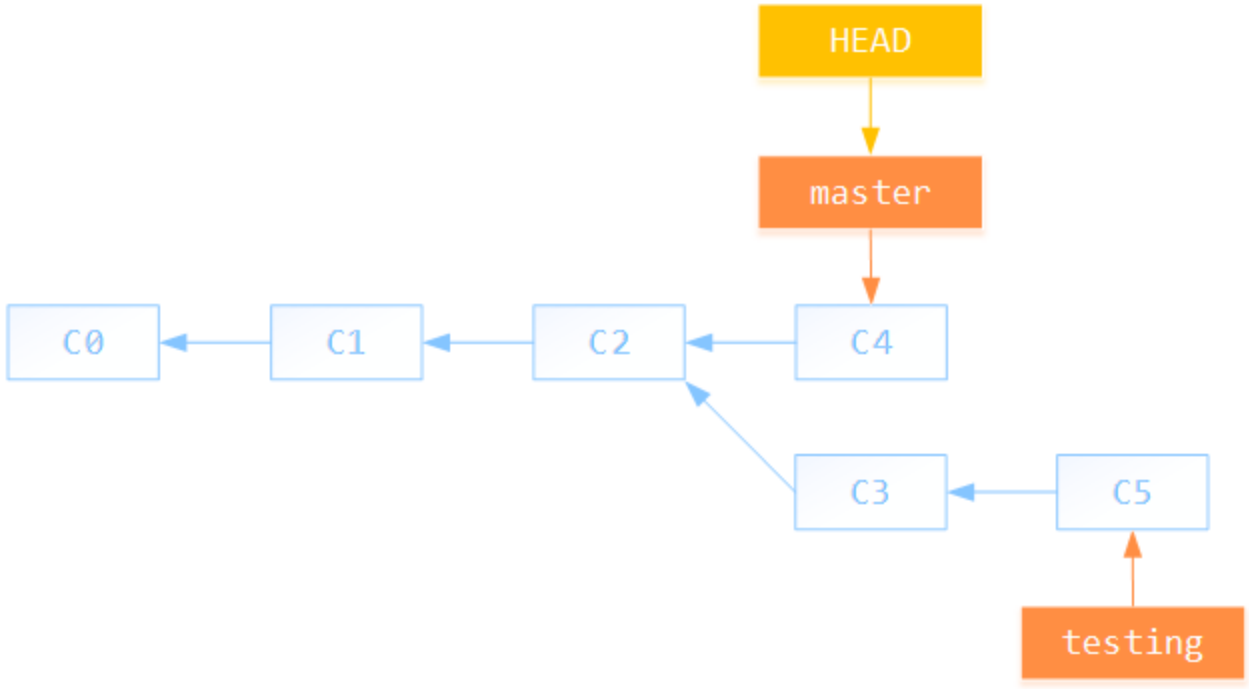


Пример → Продолжение работы над `testing`

```
$ git checkout testing
Switched to branch "testing"
$ vim index.html
$ git commit -a -m 'finished the new footer [issue 53]'
[testing ad82d7a] finished the new footer [issue 53]
1 file changed, 1 insertion(+)
```



Пример → Переключение на `master`



Управление ветками

Получение списка веток

Команда `git branch` без параметров или `git branch --list` отображает список *локальных* веток. Текущая ветка отмечена символом `*`.

```
$ git branch
  hotfix
* master
  testing
```

Параметр `-v` (`--verbose`) показывает информацию последнего коммита для каждой ветки.

```
$ git branch -v
  hotfix 782fd34 add scott to the author list in the readmes
* master  7a98805 Merge branch 'testing'
  testing  93b412c fix javascript issue
```

Получение списка веток (продолжение)

Опция `-a` или `--all` позволяет отобразить список *всех* веток (локальных + удалённых):

```
$ git branch --all
* master
  remotes/origin/master
```

Получение списка веток (продолжение)

Опция `--merged` показывает в списке ветки, которые уже влиты в текущую:

```
$ git branch --merged
  hotfix
* master
```

Опция `--no-merged` показывает в списке ветки, которые ещё *не* влиты в текущую:

```
$ git branch --no-merged
  testing
```

Создание ветки

Команда `git branch <branch> [commit]` позволяет создать ветку с именем `branch` на текущем или указанном коммите `commit`.

Пример:

```
$ git branch
* master

$ git branch jira_234

$ git branch
  jira_234
* master
```

Переключение на ветку

Команда `git checkout <branch>` позволяет переключиться на указанную ветку `branch`. Она обновляет содержимое рабочей папки согласно послендему коммиту в этой ветке.

```
$ git checkout jira_234
Switched to branch 'jira_234'

$ git checkout master
Switched to branch 'master'
```

Удаление ветки

Команда `git branch -d <branch>` *безопасно* удаляет указанную ветку:

```
$ git branch -d jira_234
error: The branch 'jira_234' is not fully merged.
If you are sure you want to delete it, run 'git branch -D jira_234'.
```

Команда `git branch -D <branch>` *гарантированно* удаляет указанную ветку:

```
$ git branch -D jira_234
Deleted branch jira_234 (was 7d7abf4).
```

Переименование ветки

Команда `git branch -m <newname>` переименовывает текущую ветку в `<newname>`.

```
git branch -m <newname>
git branch -m <oldname> <newname>
```

Осиротелые ветки

"Осиротелые" коммиты не имеют родительских коммитов.
Необходимы, чтобы вести несколько независимых историй в одном репозитории, например:

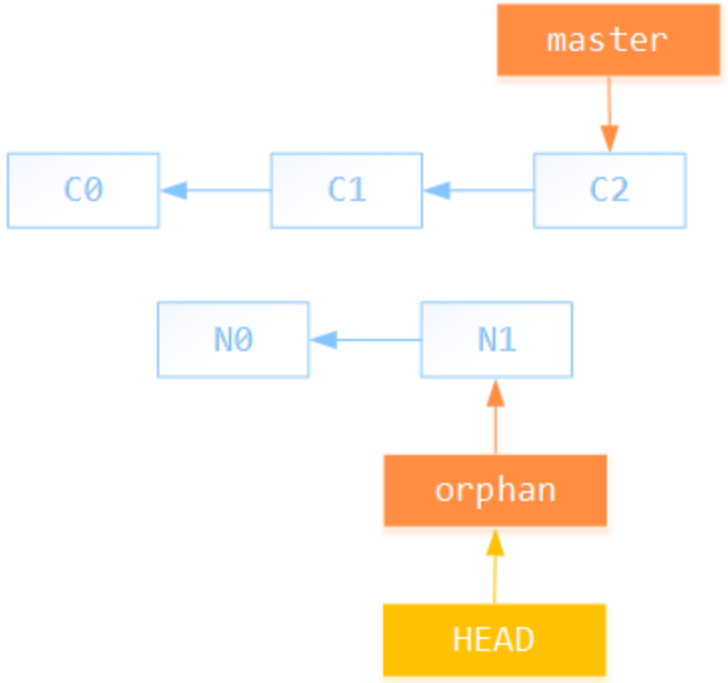
- вспомогательная конфигурация проекта в отдельной ветке
- несколько проектов в одном репозитории
- документация и программный код разделены

Создаются с помощью команды

```
git checkout --orphan <name>
git rm -rf .
<do work>
git add <your files>
git commit -m 'Initial commit'
```

Изначально файлы с прошлого коммита добавлены в индекс.

Осиротелые ветки (продолжение)



Практика 💪

1. Получите список локальных веток.
2. Получите список всех веток.
3. Создайте локальную ветку `test` и переключитесь на неё.
4. Добавьте несколько файлов в рабочей директории и создайте новый коммит.
5. Посмотрите историю коммитов
6. Переключитесь на ветку `master` (`main`). Убедитесь, что файлы созданные на шаге 4 отсутствуют. Почему так?
7. Отобразите список всех коммитов всех веток в виде графа (`git log --graph --oneline --decorate --all`)

Заключение

Ветка в Git - это именнованый указатель на коммит.

`HEAD` - указатель на текущий коммит.

Команда `git branch` позволяет управлять ветками:

- посмотреть список веток
- создать ветку
- удалить ветку
- переименовать ветку
- другое (см. справку)

Команда `git checkout` позволяет переключиться на указанную ветку.

Ваши вопросы ?