

**..МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ
ФЕДЕРАЦИИ**

**Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Кафедра инфокоммуникаций
«Основы ветвления GIT»**

**Отчет по лабораторной работе № 1.3
по дисциплине «Основы программной инженерии»**

Выполнил студент группы ПИЖ-б-о-21-1

Логвинов Иван Васильевич « » 2022г.

Подпись студента _____

Работа защищена « » _____ 20__ г.

Проверил Воронкин Р.А. _____

(подпись)

Ставрополь 2022

Цель работы: исследование базовых возможностей по работе с локальными и удаленными ветками Git.

Ход работы:

1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT.

Создайте новый репозиторий

Репозиторий содержит все файлы проекта, включая историю изменений. У вас уже есть репозиторий проекта в другом месте? [Импортируйте репозиторий](#).

Владелец *

 IvanLogvinov11 ▾

Имя репозитория *

/ lab3_3 ✓

Отличные названия репозитория короткие и запоминающиеся. Вам нужно вдохновение? Как насчет **вероятной брокколи?**

Описание (необязательно)



Общедоступный

Любой пользователь Интернета может увидеть этот репозиторий. Вы выбираете, кто может совершать.



Личное

Вы выбираете, кто может видеть и фиксировать в этом репозитории.

Инициализируйте этот репозиторий с помощью:

Пропустите этот шаг, если вы импортируете существующий репозиторий.



Добавьте файл README

Здесь вы можете написать длинное описание для вашего проекта. [Узнать больше](#).

Добавить .gitignore

Выберите, какие файлы не отслеживать, из списка шаблонов. [Узнать больше](#).

.шаблон gitignore: Python ▾

Выберите лицензию

Лицензия сообщает другим пользователям, что они могут и чего не могут делать с вашим кодом. [Узнать больше](#).

Лицензия: MIT License ▾

Это установит  main в качестве ветки по умолчанию. Измените имя по умолчанию в настройках.



Вы создаете общедоступный репозиторий в своем личном кабинете.

Создать репозиторий

Рисунок 1 – Создание общедоступного репозитория

3. Создать три файла: 1.txt, 2.txt, 3.txt.



Рисунок 2 – Создание файлов

4. Проиндексировать первый файл и сделать коммит с комментарием "add 1.txt file".

```
C:\Users\Иван\Desktop>git add 1.txt
fatal: not a git repository (or any of the parent directories): .git

C:\Users\Иван\Desktop>cd lab3_3

C:\Users\Иван\Desktop\lab3_3>git add 1.txt

C:\Users\Иван\Desktop\lab3_3>git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   1.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    2.txt
    3.txt

C:\Users\Иван\Desktop\lab3_3>
```

Рисунок 3 – Индексация первого файла и создание коммита

5. Проиндексировать второй и третий файлы.

```

C:\Users\Иван\Desktop\lab3_3>git add 2.txt

C:\Users\Иван\Desktop\lab3_3>git add 3.txt

C:\Users\Иван\Desktop\lab3_3>git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   1.txt
        new file:   2.txt
        new file:   3.txt

C:\Users\Иван\Desktop\lab3_3>

```

Рисунок 4 – Добавление файлов 2 и 3 в индекс

6. Переписать уже сделанный коммит с новым комментарием "add 2.txt and 3.txt."

```

C:\Users\Иван\Desktop\lab3_3>git commit --amend -m "add 2.txt 3.txt"
[main b04beff] add 2.txt 3.txt
Author: Ivan <112752273+IvanLogvinov11@users.noreply.github.com>
Date: Fri Nov 25 22:57:21 2022 +0300
6 files changed, 151 insertions(+)
create mode 100644 .gitignore
create mode 100644 1.txt
create mode 100644 2.txt
create mode 100644 3.txt
create mode 100644 LICENSE
create mode 100644 README.md

C:\Users\Иван\Desktop\lab3_3>

```

Рисунок 5 – Перепись коммита

7. Создать новую ветку my_first_branch.

```

C:\Users\Иван\Desktop\lab3_3>git branch my_first_branch

C:\Users\Иван\Desktop\lab3_3>git branch
* main
  my_first_branch

C:\Users\Иван\Desktop\lab3_3>

```

Рисунок 6 – Создание новой ветки

8. Перейти на ветку и создать новый файл in_branch.txt, закоммитить изменения.

```
C:\Users\Иван\Desktop\lab3_3>git checkout my_first_branch
Switched to branch 'my_first_branch'

C:\Users\Иван\Desktop\lab3_3>git branch
  main
* my_first_branch

C:\Users\Иван\Desktop\lab3_3>git add .

C:\Users\Иван\Desktop\lab3_3>git status
On branch my_first_branch
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   in_branch.txt

C:\Users\Иван\Desktop\lab3_3>git commit -m "add in_branch.txt"
[my_first_branch 4d2bc3f] add in_branch.txt
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 in_branch.txt

C:\Users\Иван\Desktop\lab3_3>
```

Рисунок 7 – Создание коммита в новой ветке

9. Вернуться на ветку master.

```
C:\Users\Иван\Desktop\lab3_3>git checkout main
Switched to branch 'main'
Your branch and 'origin/main' have diverged,
and have 1 and 1 different commits each, respectively.
  (use "git pull" to merge the remote branch into yours)

C:\Users\Иван\Desktop\lab3_3>git branch
* main
  my_first_branch

C:\Users\Иван\Desktop\lab3_3>
```

Рисунок 10 – Возвращение на ветку «main»

10. Создать и сразу перейти на ветку new_branch.

```
C:\Users\Иван\Desktop\lab3_3>git checkout -b new_branch
Switched to a new branch 'new_branch'

C:\Users\Иван\Desktop\lab3_3>git branch
  main
  my_first_branch
* new_branch

C:\Users\Иван\Desktop\lab3_3>
```

Рисунок 11 – Одновременное создание и переход на ветку

11. Сделать изменения в файле 1.txt, добавить строчку “new row in the 1.txt file”, закоммитить изменения.

```
C:\Users\Иван\Desktop\lab3_3>git add .  
  
C:\Users\Иван\Desktop\lab3_3>git commit -m "add inform in 1.txt"  
[new_branch cba2dfd] add inform in 1.txt  
1 file changed, 1 insertion(+)  
  
C:\Users\Иван\Desktop\lab3_3>
```

Рисунок 12 – Создание изменений в файле 1.txt и коммита

12. Перейти на ветку master и слить ветки master и my_first_branch, после чего слить ветки master и new_branch.

```
C:\Users\Иван\Desktop\lab3_3>git checkout main  
Switched to branch 'main'  
Your branch and 'origin/main' have diverged,  
and have 1 and 1 different commits each, respectively.  
(use "git pull" to merge the remote branch into yours)  
  
C:\Users\Иван\Desktop\lab3_3>git merge my_first_branch  
Updating b04beff..4d2bc3f  
Fast-forward  
  in_branch.txt | 0  
  1 file changed, 0 insertions(+), 0 deletions(-)  
  create mode 100644 in_branch.txt  
  
C:\Users\Иван\Desktop\lab3_3>git merge new_branch  
Merge made by the 'ort' strategy.  
  1.txt | 1 +  
  1 file changed, 1 insertion(+)  
  
C:\Users\Иван\Desktop\lab3_3>
```

Рисунок 13 – Слияние веток main, my_first_branch и new_branch

13. Удалить ветки my_first_branch и new_branch.

```
C:\Users\Иван\Desktop\lab3_3>git branch -d my_first_branch  
Deleted branch my_first_branch (was 4d2bc3f).  
  
C:\Users\Иван\Desktop\lab3_3>git branch -d new_branch  
Deleted branch new_branch (was cba2dfd).  
  
C:\Users\Иван\Desktop\lab3_3>git branch  
* main  
  
C:\Users\Иван\Desktop\lab3_3>
```

Рисунок 14 – Удаление веток

14. Создать ветки branch_1 и branch_2.

```
C:\Users\Иван\Desktop\lab3_3>git branch branch_1
C:\Users\Иван\Desktop\lab3_3>git branch branch_2
C:\Users\Иван\Desktop\lab3_3>git branch
  branch_1
  branch_2
* main
C:\Users\Иван\Desktop\lab3_3>
```

Рисунок 15 – Создание новых веток

15. Перейти на ветку branch_1 и изменить файл 1.txt, удалить все содержимое и добавить текст “fix in the 1.txt”, изменить файл 3.txt, удалить все содержимое и добавить текст “fix in the 3.txt”, закоммитить изменения.

```
C:\Users\Иван\Desktop\lab3_3>git checkout branch_1
Switched to branch 'branch_1'

C:\Users\Иван\Desktop\lab3_3>git branch
* branch_1
  branch_2
  main

C:\Users\Иван\Desktop\lab3_3>git add .
C:\Users\Иван\Desktop\lab3_3>git add .

C:\Users\Иван\Desktop\lab3_3>git status
On branch branch_1
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   1.txt
        modified:   3.txt

C:\Users\Иван\Desktop\lab3_3>git commit -m "modified 1.txt 3.txt"
[branch_1 4bf2c87] modified 1.txt 3.txt
 2 files changed, 2 insertions(+), 1 deletion(-)

C:\Users\Иван\Desktop\lab3_3>
```

Рисунок 16 – Изменение файлов 1.txt и 3.txt, внесение их в коммит на ветке branch_1

16. Перейти на ветку `branch_2` и также изменить файл `1.txt`, удалить все содержимое и добавить текст “My fix in the 1.txt”, изменить файл `3.txt`, удалить все содержимое и добавить текст “My fix in the 3.txt”, закоммитить изменения.

```
C:\Users\Иван\Desktop\lab3_3>git checkout branch_2
Switched to branch 'branch_2'

C:\Users\Иван\Desktop\lab3_3>git branch
  branch_1
* branch_2
  main

C:\Users\Иван\Desktop\lab3_3>git add .

C:\Users\Иван\Desktop\lab3_3>git status
On branch branch_2
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   1.txt
        modified:   3.txt

C:\Users\Иван\Desktop\lab3_3>git commit -m "modified in br_2 1.txt 3.txt"
[branch_2 09204a7] modified in br_2 1.txt 3.txt
 2 files changed, 2 insertions(+), 1 deletion(-)

C:\Users\Иван\Desktop\lab3_3>
```

Рисунок 17 – Изменение файлов `1.txt` и `3.txt`, внесение их в коммит на ветке `branch_2`

17. Слить изменения ветки `branch_2` в ветку `branch_1`.

```
C:\Users\Иван\Desktop\lab3_3>git checkout branch_1
Switched to branch 'branch_1'

C:\Users\Иван\Desktop\lab3_3>git branch
* branch_1
  branch_2
  main

C:\Users\Иван\Desktop\lab3_3>git merge branch_2
Auto-merging 1.txt
CONFLICT (content): Merge conflict in 1.txt
Auto-merging 3.txt
CONFLICT (content): Merge conflict in 3.txt
Automatic merge failed; fix conflicts and then commit the result.

C:\Users\Иван\Desktop\lab3_3>
```

Рисунок 18 – Слияние веток `branch_1` и `branch_2`

18. Решить конфликт файла 1.txt в ручном режиме, а конфликт 3.txt используя команду `git mergetool` с помощью одной из доступных утилит, например Meld.

```
C:\Users\Иван\Desktop\lab3_3>git status
On branch branch_1
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Changes to be committed:
  modified:   1.txt

Unmerged paths:
  (use "git add <file>..." to mark resolution)
    both modified:   3.txt

C:\Users\Иван\Desktop\lab3_3>
```

Рисунок 19 – Решение конфликта вручную

```
C:\Users\Иван\Desktop\lab3_3>git mergetool

This message is displayed because 'merge.tool' is not configured.
See 'git mergetool --tool-help' or 'git help config' for more details.
'git mergetool' will now attempt to use one of the following tools:
tortoisemerge emerge vimdiff nvimdiff
Merging:
3.txt

Normal merge conflict for '3.txt':
  {local}: modified file
  {remote}: modified file
Hit return to start merge resolution tool (vimdiff):
```

Рисунок 19 – Решение конфликта с помощью mergetool

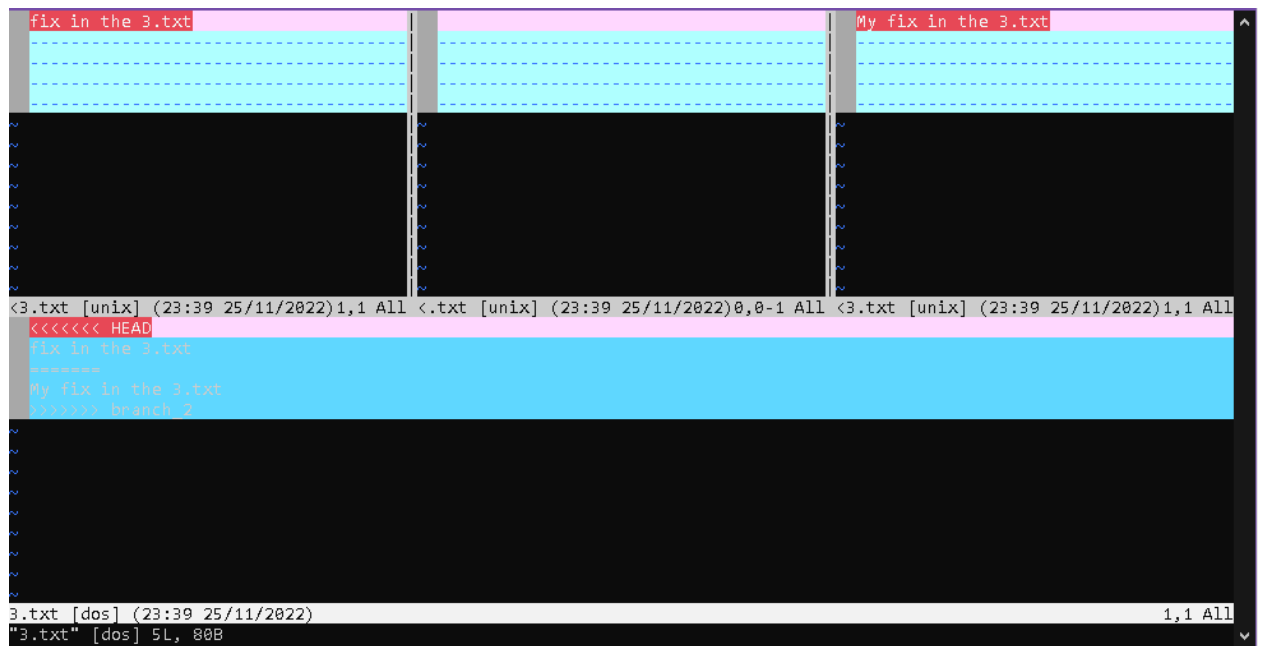


Рисунок 20 – Меню vimdiff

19. Отправить ветку branch_1 на GitHub.

```
C:\Users\Иван\Desktop\lab3_3>git push --set-upstream origin branch_1
Enumerating objects: 17, done.
Counting objects: 100% (17/17), done.
Delta compression using up to 8 threads
Compressing objects: 100% (12/12), done.
Writing objects: 100% (17/17), 2.83 KiB | 1.41 MiB/s, done.
Total 17 (delta 5), reused 3 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (5/5), done.
remote:
remote: Create a pull request for 'branch_1' on GitHub by visiting:
remote:   https://github.com/IvanLogvinov11/lab3_3/pull/new/branch_1
remote:
To https://github.com/IvanLogvinov11/lab3_3.git
 * [new branch]      branch_1 -> branch_1
branch 'branch_1' set up to track 'origin/branch_1'.

C:\Users\Иван\Desktop\lab3_3>
```

Рисунок 21 – Отправка изменений на сервер

20. Создать средствами GitHub удаленную ветку branch_3.

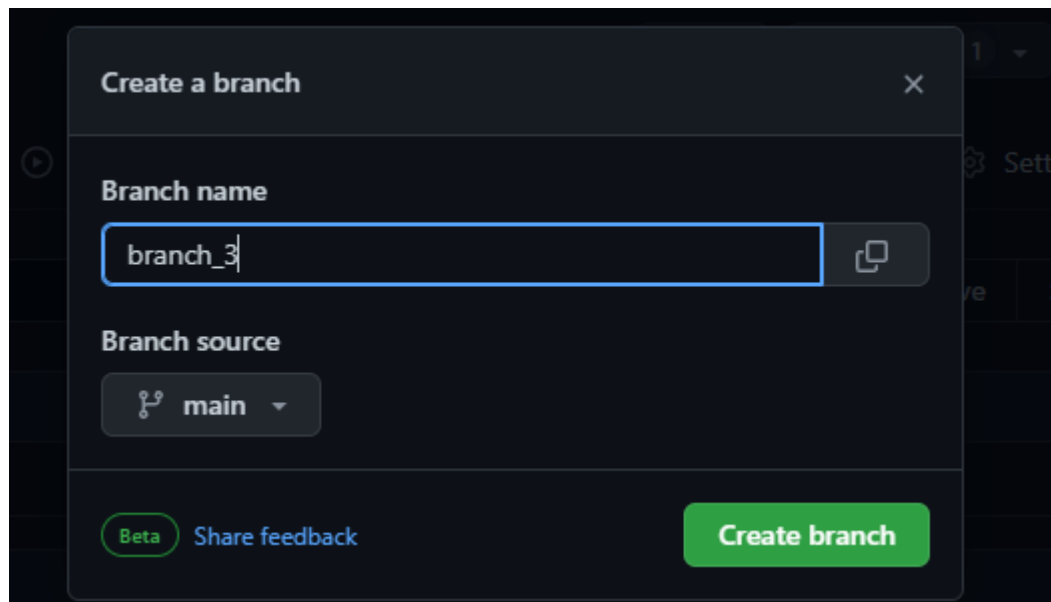


Рисунок 22 – Создание ветки на удаленном репозитории

21. Создать в локальном репозитории ветку отслеживания удаленной ветки branch_3.

```
C:\Users\UESR\lab3>git fetch --all
From https://github.com/MaxDrill/lab3
* [new branch]      branch_3    -> origin/branch_3

C:\Users\UESR\lab3>git checkout --track origin/branch_3
Switched to a new branch 'branch_3'
branch 'branch_3' set up to track 'origin/branch_3'.

C:\Users\UESR\lab3>
```

Рисунок 23 – Создание ветки отслеживания branch_3

22. Перейти на ветку branch_3 и добавить файл 2.txt строку "the final fantasy in the 4.txt file".

```
C:\Users\Иван\Desktop\lab3_3>git commit -m "add 2.txt in br_3"
[branch_3 58f8011] add 2.txt in br_3
1 file changed, 1 insertion(+)
create mode 100644 2.txt

C:\Users\Иван\Desktop\lab3_3>
```

Рисунок 24 – Переход на ветку branch_3 и изменение файла

23. Выполнить перемещение ветки master на ветку branch_2.

```
C:\Users\Иван\Desktop\lab3_3>git checkout branch_2
Switched to branch 'branch_2'

C:\Users\Иван\Desktop\lab3_3>git rebase main
Current branch branch_2 is up to date.

C:\Users\Иван\Desktop\lab3_3>git checkout main
Switched to branch 'main'
Your branch and 'origin/main' have diverged,
and have 4 and 1 different commits each, respectively.
(use "git pull" to merge the remote branch into yours)

C:\Users\Иван\Desktop\lab3_3>git merge branch_2
Updating 3812ca9..09204a7
Fast-forward
 1.txt | 2 +-
 3.txt | 1 +
 2 files changed, 2 insertions(+), 1 deletion(-)

C:\Users\Иван\Desktop\lab3_3>
```

Рисунок 25 – Перемещение и слияние веток master и branch_2

24. Отправить изменения веток master и branch_2 на GitHub.

```
C:\Users\Иван\Desktop\lab3_3>git push origin branch_2
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 363 bytes | 363.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'branch_2' on GitHub by visiting:
remote:   https://github.com/IvanLogvinov11/lab3_3/pull/new/branch_2
remote:
To https://github.com/IvanLogvinov11/lab3_3.git
 * [new branch]      branch_2 -> branch_2

C:\Users\Иван\Desktop\lab3_3>
```

Рисунок 26 – Отправка изменений на GitHub.

Вопросы для защиты работы

1. Что такое ветка?

Ветка в Git — это просто легковесный подвижный указатель на один из коммитов.

2. Что такое HEAD?

HEAD – это указатель, задача которого ссылаться на определенный коммит в репозитории.

3. Способы создания веток.

С помощью команды `git branch` или `git checkout -b`

4. Как узнать текущую ветку?

С помощью команды `git branch`, напротив будет знак «*»

5. Как переключаться между ветками?

С помощью команды `git checkout <имя ветки>`

6. Что такое удаленная ветка?

Удалённые ветки — это ссылки на состояние веток в удаленных репозиториях.

7. Что такое ветка отслеживания?

Ветка отслеживания — это ссылка, расположенная локально, на определённое состояние удалённых веток.

8. Как создать ветку отслеживания?

Для синхронизации `git fetch origin`, а затем `git checkout --track origin/<название_ветки>`.

9. Как отправить изменения из локальной ветки в удаленную ветку?

С помощью команды `git push origin <branch>`

10. В чем отличие команд `git fetch` и `git pull`?

`Git pull` – это сочетание команд `git fetch` (получение изменений с удаленного репозитория) и `git merge` (объединение веток).

11. Как удалить локальную и удаленную ветки?

Удаление удаленной ветки: `git push origin --delete <branch>`

Удаление локальной: `git branch -d <branch>`

12. Изучить модель ветвления `git-flow` (использовать материалы статей <https://www.atlassian.com/ru/git/tutorials/comparing-workflows/gitflow-workflow>, <https://habr.com/ru/post/106912/>). Какие основные типы веток присутствуют в модели `git-flow`? Как организована работа с ветками в модели `git-flow`? В чем недостатки `git-flow`?

`Git-flow` — альтернативная модель ветвления `Git`, в которой используются функциональные ветки и несколько основных веток.

Под каждую новую функцию нужно выделить собственную ветку, которую можно отправить в центральный репозиторий для создания резервной копии или совместной работы команды. Ветки `feature` создаются не на основе `main`, а на основе `develop`. Когда работа над функцией завершается, соответствующая ветка сливается с веткой `develop`. Функции не следует отправлять напрямую в ветку `main`.

Последовательность действий при работе по модели `Gitflow`:

1. Из ветки `main` создается ветка `develop`.
2. Из ветки `develop` создается ветка `release`.
3. Из ветки `develop` создаются ветки `feature`.

4. Когда работа над веткой `feature` завершается, она сливается в ветку `develop`.
5. Когда работа над веткой `release` завершается, она сливается с ветками `develop` и `main`.
6. Если в ветке `main` обнаруживается проблема, из `main` создается ветка `hotfix`.
7. Когда работа над веткой `hotfix` завершается, она сливается с ветками `develop` и `main`.

Первая проблема: авторам приходится использовать ветку `develop` вместо `master`, поскольку `master` зарезервирован для кода, который отправляется в продакшен. Существует сложившийся обычай называть рабочую ветвь по умолчанию `master`, и делать ответвления и слияния с ней. Большинство инструментов по умолчанию используют это название для основной ветки и по умолчанию выводят именно ее, и бывает неудобно постоянно переключаться вручную на другую ветку.

Вторая проблема процесса `git flow` – сложности, возникающие из-за веток для патчей и для релиза. Подобная структура может подойти некоторым организациям, но для абсолютного большинства она просто убийственно излишня. На сегодняшний день большинство компаний практикуют непрерывное развертывание (`continuous delivery`), что подразумевает, что основная ветвь по умолчанию может быть задеплоена (`deploy`). А значит, можно избежать использования веток для релиза и патчей, и всех связанных с ними хлопот, например, обратного слияния из веток релизов.