

**..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  
Коныжев Максим Викторович « » 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.

**Create a new repository**

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

---

**Owner \*** **Repository name \***

MaxDrill / lab3 ✓

Great repository names are short and memorable. Need inspiration? How about [refactored-bassoon?](#)

**Description (optional)**

---

☒ **Public**  
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**  
You choose who can see and commit to this repository.

---

**Initialize this repository with:**  
Skip this step if you're importing an existing repository.

☒ **Add a README file**  
This is where you can write a long description for your project. [Learn more.](#)

**Add .gitignore**  
Choose which files not to track from a list of templates. [Learn more.](#)

**.gitignore template:** Python ▾

**Choose a license**  
A license tells others what they can and can't do with your code. [Learn more.](#)

**License:** MIT License ▾

This will set **main** as the default branch. Change the default name in your [settings](#).

---

You are creating a public repository in your personal account.

---

**Create repository**

Рисунок 1 – Создание общедоступного репозитория

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

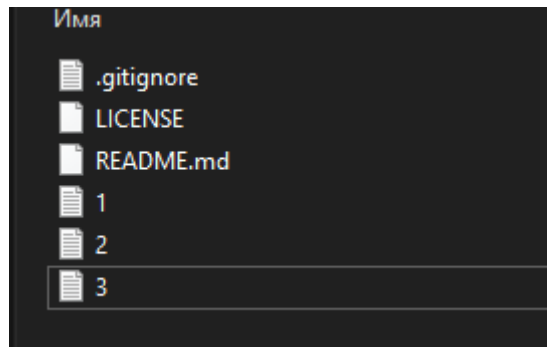


Рисунок 2 – Создание файлов

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

```
C:\Users\UESR\lab3>git add 1.txt

C:\Users\UESR\lab3>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

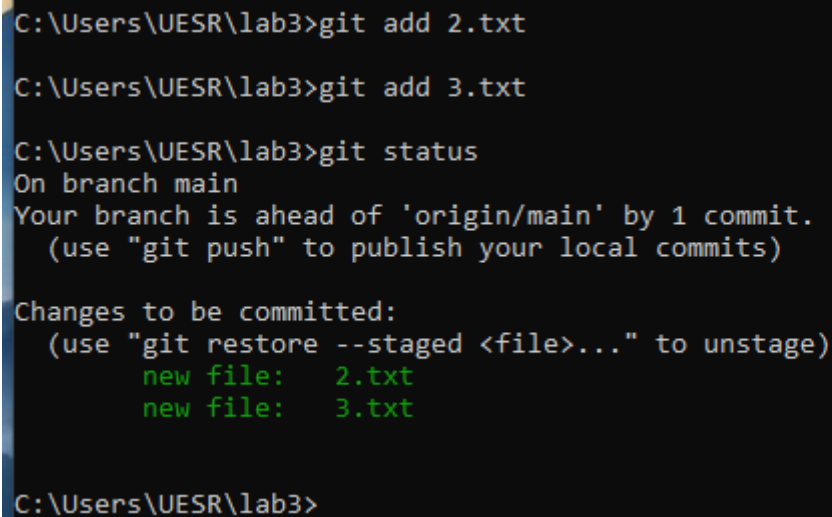
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        2.txt
        3.txt

C:\Users\UESR\lab3>git commit -m "add 1.txt"
[main 43711ce] add 1.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 1.txt

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

Рисунок 3 – Индексация первого файла и создание коммита

5. Проиндексировать второй и третий файлы.



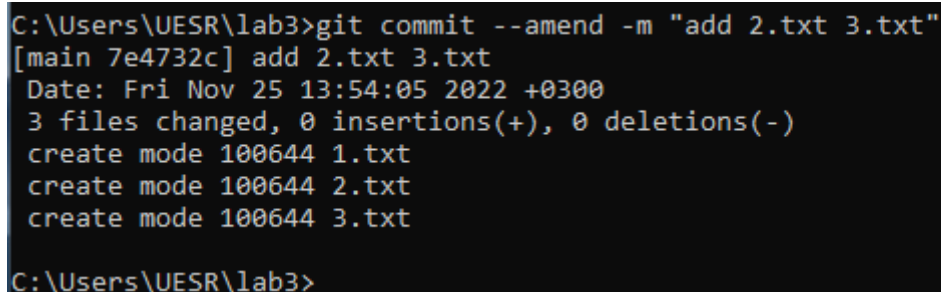
```
C:\Users\UESR\lab3>git add 2.txt
C:\Users\UESR\lab3>git add 3.txt
C:\Users\UESR\lab3>git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

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

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

Рисунок 4 – Добавление файлов 2 и 3 в индекс

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

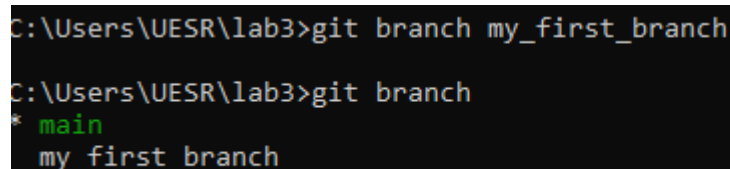


```
C:\Users\UESR\lab3>git commit --amend -m "add 2.txt 3.txt"
[main 7e4732c] add 2.txt 3.txt
Date: Fri Nov 25 13:54:05 2022 +0300
3 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 1.txt
create mode 100644 2.txt
create mode 100644 3.txt

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

Рисунок 5 – Перезапись коммита

7. Создать новую ветку my\_first\_branch.



```
C:\Users\UESR\lab3>git branch my_first_branch

C:\Users\UESR\lab3>git branch
* main
  my_first_branch
```

Рисунок 6 – Создание новой ветки

8. Перейти на ветку и создать новый файл in\_branch.txt, закоммитить изменения.

```
C:\Users\UESR\lab3>git checkout my_first_branch
Switched to branch 'my_first_branch'

C:\Users\UESR\lab3>git branch
  main
* my_first_branch

C:\Users\UESR\lab3>git add .

C:\Users\UESR\lab3>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\UESR\lab3>git commit -m "add in_branch.txt"
[my_first_branch 76e9961] add in_branch.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 in_branch.txt

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

Рисунок 7 – Создание коммита в новой ветке

9. Вернуться на ветку master.

```
C:\Users\UESR\lab3>git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

C:\Users\UESR\lab3>git branch
* main
  my_first_branch

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

Рисунок 10 – Возвращение на ветку «main»

10. Создать и сразу перейти на ветку new\_branch.

```

C:\Users\UESR\lab3>git checkout -b new_branch
Switched to a new branch 'new_branch'

C:\Users\UESR\lab3>git branch
  main
  my_first_branch
* new_branch

C:\Users\UESR\lab3>

```

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\UESR\lab3>git add .

C:\Users\UESR\lab3>git commit -m "add inform in 1.txt"
[new_branch 5910bdb] add inform in 1.txt
1 file changed, 1 insertion(+)

C:\Users\UESR\lab3>

```

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\UESR\lab3>git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

C:\Users\UESR\lab3>git merge my_first_branch
Updating 7e4732c..76e9961
Fast-forward
 in_branch.txt | 0
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 in_branch.txt

C:\Users\UESR\lab3>git merge new_branch
Merge made by the 'ort' strategy.
1.txt | 1 +
1 file changed, 1 insertion(+)

C:\Users\UESR\lab3>

```

Рисунок 13 – Слияние веток main, my\_first\_branch и new\_branch

13. Удалить ветки my\_first\_branch и new\_branch.

```

C:\Users\UESR\lab3>git branch -d my_first_branch
Deleted branch my_first_branch (was 76e9961).

C:\Users\UESR\lab3>git branch -d new_branch
Deleted branch new_branch (was 5910bdb).

C:\Users\UESR\lab3>git branch
* main

C:\Users\UESR\lab3>

```

Рисунок  
14 –  
Удаление  
веток

14. Создать ветки branch\_1 и branch\_2.

```

C:\Users\UESR\lab3>git branch branch_1

C:\Users\UESR\lab3>git branch branch_2

C:\Users\UESR\lab3>git branch
  branch_1
  branch_2
* main

C:\Users\UESR\lab3>

```

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\UESR\lab3>git checkout branch_1
Switched to branch 'branch_1'

C:\Users\UESR\lab3>git branch
* branch_1
  branch_2
  main

C:\Users\UESR\lab3>git add .

C:\Users\UESR\lab3>git add .

C:\Users\UESR\lab3>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\UESR\lab3>git commit -m "modified 1.txt 3.txt"
[branch_1 a654387] modified 1.txt 3.txt
 2 files changed, 2 insertions(+), 1 deletion(-)

C:\Users\UESR\lab3>

```

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\UESR\lab3>git checkout branch_2
Switched to branch 'branch_2'

C:\Users\UESR\lab3>git branch
  branch_1
* branch_2
  main

C:\Users\UESR\lab3>git add .

C:\Users\UESR\lab3>git status
On branch branch_2
nothing to commit, working tree clean

C:\Users\UESR\lab3>git add .

C:\Users\UESR\lab3>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\UESR\lab3>git commit -m "modified in br_2 1.txt 3.txt"
[branch_2 6ea871b] modified in br_2 1.txt 3.txt
 2 files changed, 2 insertions(+), 1 deletion(-)

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

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\UESR\lab3>git checkout branch_1
Switched to branch 'branch_1'

C:\Users\UESR\lab3>git branch
* branch_1
  branch_2
  main

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

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\UESR\lab3>git add 1.txt

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

Рисунок 19 – Решение конфликта вручную

```

C:\Users\UESR\lab3>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): vimdiff
4 files to edit

```

Рисунок 19 – Решение конфликта с помощью mergetool

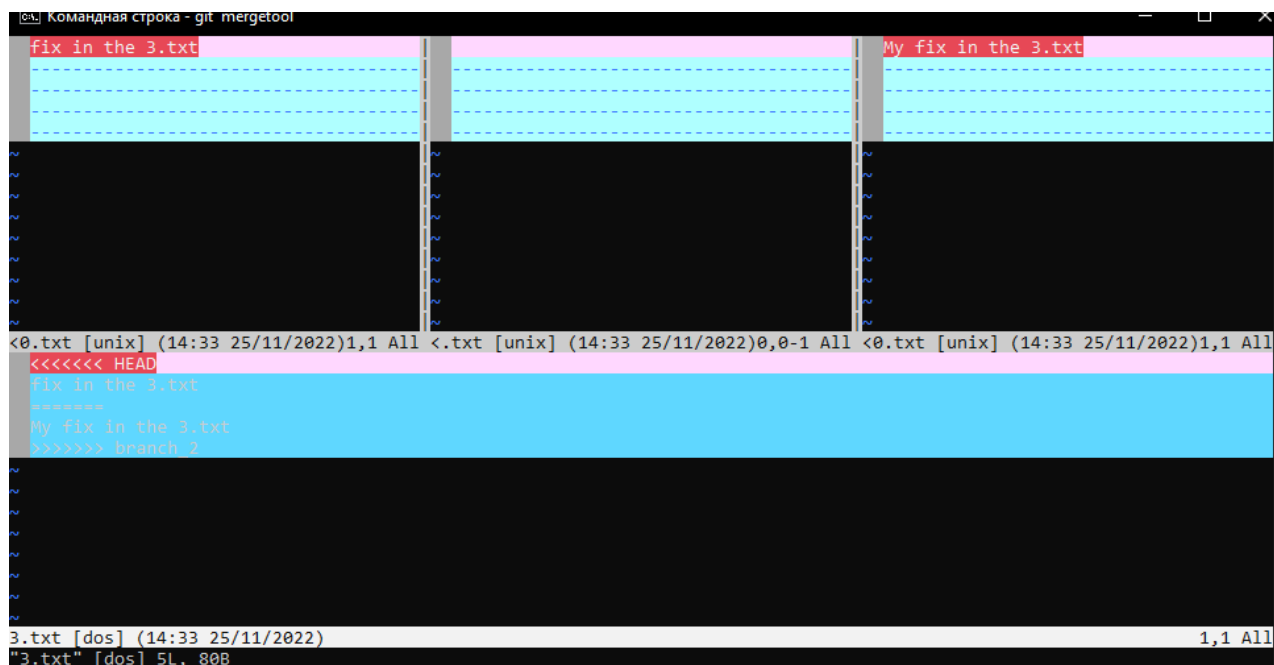


Рисунок 20 – Меню vimdiff

19. Отправить ветку branch\_1 на GitHub.

```
C:\Users\UESR\lab3>git push --set-upstream origin branch_1
Enumerating objects: 27, done.
Counting objects: 100% (27/27), done.
Delta compression using up to 4 threads
Compressing objects: 100% (19/19), done.
Writing objects: 100% (26/26), 2.51 KiB | 428.00 KiB/s, done.
Total 26 (delta 10), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (10/10), done.
remote:
remote: Create a pull request for 'branch_1' on GitHub by visiting:
remote:   https://github.com/MaxDrill/lab3/pull/new/branch_1
remote:
To https://github.com/MaxDrill/lab3.git
 * [new branch]      branch_1 -> branch_1
branch 'branch_1' set up to track 'origin/branch_1'.
C:\Users\UESR\lab3>
```

Рисунок 21 – Отправка изменений на сервер

20. Создать средствами GitHub удаленную ветку branch\_3.

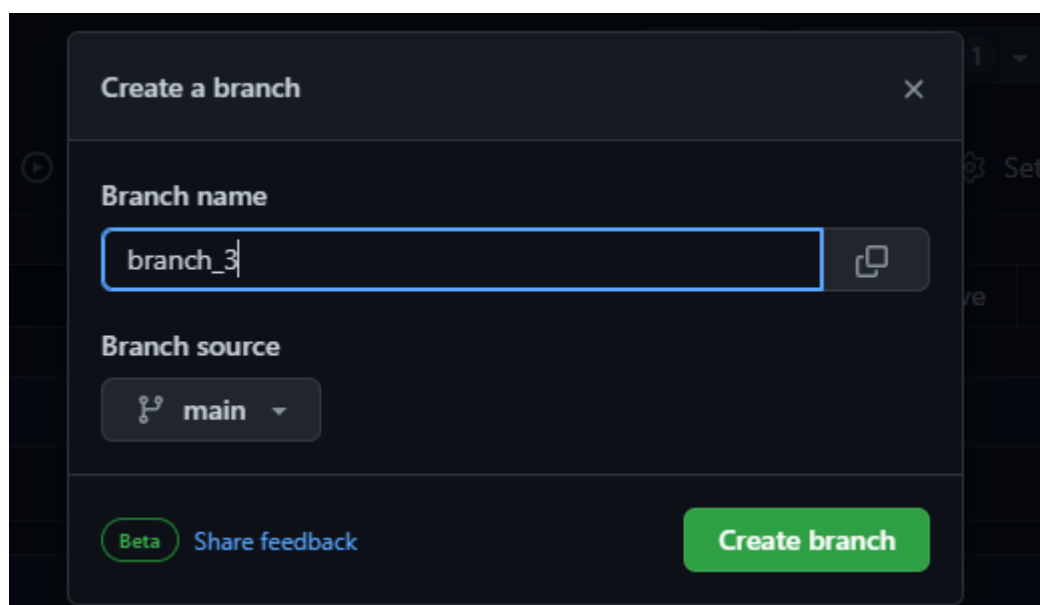


Рисунок 22 – Создание ветки на удаленном репозитории

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\UESR\lab3>git add .

C:\Users\UESR\lab3>git commit -m "add 2.txt in br_3"
[branch_3 8a81bed] add 2.txt in br_3
1 file changed, 1 insertion(+)
create mode 100644 2.txt
```

Рисунок 24 – Переход на ветку branch\_3 и изменение файла

23. Выполнить перемещение ветки master на ветку branch\_2.

```
C:\Users\UESR\lab3>git checkout branch_2
Switched to branch 'branch_2'

C:\Users\UESR\lab3>git rebase main
Current branch branch_2 is up to date.

C:\Users\UESR\lab3>git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 4 commits.
  (use "git push" to publish your local commits)

C:\Users\UESR\lab3>git merge branch_2
Updating 616dee7..6ea871b
Fast-forward
 1.txt | 2 +-
 3.txt | 1 +
 2 files changed, 2 insertions(+), 1 deletion(-)

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

Рисунок 25 – Перемещение и слияние веток master и branch\_2

24. Отправить изменения веток master и branch\_2 на GitHub.

```
C:\Users\UESR\lab3>git push origin main
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/MaxDrill/lab3.git
    1bf595d..6ea871b  main -> main

C:\Users\UESR\lab3>git push origin branch_2
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'branch_2' on GitHub by visiting:
remote:   https://github.com/MaxDrill/lab3/pull/new/branch_2
remote:
To https://github.com/MaxDrill/lab3.git
 * [new branch]      branch_2 -> branch_2
C:\Users\UESR\lab3>
```

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.