

# Лабораторная работа №1

## Управление версиями

Ишанова А.И. группа НФИ-02-19

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>3</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>4</b>
2.1	Настройка git . . . . .	4
2.2	Подключение репозитория к github . . . . .	6
2.3	Первичная конфигурация . . . . .	7
2.4	Конфигурация git-flow . . . . .	9
<b>3</b>	<b>Выводы</b>	<b>13</b>
<b>4</b>	<b>Контрольные вопросы</b>	<b>14</b>

# List of Figures

2.1	создание каталогов лабораторных работ . . . . .	4
2.2	конфигурация и генерация ssh-ключа . . . . .	5
2.3	копирование ssh-ключа . . . . .	5
2.4	добавление ssh-ключа на github.com . . . . .	6
2.5	репозитория на GitHub . . . . .	6
2.6	инициализация системы git . . . . .	7
2.7	создание заготовки README.md . . . . .	7
2.8	первый коммит . . . . .	7
2.9	добавление файла лицензии . . . . .	8
2.10	список игнорируемых файлов . . . . .	8
2.11	скачивание шаблона игнорируемых файлов . . . . .	8
2.12	добавление новых файлов . . . . .	8
2.13	коммит и выкладка на github . . . . .	9
2.14	инициализация git-flow . . . . .	9
2.15	установка префикса и проверка ветки, на которой нахо- димся . . . . .	10
2.16	создание и работа с релизом . . . . .	10
2.17	заливаем релизную ветку в основную . . . . .	11
2.18	отправка данных на github . . . . .	11
2.19	отображение релиза на github . . . . .	12
4.1	схема работы VCS . . . . .	15

# **Глава 1**

## **Цель работы**

Изучить идеологию и применение средств контроля версий.

## Глава 2

# Выполнение лабораторной работы

### 2.1 Настройка git

1. Создала учетную запись на <http://github.com>.
2. Создала структуру каталога лабораторных работ (команды `mkdir`, `cd`). (рис.2.1)

```
iMac-Alina:~ alinaishanova$ mkdir work
iMac-Alina:~ alinaishanova$ cd work
iMac-Alina:work alinaishanova$ mkdir 2020-2021
iMac-Alina:work alinaishanova$ cd 2020-2021
iMac-Alina:2020-2021 alinaishanova$ mkdir "Введение в научное программирование"
iMac-Alina:2020-2021 alinaishanova$ cd "Введение в научное программирование"
iMac-Alina:Введение в научное программирование alinaishanova$ mkdir laboratory
iMac-Alina:Введение в научное программирование alinaishanova$ cd laboratory
iMac-Alina:laboratory alinaishanova$ mkdir lab01
iMac-Alina:laboratory alinaishanova$ cd
iMac-Alina:~ alinaishanova$ cd
iMac-Alina:~ alinaishanova$ cd work
```

Figure 2.1: создание каталогов лабораторных работ

3. Настроила систему контроля версий git — проводим конфигурацию (команды `git config -global user.name` и `user.email`, `git config -global core.quotepath false`), создаем ssh-ключ (`ssh-keygen -C`) (рис.2.2.), копируем (`cat`) (рис.2.3.) и добавляем его на `github.com`(рис.2.4.).

```
iMac-Alina:work alinaishanova$ git config --global user.name "Ishanova Alina"
iMac-Alina:work alinaishanova$ git config --global user.email "WaterDragon99@yandex.ru"
iMac-Alina:work alinaishanova$ git config --global quotepath false
error: key does not contain a section: quotepath
iMac-Alina:work alinaishanova$ git config --global core.quotepath false
iMac-Alina:work alinaishanova$ ssh-keygen -C "Ishanova Alina <WaterDragon99@yandex.ru>"
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/alinaishanova/.ssh/id_rsa): /Users/alinaishanova/.ssh/id_rsa
/Users/alinaishanova/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /Users/alinaishanova/.ssh/id_rsa.
Your public key has been saved in /Users/alinaishanova/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:19oRX6YFgHHZ8AYEKy86G0U+UTXvf+tfIRqz0bko/p4 Ishanova Alina <WaterDragon99@yandex.ru>
The key's randomart image is:
+--[RSA 3072]-----+
|
|  +BX
|  . +o X
|  o . . +
|  . o . . o . . .
|  + o . . S + + + .
|  . = + B B o .
|  o o + . . . .
|  o . . . . .
|  . . . o E o . .
+-----[SHA256]-----+
```

Figure 2.2: конфигурация и генерация ssh-ключа

```
iMac-Alina:work alinaishanova$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQgC4kx9DRzPMxEnrsYdpPY7GX/g0SYT00t0TFwZJc3AC346r18qd9EECwW9tyeS1xTCYgybufrqRyqdr79tPM11bVbU04673RfkIS9E
+5s76op2bTadz+FK36+QIR948Ftwr1/00qzK198PPMS6SaaqBTIRyG7tCkyI3x3qVjRd/ID1LN/3cxCqIq6tLer0qMaK9u5yAz8LS8MhSneUgPRIzw6wF5Se5q5TCYPMwAzn5jy3KZuLwG
8Kx3q86LKTJ2oqg/KXu(p0Ex51w61wz2yT0j91wyk7/a5xH2akPRAjSc0n4G14816s2d5q8KzN2kq3KGMjrg3YU85xFO1YafuncYLBXE+181ajv6cuW8J06cuVW8RfuF189Q0C1ARuA66rjB
k1zuPRT+CE1hgBwJc00Hx3/MecG09haq6S0Rr+o15d3AUnshqAsmK0Wb7peb0EzY5AGrpvCyB24PM6Gey5RyWgSn02Sc= Ishanova Alina <WaterDragon99@yandex.ru>
```

Figure 2.3: копирование ssh-ключа

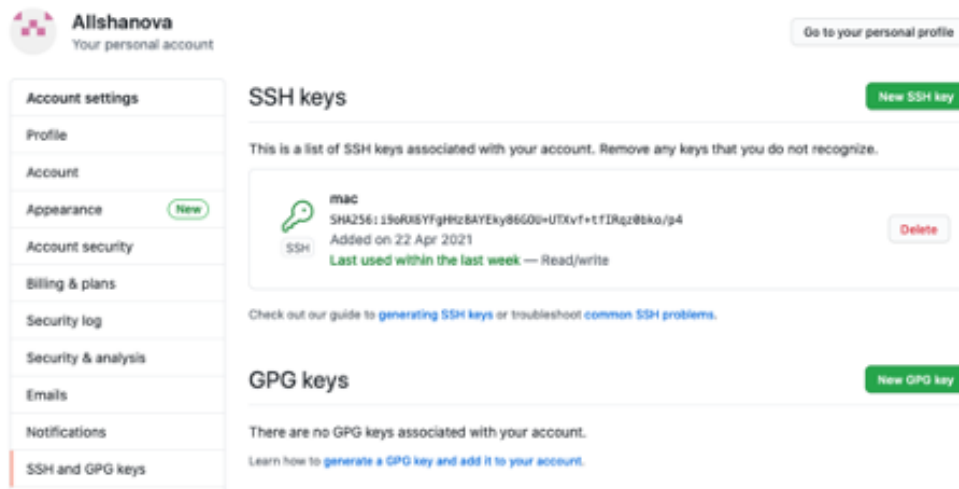


Figure 2.4: добавление ssh-ключа на github.com

## 2.2 Подключение репозитория к github

1. Создала репозиторий на GitHub. (рис.2.5.)

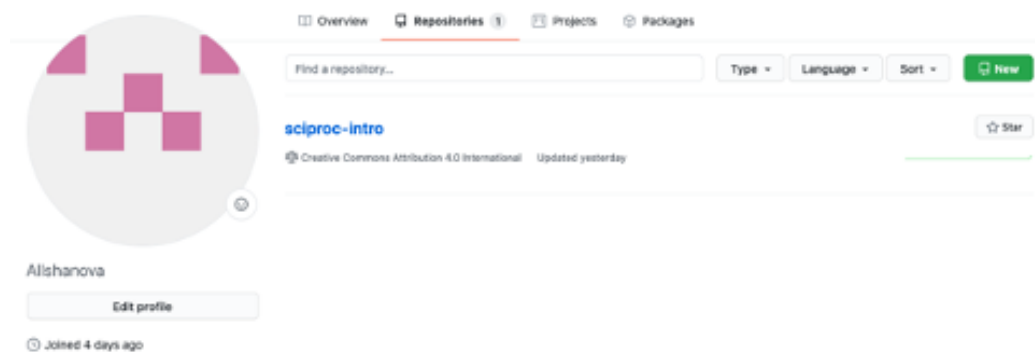


Figure 2.5: репозитория на GitHub

2. Перешла в каталог laboratory (команда cd) и инициализировала системы git (git init). (рис.2.6.)

```
iMac-Alina:work alinaishanova$ cd 2020-2021/"Введение в научное программирование"/laboratory
iMac-Alina:laboratory alinaishanova$ git init
Initialized empty Git repository in /Users/alinaishanova/work/2020-2021/Введение в научное программирование/laboratory/.git/
```

Figure 2.6: инициализация системы git

3. Создала заготовку для файла README.md. (команды echo и add)  
(рис.2.7)

```
iMac-Alina:laboratory alinaishanova$ echo "# Лабораторные работы" >> README.md
iMac-Alina:laboratory alinaishanova$ git add README.md
```

Figure 2.7: создание заготовки README.md

4. Сделала первый коммит (git commit -m) и выложила на github  
(git remote add origin и git push -u origin master). (рис.2.8.)

```
iMac-Alina:laboratory alinaishanova$ git commit -m "first commit"
[[master (root-commit) db75330] first commit
 1 file changed, 1 insertion(+)
   create mode 100644 README.md
iMac-Alina:laboratory alinaishanova$ git remote add origin git@github.com:AlIshanova/sciproc-intro.git
iMac-Alina:laboratory alinaishanova$ git push -u origin master
[Enter passphrase for key '/Users/alinaishanova/.ssh/id_rsa':
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 256 bytes | 256.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To github.com:AlIshanova/sciproc-intro.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

Figure 2.8: первый коммит

## 2.3 Первичная конфигурация

1. С помощью команды wget довила файл лицензии. (рис.2.9.)



```

iMac-Alina:laboratory alinaishanova$ wget https://creativecommons.org/licenses/by/4.0/legalcode.txt -O LICENSE
--2021-04-22 21:33:25-- https://creativecommons.org/licenses/by/4.0/legalcode.txt
Распознаётся creativecommons.org (creativecommons.org) - 104.20.155.16, 104.20.150.16, 172.67.34.140
Подключение к creativecommons.org (creativecommons.org)[104.20.155.16]:443... соединение установлено.
HTTP-запрос отправлен. Ожидание ответа... 200 OK
Длина: нет данных [text/plain]
Сохранение в: «LICENSE»

LICENSE [ «» ] 18,22K --,~KB/s за 0,002s
2021-04-22 21:33:25 (0,32 MB/s) - «LICENSE» сохранён [18657]

```

Figure 2.9: добавление файла лицензии

- Посмотрела список имеющихся шаблонов и скачала шаблон игнорируемых файлов для C (команда `curl -L -s`). (рис.2.10. и рис.2.11.)

```

iMac-Alina:laboratory alinaishanova$ curl -L -s https://www.gitignore.io/api/list
ic,ic-bitrix,a-frame,actionsript,ada
adobe,advancedinstaller,adventuregamestudio,agda,al
alterquartusii,altium,amplify,android,androidstudio
angular,anjuta,ansible,apachecordova,apachehadoop
appbuilder,appceleratoritanium,appcode,appcode=all,appcode=ini
appengine,apimastudio,arc4nist,archive,archives
archlinuxpackages,aspenicore,assembler,ate,almelstudio
ats,audio,automationstudio,autotools,autotools=strict
aur,azurefunctions,backup,ballerina,baseres
basic,batch,bazaar,bezel,bitrise
bitrix,bittorrent,blackbox,bloop,bluej
bookdown,bower,briccc,buck,c
c++,cake,cakephp,cakephp2,cakephp3
calabash,carthage,certificates,ceylon,cfwheels
chefcookbook,chocolatey,clean,clion,clion=all
clion=ini,clajura,claud9,cmake,cocapods
cocos2dx,cocoscreator,code,code-java,codeblocks
codecomposerstudio,codeigniter,codeis,codekit,codeiniffer
coffeescript,commonlisp,compdoc,composer,compressed
compressedarchive,compression,conan,concrete5,cq
cordova,craftcms,crashlytics,crbasic,crossbar
crystal,cs-cart,csharp,cuda,cvs
cypressio,d,dart,darteditor,data
database,datarecovery,dbviewer,defold,delphi
efframe,diff,direnv,diskimage,django
dm,docfx,docpress,dccz,dotenv
dotfilesash,dotnetcore,dotnetsettings,dreamweaver,dropbox
drupal,drupal7,drupal8,e2studio,eagle
easybook,eclipse,eiffelstudio,elasticsearch,elisp
elivite,elm,emacs,ember,enzyme

```

Figure 2.10: список игнорируемых файлов

```

yii,yii2,zendframework,zephir,zig
zsh,zukencr800iMac-Alina:laboratory alinaishanova$ curl -L -s https://www.gitignore.io/api/c >> .gitignore

```

Figure 2.11: скачивание шаблона игнорируемых файлов

- Добавила новые файлы (`git add`), выполнила коммит (`git commit -a`) и отправила на github (`git push`). (рис.2.12. и рис 2.13.)

```

iMac-Alina:laboratory alinaishanova$ git add .

```

Figure 2.12: добавление новых файлов

```

iMac-Alina:laboratory alinaishanova$ git commit -a
[master 51d884e] LAB01
 2 files changed, 455 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 LICENSE
iMac-Alina:laboratory alinaishanova$ git push
Enter passphrase for key '/Users/alinaishanova/.ssh/id_rsa':
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 6.44 KiB | 6.44 MiB/s, done.
Total 4 (delta 0), reused 0 (delta 0)
To github.com:AlIshanova/sciproc-intro.git
 db75330..51d884e  master -> master
iMac-Alina:laboratory alinaishanova$ █

```

Figure 2.13: коммит и выкладка на github

## 2.4 Конфигурация git-flow

1. Инициализировала git-flow. (команда git init), проверила, что нахожусь на ветке develop (git branch). (рис.2.14. и рис.2.15.)

```

iMac-Alina:laboratory alinaishanova$ git init flow█

```

Figure 2.14: инициализация git-flow

```

Which branch should be used for bringing forth production releases?
- master
Branch name for production releases: [master] master
Branch name for "next release" development: [develop] develop

How to name your supporting branch prefixes?
Feature branches? [feature/] feature/
Release branches? [release/] release/
Hotfix branches? [hotfix/] hotfix/
Support branches? [support/] support/
Version tag prefix? [] v
iMac-Alina:laboratory alinaishanova$ git branch
* develop
  master

```

Figure 2.15: установка префикса и проверка ветки, на которой находимся

2. Создала релиз с версией 1.0.0 (git flow release start), записала версию (echo), добавила в индекс (git add и git commit -am). (рис.2.16.)

```

iMac-Alina:laboratory alinaishanova$ git flow release start 1.0.0
Switched to a new branch 'release/1.0.0'

Summary of actions:
- A new branch 'release/1.0.0' was created, based on 'develop'
- You are now on branch 'release/1.0.0'

Follow-up actions:
- Bump the version number now!
- Start committing last-minute fixes in preparing your release
- When done, run:

    git flow release finish '1.0.0'

iMac-Alina:laboratory alinaishanova$ echo "1.0.0" >> VERSION
iMac-Alina:laboratory alinaishanova$ git add .
iMac-Alina:laboratory alinaishanova$ git commit -am 'chore(main): add version'
[release/1.0.0 d709b64] chore(main): add version
1 file changed, 1 insertion(+)
create mode 100644 VERSION

```

Figure 2.16: создание и работа с релизом

3. Завершила релиз и слила его в основную ветку (git flow release finish). (рис.2.17.)

```
iMac-Alina:laboratory alinaishanova$ git flow release finish 1.0.0
Branches 'master' and 'origin/master' have diverged.
And local branch 'master' is ahead of 'origin/master'.
Switched to branch 'develop'
Merge made by the 'recursive' strategy.
VERSION | 1 +
1 file changed, 1 insertion(+)
create mode 100644 VERSION
Deleted branch release/1.0.0 (was d709b64).

Summary of actions:
- Latest objects have been fetched from 'origin'
- Release branch has been merged into 'master'
- The release was tagged 'v1.0.0'
- Release branch has been back-merged into 'develop'
- Release branch 'release/1.0.0' has been deleted

iMac-Alina:laboratory alinaishanova$
```

Figure 2.17: заливаем релизную ветку в основную

#### 4. Отправила данные на github (git push – all, –tags ).(рис.2.18.)

```
iMac-Alina:laboratory alinaishanova$ git push --all
Enter passphrase for key '/Users/alinaishanova/.ssh/id_rsa':
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 426 bytes | 426.00 KiB/s, done.
Total 5 (delta 3), reused 0 (delta 0)
remote: Resolving deltas: 100% (3/3), completed with 1 local object.
To github.com:Alinaishanova/sciproc-intro.git
51d884e..0d3da53 master -> master
* [new branch]      develop -> develop
iMac-Alina:laboratory alinaishanova$ git push --tags
Enter passphrase for key '/Users/alinaishanova/.ssh/id_rsa':
Enumerating objects: 1, done.
Counting objects: 100% (1/1), done.
Writing objects: 100% (1/1), 170 bytes | 170.00 KiB/s, done.
Total 1 (delta 0), reused 0 (delta 0)
To github.com:Alinaishanova/sciproc-intro.git
* [new tag]         v1.0.0 -> v1.0.0
```

Figure 2.18: отправка данных на github

#### 5. Релиз есть на GitHub. (рис.2.19.)

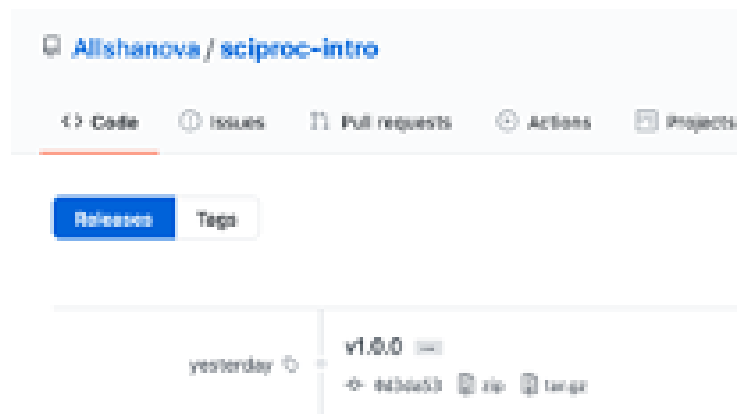


Figure 2.19: отображение релиза на github

## Глава 3

### Выводы

В ходе выполнения лабораторной работы я познакомилась с основными командами git, работой с локальным и удаленным репозиториями, завела аккаунт на github, настроила git, подключила к нему репозиторий, выполнила конфигурацию git и git-flow.

## Глава 4

### Контрольные вопросы

1. **Что такое VCS и для решения каких задач они предназначены?** Система контроля версий – программное обеспечение для облегчения работы с изменяющейся информацией. Они применяются при работе нескольких человек над одним проектом. При внесении изменений в содержание проекта система контроля версий позволяет их фиксировать, совмещать изменения, произведенные разными участниками проекта, производить откат к любой более ранней версии проекта, если это требуется.
2. **Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.** Хранилище – единый репозиторий для хранения файлов. Commit – сохранение добавленных изменений и всех измененных файлов. История – все изменения, все версии. (находится в хранилище) Рабочая копия – это то, с чем работают разработчики, куда вносят изменения, с помощью коммита изменения с рабочей копии отправляют-

ся в репозиторий, а с помощью команды update загружается последняя версия репозитория.

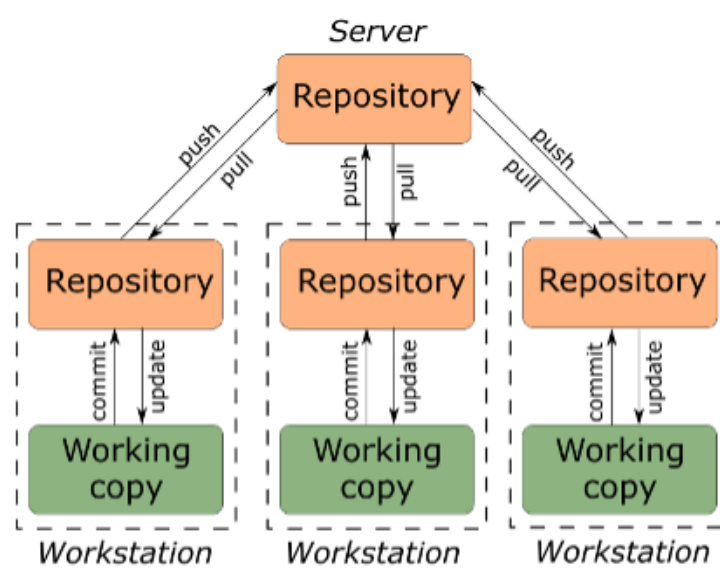


Figure 4.1: схема работы VCS

3. **Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида.**
- Централизованная: имеется единое хранилище документов, управляемое специальным сервером, который и выполняет большую часть функций по управлению версиями. Для каждого файла хранится информация о его предыдущих версиях на центральном сервере. Предназначены для бэкапирования, отслеживания и синхронизации файлов. (CVS, Subversion(SVN), Perforce)



- Децентрализованная: разработчики полностью копируют всю информацию о версиях файлов себе на компьютер. И, если откажет центральный сервер, любой разработчик может его восстановить. Предназначены для обмена изменениями, нет какой-то жестко заданной структуры репозитория с центральным сервером и у каждого есть свой полноценный репозиторий. (Git, Mercurial, Bazaar)

4. ***Опишите действия с VCS при единоличной работе с хранилищем.***

- Создание репозитория
- Залив данных на репозиторий
- Работа, коммиты и пуши

5. ***Опишите порядок работы с общим хранилищем VCS.***

- Подключение к созданному администратором репозиторию
- Скачивание данных из репозитория
- (возможно) создание различных веток
- Работа, коммиты и пуши
- Разрешение конфликтов (merge), слияние веток

6. ***Каковы основные задачи, решаемые инструментальным средством *git*?*** В персональных проектах, для которых не требуется центральный репозиторий, Git, используя различные ветки, применяется, главным образом, для отслеживания изменений и экспериментирования в вашем проекте с различными приемами, предоставляя возможность либо сливать изменения с проектом,

либо выполнять их откат. С помощью git создается рабочее дерево, осуществляется отправка всех произведенных изменений локального дерева в центральный репозиторий, просмотр текущих изменений, сохранение текущих изменений, сохранение ветки, переключение между ветками, слияние и удаление веток.

7. *Назовите и дайте краткую характеристику командам git.* – создание основного дерева репозитория: **git init** – получение обновлений (изменений) текущего дерева из центрального репозитория: **git pull** – отправка всех произведенных изменений локального дерева в центральный репозиторий: **git push** – просмотр списка измененных файлов в текущей директории: **git status** – просмотр текущих изменений: **git diff** – добавить все измененные и/или созданные файлы и/или каталоги: **git add .** – удалить файл и/или каталог из индекса репозитория **git rm имена\_файлов** – сохранить все добавленные изменения и все измененные файлы: **git commit -am 'Описание коммита'** – создание новой ветки, базирующейся на текущей: **git checkout -b имя\_ветки** – переключение на некоторую ветку: **git checkout имя\_ветки** – отправка изменений конкретной ветки в центральный репозиторий: **git push origin имя\_ветки** – слияние ветки с текущим деревом: **git merge -no-ff имя\_ветки** – удаление локальной уже слитой с основным деревом ветки: **git branch -d имя\_ветки** – принудительное удаление локальной ветки: **git branch -D имя\_ветки** – удаление ветки с центрального репозитория: **git push origin :имя\_ветки**

8. *Приведите примеры использования при работе с локальным и удаленными репозиториями.* Локальный и удаленные репозитории обмениваются данными через две команды: **git push** – отправляет данные с локального репозитория на удаленный **git pull** – сливает любые внесенные коммиты в ветку, в которой разработчик сейчас работает
9. *Что такое и зачем могут быть нужны ветви (branches)?* Ветвь в системах управления версиями — направление разработки, независимое от других. Ветвь представляет собой копию части хранилища (например, одного каталога), в которую можно вносить изменения, не влияющие на другие ветви. Документы в разных ветвях имеют одинаковую историю до точки ветвления и разные — после нее. Позволяет вносить изменения параллельно, изолировать внесенные изменения, не дестабилизируя базу. При этом можно слить ветки.
10. *Как и зачем можно игнорировать некоторые файлы при commit?* Если есть какие-то файлы, которые нельзя изменять, нужно игнорировать коммиты с ними. Для этого создается .gitignore для типов файлов, которые не нужно отслеживать.