

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЁТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2

дисциплина: Операционные системы

Студент: Манаева Варвара Евгеньевна

Группа: НФИбд-01-20

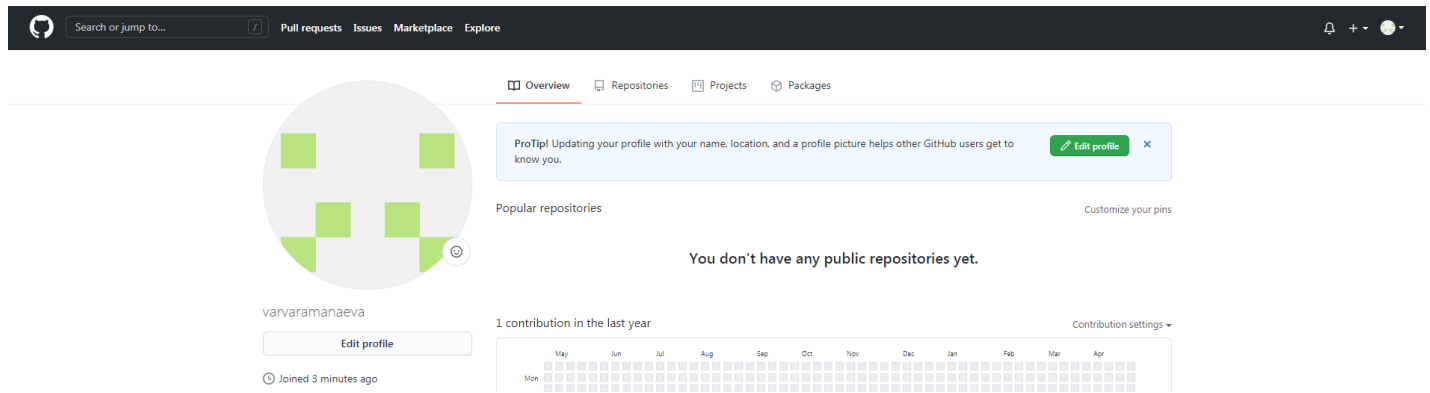
МОСКВА

2021 г.

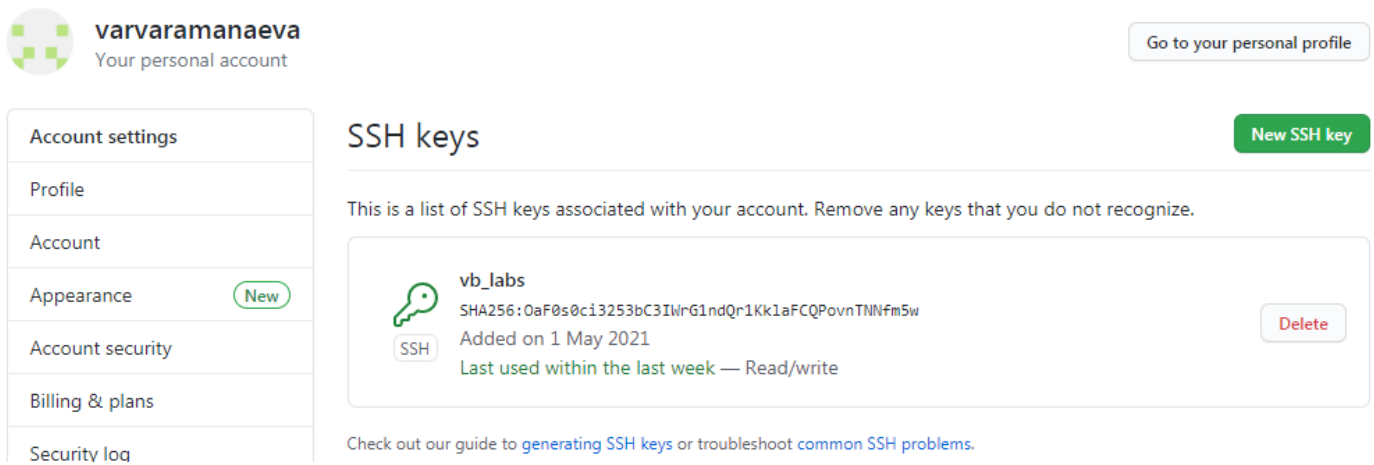
Цель: изучение идеологии и применения средств контроля версий.

Этапы работы:

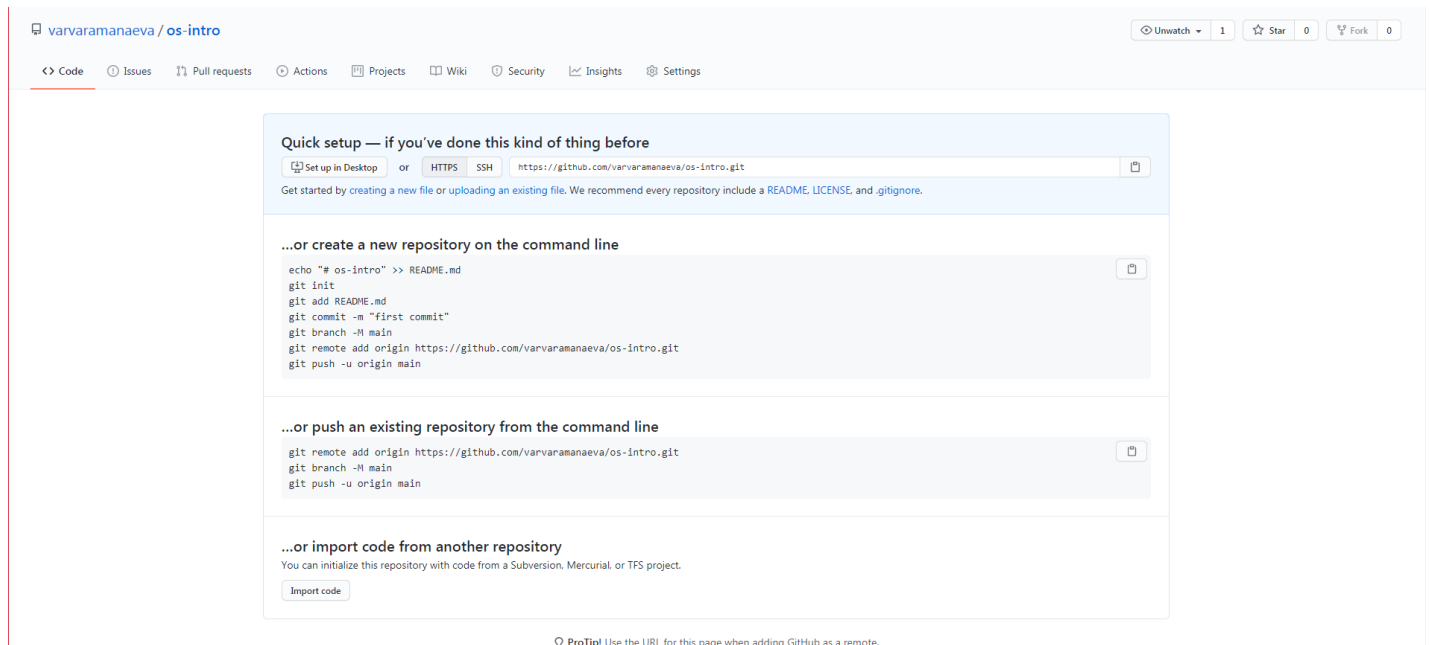
1) Была создана учётная запись на *github.com*.



2) Была настроена система контроля версий *git* для репозитория *github.com*.



3) Был создан репозиторий на *github.com* под названием *os-intro*.



4) Создан рабочий каталог.

```

[vemanaeva@vemanaeva ~]$ ls -la
.  .bash_history  .bash_profile  .cache  .dbus  .gitconfig  .local  .pki  .vboxclient-clipboard.pid  .vboxclient-draganddrop.pid  Видео  Загрузки  Музыка  Рабочий стол
.bash_logout  .bashrc  .config  .esd_auth  .ICEauthority  .mozilla  .ssh  .vboxclient-display-svga-x11.pid  .vboxclient-seamless.pid  Документы  Изображения  Общедоступные  Шаблоны
[vemanaeva@vemanaeva ~]$ mkdir laboratory
[vemanaeva@vemanaeva ~]$ cd laboratory
[vemanaeva@vemanaeva laboratory]$

```

5) Рабочий каталог был подключен к системе *git*. В нём был создан файл *README.md*, с записью «Лабораторные работы». Файл отправлен на *github.com*.

```

[vemanaeva@vemanaeva laboratory]$ git init
Initialized empty Git repository in /home/vemanaeva/laboratory/.git/
[vemanaeva@vemanaeva laboratory]$ echo "# Лабораторные работы" >> README.md
[vemanaeva@vemanaeva laboratory]$ git add README.md
[vemanaeva@vemanaeva laboratory]$ git commit -m "first commit"
[master (root-commit) af51c03] first commit
1 file changed, 1 insertion(+)
 create mode 100644 README.md
[vemanaeva@vemanaeva laboratory]$ git remote add origin git@github.com:varvaramanaeva/os-intro.git
[vemanaeva@vemanaeva laboratory]$ git push -u origin master
Counting objects: 3, done.
Writing objects: 100% (3/3), 249 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To git@github.com:varvaramanaeva/os-intro.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.

```

6) Был добавлен файл лицензии. Был добавлен шаблон игнорируемых файлов. Оба файла отправлены на *github.com*.

```

[vemanaeva@vemanaeva laboratory]$ wget https://creativecommons.org/licenses/by/4.0/legalcode.txt -O LICENSE
--2021-05-01 17:57:28-- https://creativecommons.org/licenses/by/4.0/legalcode.txt
Распознаётся creativecommons.org (creativecommons.org)... 172.67.34.140, 104.20.151.16, 104.20.150.16, ...
Подключение к creativecommons.org (creativecommons.org)[172.67.34.140]:443... соединение установлено.
HTTP-запрос отправлен. Ожидание ответа... 200 OK
Длина: нет данных [text/plain]
Сохранение в: «LICENSE»

[ <=>

2021-05-01 17:57:28 (6,97 MB/s) - «LICENSE» сохранён [18657]

[vemanaeva@vemanaeva laboratory]$ curl -L -s https://www.gitignore.io/api/c >> .gitignore
[vemanaeva@vemanaeva laboratory]$ git status
# On branch master
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       .gitignore
#       LICENSE
nothing added to commit but untracked files present (use "git add" to track)
[vemanaeva@vemanaeva laboratory]$ ls -la
.  ..  .git  .gitignore  LICENSE  README.md
[vemanaeva@vemanaeva laboratory]$ git add .
[vemanaeva@vemanaeva laboratory]$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       new file:   .gitignore
#       new file:   LICENSE
#
[vemanaeva@vemanaeva laboratory]$ git commit -a
[master 50e8945] Second commit )))
2 files changed, 455 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 LICENSE
[vemanaeva@vemanaeva laboratory]$ git push -u origin master
Counting objects: 5, done.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 6.44 KiB | 0 bytes/s, done.
Total 4 (delta 0), reused 0 (delta 0)
To git@github.com:varvaramanaeva/os-intro.git
   af51c03..50e8945  master -> master
Branch master set up to track remote branch master from origin.
[vemanaeva@vemanaeva laboratory]$

```

varvaramanaeva / os-intro

Unwatch

1

Star

0

Fork

0

<> Code

Issues

Pull requests

Actions

Projects

Wiki

Security

Insights

Settings

master

Commits on May 1, 2021

Second commit))))
varvaramanaeva committed 2 minutes ago

first commit
varvaramanaeva committed 7 minutes ago

Newer

Older

© 2021 GitHub, Inc.

Terms

Privacy

Security

Status

Docs

Contact GitHub

Pricing

API

Training

Blog

About

varvaramanaeva / os-intro

Unwatch

1

Star

0

Fork

0

<> Code

Issues

Pull requests

Actions

Projects

Wiki

Security

Insights

Settings

first commit

master

varvaramanaeva committed 7 minutes ago

0 parents commit af51c83df6e6d48d7cacb194fe148668bF8f946

Showing 1 changed file with 1 addition and 0 deletions.

Unified Split

1 README.md

... @@ -0,0 +1 @@

1 + # Лабораторные работы

Second commit))))

master

varvaramanaeva committed 3 minutes ago

1 parent af51c83 commit 50e8945b06988713065d132b32893adf54c19806

Showing 2 changed files with 455 additions and 0 deletions.

Unified Split

59 .gitignore

... @@ -0,0 +1,59 @@

1 +

2 + # Created by https://www.toptal.com/developers/gitignore/api/c

3 + # Edit at https://www.toptal.com/developers/gitignore/templates=c

4 +

5 + ## C ##

6 + # Prerequisites

7 + *.d

8 +

9 + # Object files

10 + *.o

11 + *.ko

12 + *.obj

13 + *.elf

14 +

15 + # Linker output

16 + *.ilk

17 + *.map

18 + *.exp

19 +

20 + # Precompiled Headers

21 + *.gch

22 + *.pch

23 +

24 + # Libraries

25 + *.lib

26 + *.a

27 + *.la

28 + *.lo

29 +

30 + # Shared objects (Inc. Windows DLLs)

31 + *.dll

32 + *.so

33 + *.so.*

34 + *.dylib

```

34 + *.dylib
35 +
36 + # Executables
37 + *.exe
38 + *.out
39 + *.app
40 + *.i*86
41 + *.x86_64
42 + *.hex
43 +
44 + # Debug files
45 + *.dSYM/
46 + *.su
47 + *.idb
48 + *.pdb
49 +
50 + # Kernel Module Compile Results
51 + *.mod*
52 + *.cmd
53 + .tmp_versions/
54 + modules.order
55 + Module.symvers
56 + Mkfile.old
57 + dkms.conf
58 +
59 + # End of https://www.toptal.com/developers/gitignore/api/c

```

▼ 396 ■■■■■ LICENSE 📄

```

... .. @@ -0,0 +1,396 @@
1 + Attribution 4.0 International
2 +
3 + =====
4 +
5 + Creative Commons Corporation ("Creative Commons") is not a law firm and
6 + does not provide legal services or legal advice. Distribution of
7 + Creative Commons public licenses does not create a lawyer-client or
8 + other relationship. Creative Commons makes its licenses and related
9 + information available on an "as-is" basis. Creative Commons gives no
10 + warranties regarding its licenses, any material licensed under their
11 + terms and conditions, or any related information. Creative Commons
12 + disclaims all liability for damages resulting from their use to the
13 + fullest extent possible.
14 +
15 + Using Creative Commons Public Licenses

```

...

```

376 + =====
377 +
378 + Creative Commons is not a party to its public
379 + licenses. Notwithstanding, Creative Commons may elect to apply one of
380 + its public licenses to material it publishes and in those instances
381 + will be considered the "Licensor." The text of the Creative Commons
382 + public licenses is dedicated to the public domain under the CC0 Public
383 + Domain Dedication. Except for the limited purpose of indicating that
384 + material is shared under a Creative Commons public license or as
385 + otherwise permitted by the Creative Commons policies published at
386 + creativecommons.org/policies, Creative Commons does not authorize the
387 + use of the trademark "Creative Commons" or any other trademark or logo
388 + of Creative Commons without its prior written consent including,
389 + without limitation, in connection with any unauthorized modifications
390 + to any of its public licenses or any other arrangements,
391 + understandings, or agreements concerning use of licensed material. For
392 + the avoidance of doubt, this paragraph does not form part of the
393 + public licenses.
394 +
395 + Creative Commons may be contacted at creativecommons.org.
396 +

```

7) Инициализирован *git-flow*. Префикс версий записан в *v*. Записана версия в файл версии. Релизная ветка залита в основную ветку. Данные отправлены на *github.com*.

```

[vemanaeva@vemanaeva laboratory]$ git flow init

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

How to name your supporting branch prefixes?
Feature branches? [feature/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? [] v
[vemanaeva@vemanaeva laboratory]$ git branch
* develop
  master
[vemanaeva@vemanaeva laboratory]$ 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'

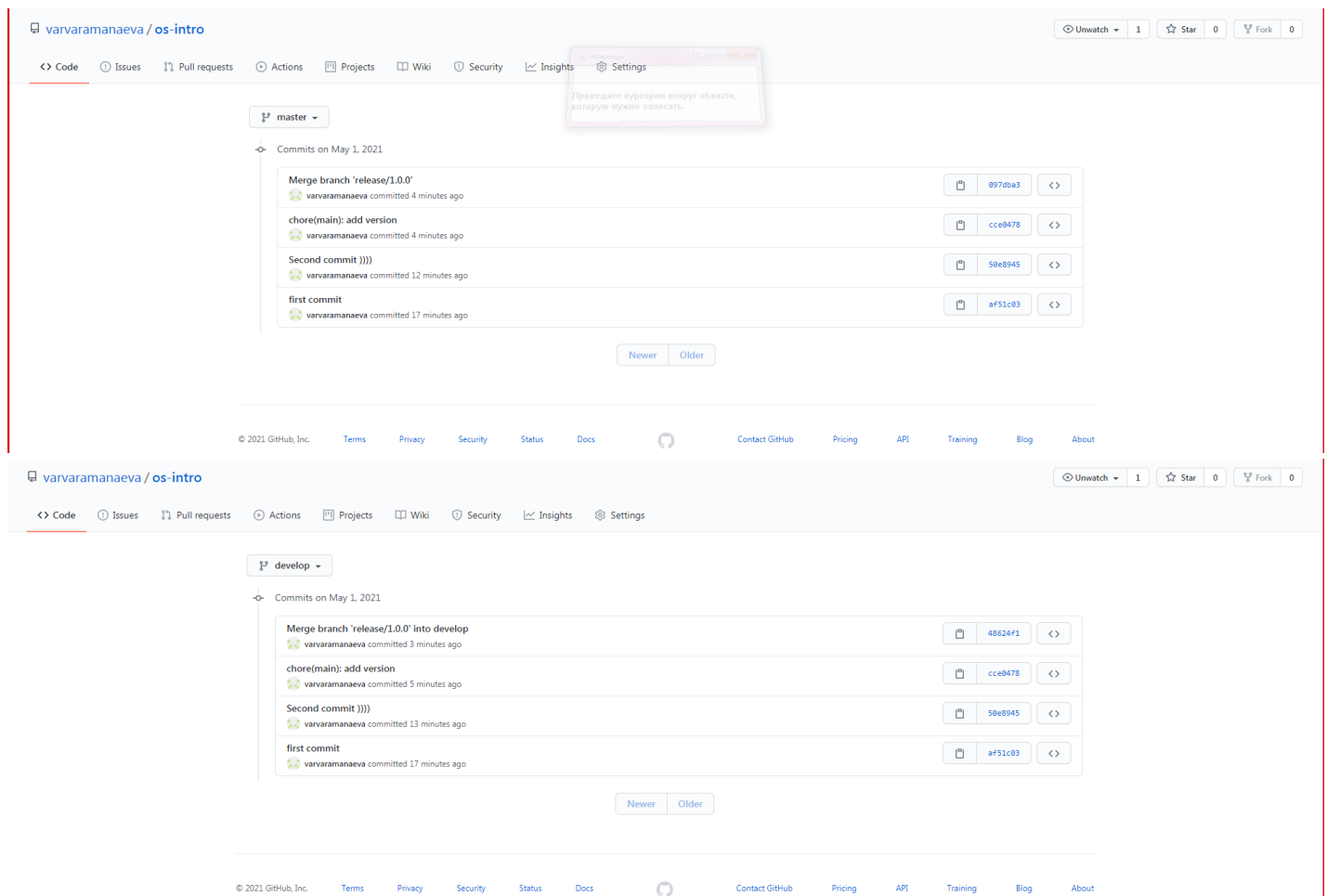
[vemanaeva@vemanaeva laboratory]$ echo "1.0.0" >> VERSION
[vemanaeva@vemanaeva laboratory]$ ls -a
.  ..  .git  .gitignore  LICENSE  README.md  VERSION
[vemanaeva@vemanaeva laboratory]$ git add .
[vemanaeva@vemanaeva laboratory]$ git commit -am 'chore(main): add version'
[release/1.0.0 cce0478] chore(main): add version
 1 file changed, 1 insertion(+)
 create mode 100644 VERSION

[vemanaeva@vemanaeva laboratory]$ 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 cce0478).

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

[vemanaeva@vemanaeva laboratory]$ █

```



Выводы: были изучены идеология и применения средств контроля версий.

Контрольные вопросы:

1) Что такое системы контроля версий (VCS) и для решения каких задач они предназначены?

Ответ: Система управления версиями (Version Control System, VCS) — программное обеспечение для облегчения работы с изменяющейся информацией. Системы контроля версий применяются при работе нескольких человек над одним проектом. Обычно основное дерево проекта хранится в локальном или удалённом репозитории, к которому настроен доступ для участников проекта. Сохраняется также и история изменений проекта, дабы в любой момент изменения, повлекшие за собой ошибку и/или признанные ненужными, могли откатить до нужного этапа развития проекта.

2) Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.

Ответ: Хранилище — основное место хранения всех данных, к которому имеют доступ все члены проекта. * Commit* — любое изменение к проекту/дополнение проекта.

История — последовательность commit-ов, то есть всех изменений/дополнений к проекту.

Рабочая копия — версия данных, которую пользователь выгружает для внесения изменений/доработки.

В хранилище хранятся история, все commit-ы с файлами и мельчайшими изменениями, а пользователь, желающий продолжить работу над проектом, вызывает себе рабочую копию (то есть, копирует информацию из тех commit-ов и версий, которые нужны ему / которые он планирует модифицировать).

3) Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида.

Ответ: централизованные VCS — системы контроля версий, где в общем хранилище лежит репозиторий, из которого каждый пользователь берёт файлы, изменяет их и возвращает, так что меняются все файлы (откатиться до предыдущей версии в случае ошибки никто не запрещал). Таким видом VCS является *Subversion*.

Децентрализованные VCS — системы контроля версий, где у каждого пользователя может быть своя версия репозитория, и получить доступ можно ко всем версиям репозитория. Например, таким видом VCS является *Git*.

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

Создаётся репозиторий. Туда отправляются первые файлы, образуя собой первую рабочую копию. Человек берёт рабочую копию, модифицирует её и выгружает уже обновлённую, создавая вторую версию проекта. Потом третью, четвёртую... Возможен откат к предыдущей версии в любой момент времени, так как даже при возврате к предыдущей версии достаточно сложно войти в противоречие с самим собой.

5) Опишите порядок работы с общим хранилищем VCS.

Создаётся репозиторий. Однако теперь необходимо добавить систему контроля версий, которая сможет разделять версии репозитория разных людей, объединять их и устранять конфликты между версиями. Откаты к предыдущим версиям теперь должны согласовываться, различные изменения вносятся в различные файлы в разных ветках хранилища дабы избежать конфликтов ещё на этапе изменения.

6) Каковы основные задачи, решаемые инструментальным средством git?

У *Git* две основных задачи: первая — хранить информацию о всех изменениях в вашем коде, начиная с самой первой строчки, а вторая — обеспечение удобства командной работы над кодом.

7) Назовите и дайте краткую характеристику командам git.

git init — создание основного дерева репозитория.

git pull — получение обновлений (изменений) текущего дерева из центрального репозитория.

**git push ** — отправка всех произведённых изменений локального дерева в центральный репозиторий.

git status — просмотр списка изменённых файлов в текущей директории.

git diff — просмотр текущих изменений.

Сохранение текущих изменений:

- *git add .* — добавить все изменённые и/или созданные файлы и/или каталоги.
- *git add имена_файлов* — добавить конкретные изменённые и/или созданные файлы и/или каталоги.
- *git rm имена_файлов* — удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории).

Сохранение добавленных изменений:

- *git commit -am 'Описание коммита'* — сохранить все добавленные изменения и все изменённые файлы.
- **git commit ** — сохранить добавленные изменения с внесением комментария через встроенный редактор.

git checkout -b имя_ветки — создание новой ветки, базирующейся на текущей.

git checkout имя_ветки — переключение на некоторую ветку (при переключении на ветку, которой ещё нет в локальном репозитории, она будет создана и связана с удалённой).

git push origin имя_ветки — отправка изменений конкретной ветки в центральный репозиторий.

git merge --no-ff имя_ветки — слияние ветки с текущим деревом.

Удаление ветки:

- *git branch -d имя_ветки* — удаление локальной уже слитой с основным деревом ветки.
- *git branch -D имя_ветки* — принудительное удаление локальной ветки.
- *git push origin :имя_ветки* — удаление ветки с центрального репозитория.

8) Приведите примеры использования при работе с локальным и удалённым репозиториями.

В локальном репозитории мы используем *git pull* чтобы получить данные текущей версии, в локально же репозитории мы изменяем её

состав (добавляем новый файл командой *git add*, например), а затем командой *git checkout имя_ветки* переключаемся на нужную ветку в удалённом репозитории и командой *git push* отправляем новую версию на рассмотрение и проверку.

9) Что такое и зачем могут быть нужны ветки (branches)?

Ветки – различные каталоги в удалённом хранилище. Могут быть нужны в проектах со сложной структурой, где постоянно нужно контролировать изменения и где ошибки в последних версиях могут «уронить» всё приложение. В таких проектах ветки могут быть удобны для хранения различных вариантов изменений файлов (на всякий случай). Наиболее же удобными ветки являются в проектах, где много людей, которые работают над разными файлами системы. В таких проектах ветки – настоящее спасение от конфликтов между разными версиями разных файлов и в таких проектах разные ветки обычно принадлежат разным людям (реже группам) для наиболее удобной работы.

10) Как и зачем можно игнорировать некоторые файлы при commit?

Некоторые файлы при commit можно игнорировать, если использовать команду *git add* с конкретными (нужными) названиями файлов. Это может быть необходимо, если в проект нет необходимости загружать исполнительные файлы, или какие-то промежуточные файлы, которые не несут в себе никакой пользы для итоговой рабочей версии, но использовались при написании файла и могут быть востребованы при продолжении разработки.