

Лабораторная работа №2

Настройка системы git

Лушин Артем Андреевич

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Контрольные вопросы	13
4	Выводы	17
	Список литературы	18

Список иллюстраций

2.1	Окно создания аккаунта	6
2.2	Профиль github	7
2.3	Предварительная конфигурация	7
2.4	Генерация ключа	7
2.5	Регистрация и сохранения ключа на сайте	8
2.6	GPG - ключ	8
2.7	Данные ключа	9
2.8	Экспорт ключа	9
2.9	gh auto	10
2.10	Создание каталогов	10
2.11	репозитории на сайте	11
2.12	Репозиторий на сайте	11
2.13	Копирование репозитория	12
2.14	Отправка данных	12

Список таблиц

1 Цель работы

Изучить идеологию и применение средств контроля версий.

Освоить умения по работе с git.

2 Выполнение лабораторной работы

1) Я создал аккаунт на github.

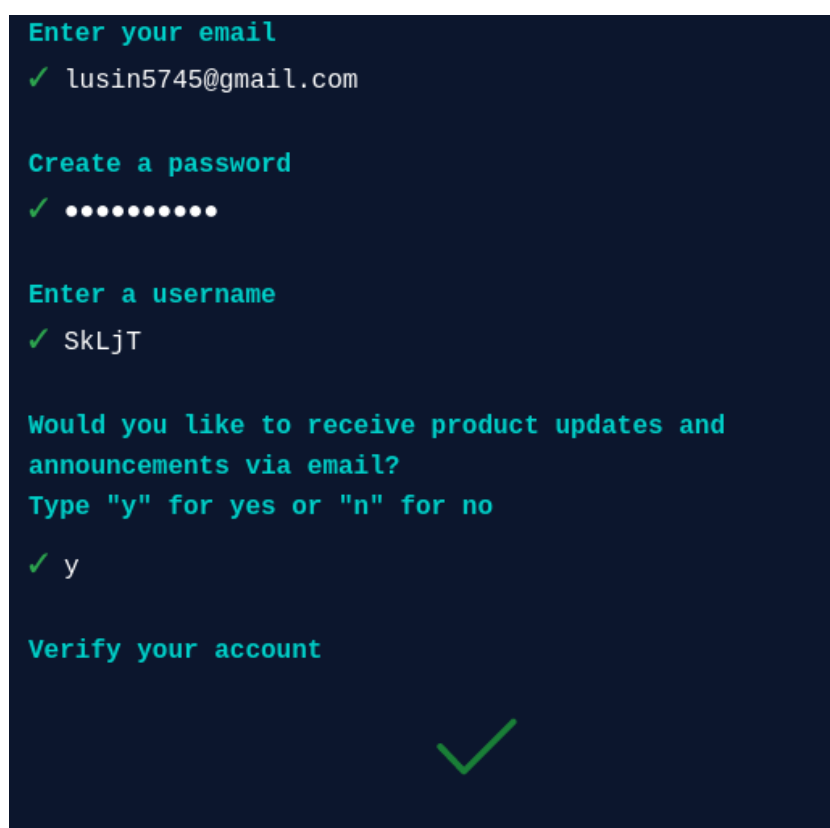


Рис. 2.1: Окно создания аккаунта

2) Профиль на github.

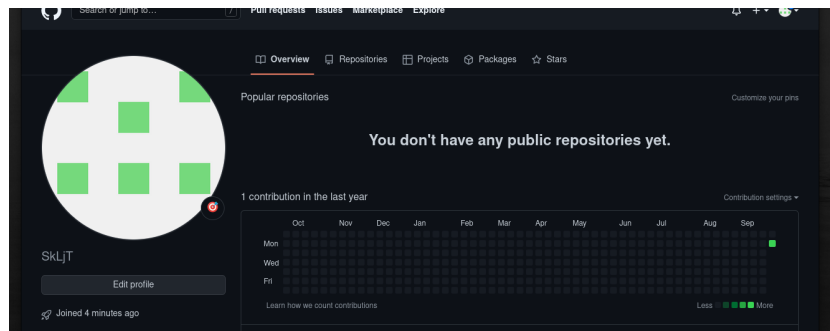


Рис. 2.2: Профиль github

3) Я сделал предварительную конфигурацию в терминале.

```
[aalushin@fedora ~]$ git config --global user.email "lusin5745@gmail.com"
[aalushin@fedora ~]$ git config --global user.name "SKLJT"
[aalushin@fedora ~]$ git config --global core.quotepath false
[aalushin@fedora ~]$ git config --global init.defaultBranch master
[aalushin@fedora ~]$ git config --global core.autocrlf input
[aalushin@fedora ~]$ git config --global core.safecrlf warn
```

Рис. 2.3: Предварительная конфигурация

4) Создание SSH -ключа.

```
[aalushin@fedora ~]$ ssh-keygen -C "artem lushin lusin5745@gmail.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/aalushin/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/aalushin/.ssh/id_rsa
Your public key has been saved in /home/aalushin/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:dN4vjEjiE83F/k1DuEaKVg14Pe10a549hGMzqIysqKY artem lushin lusin5745@gmail.com
The key's randomart image is:
---[RSA 3072]-----+
  ... . |
  ...oo.. |
  ..= +O. |
  + B + oo |
  o S + =O+ |
  . = . * X+o |
  + + o =oBo |
  . . + o .o.. |
  E... .. |
  ---[SHA256]-----+
[aalushin@fedora ~]$
```

Рис. 2.4: Генерация ключа

5) Добавления ssh -ключа на сайт.

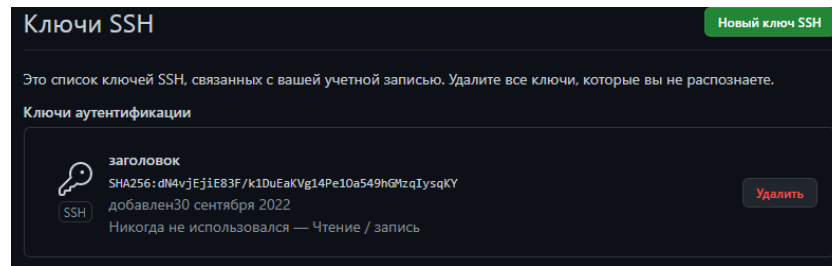


Рис. 2.5: Регистрация и сохранения ключа на сайте

6) Создание gpg - ключа.

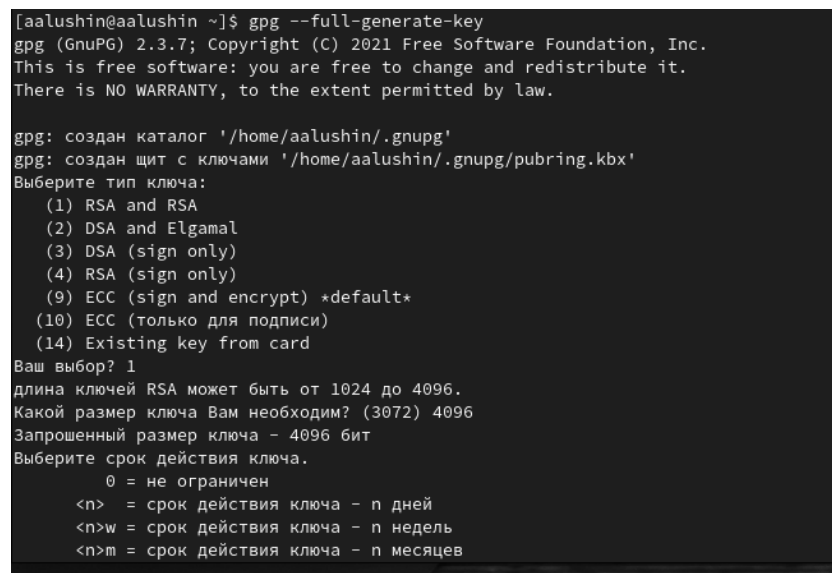


Рис. 2.6: GPG - ключ


```

Вы выбрали следующий идентификатор пользователя:
"Artem <lusin5745@gmail.com>"

Сменить (N)Имя, (C)Примечание, (E)Адрес; (O)Принять/(Q)Выход? o
Необходимо получить много случайных чисел. Желательно, чтобы Вы
в процессе генерации выполняли какие-то другие действия (печать
на клавиатуре, движения мыши, обращения к дискам); это даст генератору
случайных чисел больше возможностей получить достаточное количество энтропии.
Необходимо получить много случайных чисел. Желательно, чтобы Вы
в процессе генерации выполняли какие-то другие действия (печать
на клавиатуре, движения мыши, обращения к дискам); это даст генератору
случайных чисел больше возможностей получить достаточное количество энтропии.
gpg: /home/aalushin/.gnupg/trustdb.gpg: создана таблица доверия
gpg: создан каталог '/home/aalushin/.gnupg/openpgp-revocs.d'
gpg: сертификат отзыва записан в '/home/aalushin/.gnupg/openpgp-revocs.d/82DCE11
53DB80B042B95FB90D87BF3F8306FDED0.rev'.
открытый и секретный ключи созданы и подписаны.

pub   rsa4096 2023-02-14 [SC]
      82DCE1153DB80B042B95FB90D87BF3F8306FDED0
uid           Artem <lusin5745@gmail.com>
sub   rsa4096 2023-02-14 [E]

[aalushin@aalushin ~]$

```

Рис. 2.7: Данные ключа

7) экспорт ключа в формате ASCII по отпечатку.

```

      82DCE1153DB80B042B95FB90D87BF3F8306FDED0
uid           [ абсолютно ] Artem <lusin5745@gmail.com>
ssb   rsa4096/41C9B42BFAC94CD8 2023-02-14 [E]

[aalushin@aalushin ~]$ gpg --armor --export D87BF3F8306FDED0
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBGPvzSBEADcKnFJ0rF2L0gJZZUXhudFh2rJFT7q+TERYbDF2CaATBuX1lnR
jLCEt73e8qpbFeyz95dg7rjUxr44nrjA+BVf5mjtua4rkLSI/WCBY04zjrYEt/6N
6Cg0hHo035qgkGiG23Z4Yg0AN06//fn/IgG35pNkrPoi5Eqy/a3wMcmzYuFZYGio
ggIVfVUL7VTsWZvyXiIL7Q2CumI/BHbqx4gSWpFQXNUcVDYJyh4xPeXVKfXUtuzy
AMj5Pt96+Q6E5M10Q5MuX0FUPCE3q7q0SR57LRIafTEl7Yb3zcuvGESVN9UJ3z90
IVHKS7gw0nDBd2qNEjCQfyQeHcrJSR+PDV5AHD/0tPvGTDBmR7RRV8tIGgBAfYSZ
UDMcqMYxwZxsgl8Wlo5MLLP0Re0Jt3ZaUwPYiEN0Rp6dsUNNtpn9kUL2jK8VgMC
QGILJDrfmKiY6/a0+KdUjxahsAeJy94Qzytbt1c6UpW0PEHPQgBs+mf5townjNNU
Q+yj+blhYAsrA6s66eCUGFLDorb+aTppyQBLY7XModuWtQTsNP5Ck7GA18ay4k3u
6UzrT0Qzu08KlisdXIN6SbV/Fq84WzURbx9AZRALsFls+g+IppksZc+jK3509I8J
/ZB80iM9jw15UJwwLkoBKbePstKDzFHu76YT8ucltqPXEUSsC/KabUDp8QARAQAB
tBtBcnRlbnSA8bHVzaW41NzQ1QGdtYWlsLmNvbT6JALEEEwEiADSWIQSC30EVPBgL
BCuV+5DYe/P4MG/e0AUCY+u/OwIbAwULCQgHAgIiAgYVCgkICwIEFgIDAQIeBwIX
gAAKCRDYE/P4MG/e0DeFD/9f1dhhq0j/vQqRuTDPWVWkbhPK2XWk/0Xknzfr3wT2
ELB95vIIs2pFzgkYAHf0sC0NgOyA3fquUss8uZnCDVMXUsIdAFT/hvdp/DVKujGA
aaFpNri4EZtjfvBUL+rATRUpKAq00NHMJ18D7/Sg9vocsJURSqrpmLsK3pVHqXL
CMkEpKetqbdWw83ENWrAMr8Pj5F0KdaIpEupk+3Ju5AWN+o/6JRENms3T5H/oPVf

```

Рис. 2.8: Экспорт ключа

8) Авторизация в gh.

```
gh auth login~[aalushin@aalushin Операционные системы]$ gh auth login
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations? SSH
? Upload your SSH public key to your GitHub account? /home/aalushin/.ssh/id_rsa.
? Title for your SSH key: SHA256:dN4vjEjiE83F/k1DuEaKVg14Pe10a549hGMzqIysqKY
? How would you like to authenticate GitHub CLI? Login with a web browser

First copy your one-time code: 27A4-B4FC
Press Enter to open github.com in your browser...
/ Authentication complete.
- gh config set -h github.com git_protocol ssh
/ Configured git protocol
HTTP 422: Validation Failed (https://api.github.com/user/keys)
key is already in use
[aalushin@aalushin Операционные системы]$
```

Рис. 2.9: gh auto

9) Я создал каталоги с последним названием “Операционные системы”

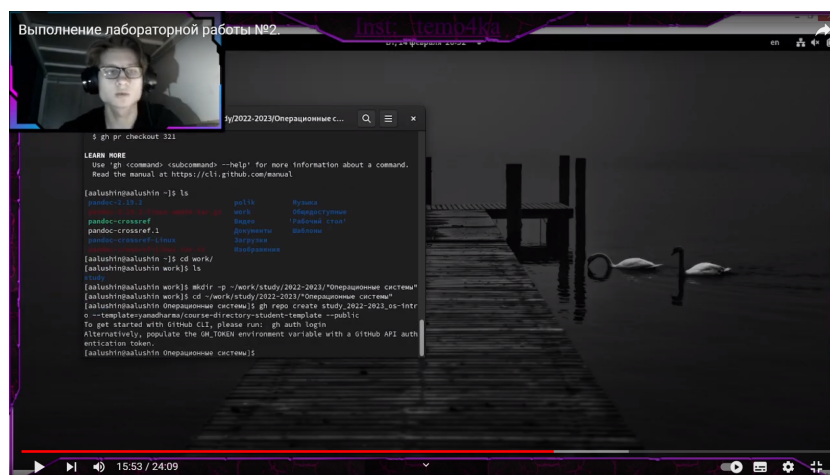


Рис. 2.10: Создание каталогов

10)Создание репозиториев на сайте github.

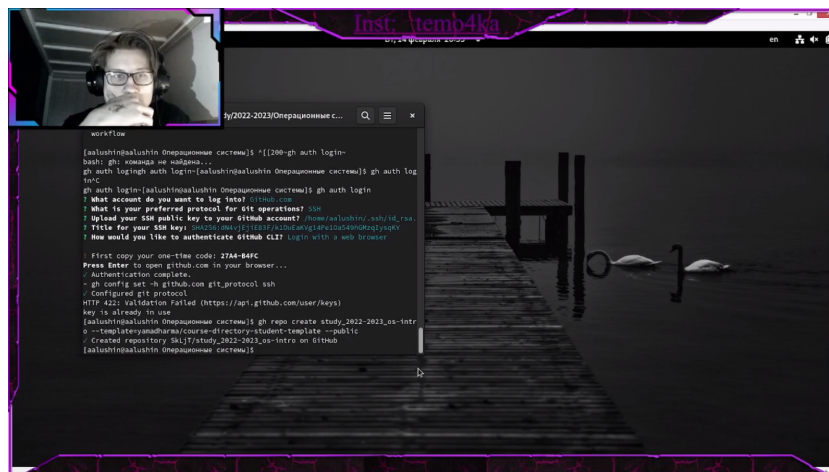


Рис. 2.11: репозитории на сайте

11) Проверка наличия репозитория на сайте.

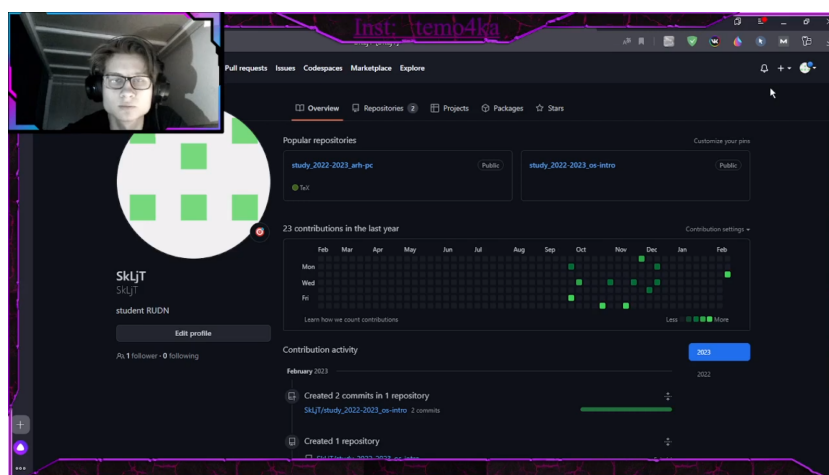


Рис. 2.12: Репозиторий на сайте

12) Копирование репозитория с сайта на компьютер.

```
aalushin@aalushin:~/work/study/2022-2023/Операционные с...
tation-markdown-template.git) зарегистрирован по пути «template/presentation»
Подмодуль «template/report» (https://github.com/yamadharma/academic-laboratory-r
eport-template.git) зарегистрирован по пути «template/report»
Клонирование в «/home/aalushin/work/study/2022-2023/Операционные системы/os-intro
template/presentation»...
remote: Enumerating objects: 82, done.
remote: Counting objects: 100% (82/82), done.
remote: Compressing objects: 100% (57/57), done.
remote: Total 82 (delta 28), reused 77 (delta 23), pack-reused 0
Получение объектов: 100% (82/82), 92.90 КиБ | 761.00 КиБ/с, готово.
Определение изменений: 100% (28/28), готово.
Клонирование в «/home/aalushin/work/study/2022-2023/Операционные системы/os-intro
template/report»...
remote: Enumerating objects: 101, done.
remote: Counting objects: 100% (101/101), done.
remote: Compressing objects: 100% (70/70), done.
remote: Total 101 (delta 40), reused 88 (delta 27), pack-reused 0
Получение объектов: 100% (101/101), 327.25 КиБ | 1.02 МиБ/с, готово.
Определение изменений: 100% (40/40), готово.
Submodule path 'template/presentation': checked out 'b1be3800ee91f5809264cb755d3
16174540b753e'
Submodule path 'template/report': checked out '1d1b61dcac9c287a83917b82e3aef11a3
9b1e3b2'
[aalushin@aalushin Операционные системы]$
```

Рис. 2.13: Копирование репозитория

13) Отправка измененных данных на сайт.

```
aalushin@aalushin:~/work/study/2022-2023/Операционные с...
by
create mode 100755 project-personal/stage6/report/pandoc/filters/pandoc_tableno
s.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/__i
nit__.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/cor
e.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/mai
n.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/pan
docattributes.py
create mode 100644 project-personal/stage6/report/report.md
[aalushin@aalushin os-intro]$ git push
Перечисление объектов: 40, готово.
Подсчет объектов: 100% (40/40), готово.
При сжатии изменений используется до 2 потоков
Сжатие объектов: 100% (30/30), готово.
Запись объектов: 100% (38/38), 342.39 КиБ | 2.00 МиБ/с, готово.
Всего 38 (изменений 4), повторно использовано 0 (изменений 0), повторно использо
вано пакетов 0
remote: Resolving deltas: 100% (4/4), completed with 1 local object.
To github.com:SkLjT/study_2022-2023_os-intro.git
d3073c7..d1ec7bd master -> master
[aalushin@aalushin os-intro]$
```

Рис. 2.14: Отправка данных

3 Контрольные вопросы

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

Система контроля версий (VCS) - программное обеспечение для облегчения работы с изменяющейся информацией. Система управления версиями позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое. Такие системы наиболее широко используются при разработке программного обеспечения для хранения исходных кодов разрабатываемой программы. Однако они могут с успехом применяться и в других областях, в которых ведётся работа с большим количеством непрерывно изменяющихся электронных документов.

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

- Хранилище – репозиторий - место хранения всех версий и служебной информации.
- Commit - это команда для записи индексированных изменений в репозиторий.
- История – место, где сохраняются все коммиты, по которым можно посмотреть данные о коммитах.
- Рабочая копия – текущее состояние файлов проекта, основанное на версии, загруженной из хранилища.

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

- Централизованные системы – это системы, в которых одно основное хранилище всего проекта, и каждый пользователь копирует необходимые ему файлы, изменяет и вставляет обратно. Пример – Subversion.

- Децентрализованные системы – система, в которой каждый пользователь имеет свой вариант репозитория и есть возможность добавлять и забирать изменения из репозитория. Пример – Git.

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

- В рабочей копии, которую исправляет человек, появляются правки, которые отправляются в хранилище на каждом из этапов. То есть в правки в рабочей копии появляются, только если человек делает их (отправляет их на сервер) и никак по-другому.

5) Опишите порядок работы с общим хранилищем VCS.

- Если хранилище общее, то в рабочую копию каждого, кто работает над проектом, приходят изменения, отправленные на сервер одним из команды. Рабочая правка каждого может изменяться вне зависимости от того, делает ли конкретный человек правки или нет.

6) Каковы основные задачи, решаемые инструментальным средством git?

- У Git две основных задачи: первая — хранить информацию обо 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.

- создание основного дерева репозитория: `git init`
- получение обновлений (изменений) текущего дерева из центрального репозитория: `git pull`
- отправка всех произведённых изменений локального дерева в центральный репозиторий: `git push`
- просмотр списка изменённых файлов в текущей директории: `git status`
- просмотр текущих изменений: `git diff`
- сохранение текущих изменений: – добавить все изменённые и/или созданные файлы и/или каталоги: `git add .`
- добавить конкретные изменённые и/или созданные файлы и/или каталоги: `git add`
- удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории): `git rm имена_файлов`
- сохранить все добавленные изменения и все изменённые файлы: `git commit -am 'Описание коммита'`
- сохранить добавленные изменения с внесением комментария через встроенный редактор: `git commit`
- создание новой ветки, базирующейся на текущей: `git checkout -b имя_ветки`
- переключение на некоторую ветку: `git checkout имя_ветки` (при переключении на ветку, которой ещё нет в локальном репозитории, она будет создана и связана с удалённой)
- 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 удалённым

репозиториями.

- Работа с удаленным репозиторием: `git remote` – просмотр списка настроенных удаленных репозиторияев.

- Работа с локальным репозиторием: `git status` - выводит информацию обо всех изменениях, внесенных в дерево директорий проекта по сравнению с последним коммитом рабочей ветки

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

- Ветка (англ. branch) — это последовательность коммитов, в которой ведётся параллельная разработка какого-либо функционала. Ветки нужны, чтобы несколько программистов могли вести работу над одним и тем же проектом или даже файлом одновременно, при этом не мешая друг другу. Кроме того, ветки используются для тестирования экспериментальных функций: чтобы не повредить основному проекту, создается новая ветка специально для экспериментов.

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

- Игнорируемые файлы — это, как правило, артефакты сборки и файлы, генерируемые машиной из исходных файлов в вашем репозитории, либо файлы, которые по какой-либо иной причине не должны попадать в коммиты. В Git нет специальной команды для указания игнорируемых файлов: вместо этого необходимо вручную отредактировать файл `.gitignore`. Временно игнорировать изменения в файле можно командой `git update-index --assume-unchanged`

4 Выводы

Я изучил идеологию и применение средств контроля версий, а также освоил умения по работе с git.

Список литературы