

Отчёт по лабораторной работе №2

Специальность: архитектура компьютеров

Кудинец Максим Антонович

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
4.1	Установка git и gh	9
4.2	Базовая настройка git.	9
4.3	Создание ssh ключа.	9
4.3.1	Добавление ssh-ключа в учетную запись ГитХаб.	10
4.4	Создание PGP ключа.	10
4.4.1	Добавление ключа на ГитХаб.	12
4.4.2	Настройка автоматических подписей коммитов git	13
4.5	Настройка gh	13
4.6	Создание и настройка репозитория курса.	13
5	Контрольные вопросы	15
6	Выводы	19
	Список литературы	20

Список иллюстраций

4.1	Ввод команд в терминал	9
4.2	Создание ключа	10
4.3	Новый ключ ssh	10
4.4	Создание нового pg ключа	11
4.5	Вывод ключа в терминал	12
4.6	Новый ключ PGP	12
4.7	Настройка необходимых подписей коммитов	13
4.8	Настройка gh и авторизация в браузере	13
4.9	Созданный репозиторий, папка первой лабораторной работы . . .	14
4.10	Отправка файлов на сервер	14

Список таблиц

1 Цель работы

Целью данной работы является изучение идеологии и применения средств контроля версий и освоение умения по работе с git.

2 Задание

1. Создать базовую конфигурацию для работы с git.
2. Создать ключ SSH.
3. Создать ключ PGP.
4. Настроить подписи git.
5. Зарегистрироваться на Github.
6. Создать локальный каталог для выполнения заданий по предмету.

3 Теоретическое введение

Системы контроля версий (Version Control System, VCS) 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. При внесении изменений в содержание проекта система контроля версий позволяет их фиксировать, совмещать изменения, произведённые разными участниками проекта, производить откат к любой более ранней версии проекта, если это требуется.

В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером. Участник проекта (пользователь) перед началом работы посредством определённых команд получает нужную ему версию файлов. После внесения изменений, пользователь размещает новую версию в хранилище. При этом предыдущие версии не удаляются из центрального хранилища и к ним можно вернуться в любой момент. Сервер может сохранять не полную версию изменённых файлов, а производить так называемую дельта-компрессию — сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных.

Системы контроля версий поддерживают возможность отслеживания и разрешения конфликтов, которые могут возникнуть при работе нескольких человек над одним файлом. Можно объединить (слить) изменения, сделанные разными участниками (автоматически или вручную), вручную выбрать нужную версию,

отменить изменения вовсе или заблокировать файлы для изменения. В зависимости от настроек блокировка не позволяет другим пользователям получить рабочую копию или препятствует изменению рабочей копии файла средствами файловой системы ОС, обеспечивая таким образом, привилегированный доступ только одному пользователю, работающему с файлом.

Системы контроля версий также могут обеспечивать дополнительные, более гибкие функциональные возможности. Например, они могут поддерживать работу с несколькими версиями одного файла, сохраняя общую историю изменений до точки ветвления версий и собственные истории изменений каждой ветви. Кроме того, обычно доступна информация о том, кто из участников, когда и какие изменения вносил. Обычно такого рода информация хранится в журнале изменений, доступ к которому можно ограничить.

В отличие от классических, в распределённых системах контроля версий центральный репозиторий не является обязательным.

Среди классических VCS наиболее известны CVS, Subversion, а среди распределённых — Git, Bazaar, Mercurial. Принципы их работы схожи, отличаются они в основном синтаксисом используемых в работе команд.

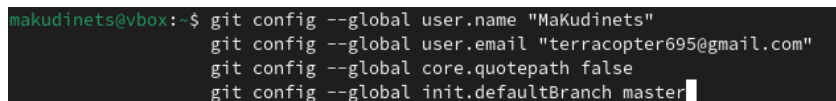
4 Выполнение лабораторной работы

4.1 Установка git и gh

Установим гит командой `dnf install git`, установим gh командой `dnf install gh`

4.2 Базовая настройка git.

Открываем терминал. При помощи команд `git config --global user.name` и `git config --global user.email` зададим имя пользователя и адрес электронной почты. При помощи команды `git config --global core.quotepath false` настроим utf-8 в выводе сообщений git. При помощи команды `git config --global init.defaultBranch master` зададим начальной ветке имя master. (рис. 4.1)



```
makudinets@vbox:~$ git config --global user.name "MaKudinets"
git config --global user.email "terracooper695@gmail.com"
git config --global core.quotepath false
git config --global init.defaultBranch master
```

Рис. 4.1: Ввод команд в терминал

4.3 Создание ssh ключа.

Для создания ключа используем команду `ssh-keygen -t` в терминале. (рис. 4.2) Зададим ключу размер 4096 бит. Сменим пароль при помощи команды `ssh-keygen -p`.



Рис. 4.2: Создание ключа

4.3.1 Добавление ssh-ключа в учетную запись ГитХаб.

Копируем созданный ключ и переносим его на сайт гитхаб в раздел ssh и gpg keys.

Создаем новый ключ, задаем ему название и переносим ключ в поле key, добавляем ключ на сайт. (рис. 4.3)

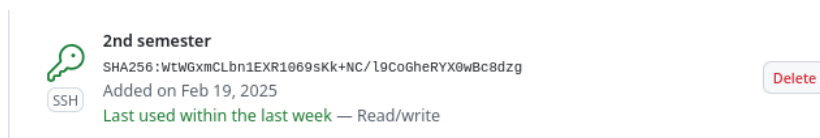


Рис. 4.3: Новый ключ ssh

4.4 Создание PGP ключа.

Генерируем ключ командой `gpg --full-generate-key`, настраиваем его по заданным требованиям. (рис. 4.4)

```

makudinets@bpi:~$ gpg --full-generate-key
gpg (GnuPG) 2.4.4; Copyright (C) 2024 g10 Code GmbH
This is free software; you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Выберите тип ключа:
(1) RSA and RSA
(2) DSA and Elgamal
(3) DSA (sign only)
(4) RSA (sign only)
(9) ECC (sign and encrypt) *default*
(10) ECC (только для подписи)
(14) Existing key from card
Ваш выбор? 1
длина ключей RSA может быть от 1024 до 4096.
Какой размер ключа Вам необходим? (3072) 4096
Запрошенный размер ключа - 4096 бит
Выберите срок действия ключа.
0 = не ограничен
<n> = срок действия ключа - n дней
<nw> = срок действия ключа - n недель
<nм> = срок действия ключа - n месяцев
<ny> = срок действия ключа - n лет
Срок действия ключа? (0) 0
Срок действия ключа не ограничен
Все верно? (y/N) y

GnuPG должен составить идентификатор пользователя для идентификации ключа.

Ваше полное имя: Кудинец Максим Антонович
Адрес электронной почты: terracopter695@gmail.com
Примечание:
Используется таблица символов 'utf-8'.
Вы выбрали следующий идентификатор пользователя:
"Кудинец Максим Антонович <terracopter695@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: сертификат отныне записан в '/home/makudinets/.gnupg/openpgp-revocs.d/CEDABC09FC3228305BBBEA258AA58BEE69CCETFC.rev'.
открытый и секретный ключи созданы и подписаны.

```

Рис. 4.4: Создание нового pg ключа

Выводим ключ в терминал командой `gpg --list-secret-keys --keyid-format LONG`.
После этого экспортируем его командой `gpg --armor --export`. (рис. 4.5)

```

makudinets@vbox: ~$ gpg --armor --export CEDABC09FC32283058B8EA258AA58BEE69CCE7FC
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBGe4qr8BEAD0tic0Rrxj/I3/CoeWe0SCZJ6lXNe7nmoUQvkFvXsMbh4QPLPg
H79oEU6b0ZysGTYZ35lroSHJZCZqEm+RnrugKWS55LseylVLjl+VfxqFSdDCECs
pcZ0k1NqZufmZamYM58Up1ZrY+yuKdBoyYkJu+ozmXCusg4AbWgq8X4nNxmMQYG
vdmCI30lX8MGmVu63Zd/SEIXNiGJ2Bn/SSRjtUET6t+iT7Vf6Bd8cjjudns0jTPGZ
/IuufXx2CTuR9rIc8t7monKYHEwLAAQSEGCzkk5Tdpej2A7fnvHJm6S1NAdQyhH4
DUAPSVr7czWB9w8DeEceTXZwhCnSiU83q6Nofm75UQHm3+87Pxgedl6skG9D9WUq
gCzUVdyEWvD6fBQklqq3GBAS900CUK1FOt0tCxcfc0YsomwB6ErDxXhmQDLAo3G
LVZTXBLXBDIsSN/V1TVR0fkea0eHHdTURXK6RlDWFUcd1Z0/uoDdXEJt6Z7cLfaO
R54mVv7qgSMlZuA2WRicnTYvj5/W/m/46MnxC+zXrMn+jH6AGX2cNXLhacJhl/fB
VB9Qntu0TCG1BTxvv4E6CHfzDgt0w1wCjghXZdyAa8DIp8dRxnNdGacMy5aYy/H0
Ez1a6lhl6LqhQI38MIHeLh+HGBpbw8rFNaaTn3NxyjgSqUY5SuSHOXIAQARAQAB
tEn0mt6D0LTQuNC90LXRhIDQnNCw0LrRgdC40LWg0JDQvdG90L7QvdC+0LLQuNGH
IDx0ZXJyYWNvcHRlcjY5NUBnbWpbc5jb20+iQJRBBMBCAA7FiEEZtq8CfwyKDBY
u+oliqWL7mnM5/wfAme4qr8CGwMFCwkIBwICIGFQoJCAsCBBCAwEChgcCF4AA
CgkqIqWL7mnM5/w54g/+PAHPtldf0LdLpJCGpJX22yNpAchkJrsbn3AKyK3aFLAQ
TURLF8Qx4ChyRqoba5eTYKiD2HeDtkjR/0XWQ4W6DqAmFRznG6rL5AX2eUpn1P8w
zIQpESEIqfIINkXxelvTpsboDaM/lwE5QNvny58ekYA3L9GV+7irpuh4Dr46dhq
F9YIvCLaon2/y6Q7sEbTtsZYWFPagpFbhq5lwc5xaoYQmZq4nFPALeLx2wnT5HAA
2c3fmsZwf00j/OLLR0qRdLWGN4MubjoUSxtnoW8vFY7GPQ5WV18MjB7BCogngaEZ
4Ayu56oQfXou/YusEShywwRL5i2i+Dz10pr9X5l/EaSXQ2nJ2/SSbaVQe/4GtLqF
lAkWLq7YVW8mKMTQ0gQlhxM/0vfdL7lQ1MkqoRjXcZHCpc62jA7qAad7ll1W4pz9
McvlCp6Dg4wTj7KNdXE7oZlrRsib0jPq1CSGGAxvZANmzOdL/nrRF0ztcBrPhsI
L20KMcLl1mkp+f6s66WpV4uhHiX+xSSkUtbqgIYtZAX+CyHSSqV6CBAx0vtpRtBy
n824N4bqQu/0WSUH8TDVSAoKLYnCutgo3GhAG5DIhg4jKP34tUtANIPa0oiDu
ONC1au149SN+54cDrf/r1cw2pKXUEJXuQR725MsfVtpEjhYxkEL5jEu/q5YeN4u5
Ag0EZ71qvWEQAMrmMoZf+Jr8nYckgXyvEkTcMxxtmNo62sHh/BQQZrQ6PXLI0N4W
9QZy1irEXcwLEZvfyYv1j7mb6qcANbxX/x0psk8+vcYJHeVnMlBSUQbF0C4tsdxy
q2h4areqWnueIigdmzegGxvCMQywwVY8992WjK9yr2uJQ0B6Sv4tGFK5KplWrXfa
eDRtojsL588IfzUXCreN/5Ira90C0qjxxu9X0Fg5AsSx0W2YjthTtKiSmv4dZSg
V17UuJJZL3TuvMmJhV5mGwa60CGZ9l5oe8V/8kiFN0uyIU/kMwPXC7bPs1TeLW
RLjPWxORD++XC00Eqisn6KD/e2BQ0MDBL9b0hbyfavnF3RcLrj1xD6gidCh7wog
ltYIFKrgAb6D+Z9JdKquWN5M40GdM8BUeEGwdB9Xf7P8aj/gh6jVskHXCQo6SA0Z
LU07Ka8suu3jrFVqbojH18b3N4RT7gyPMwzhLmDpfdCEe16MJIReVfPGrAKML0pt
0w70AQF3Gsv+vCyaL3Lf3tDnDgtsw97ryjQ2VUQ2+3joB9qPHBVI1bK0PrcPC

```

Рис. 4.5: Вывод ключа в терминал

4.4.1 Добавление ключа на ГитХаб.

Скопировав ключ, переносим его на ГитХаб, создаем на сайте новый ключ и вставляем скопированный ключ в необходимое поле. (рис. 4.6)

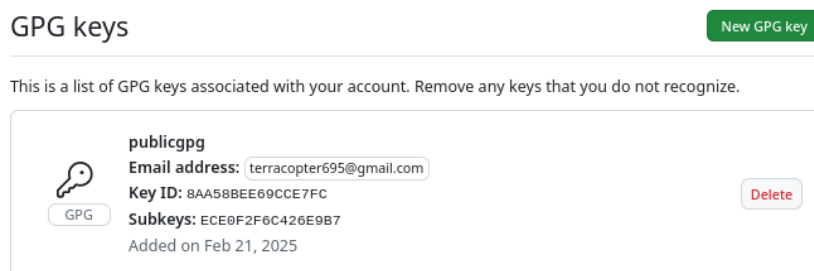
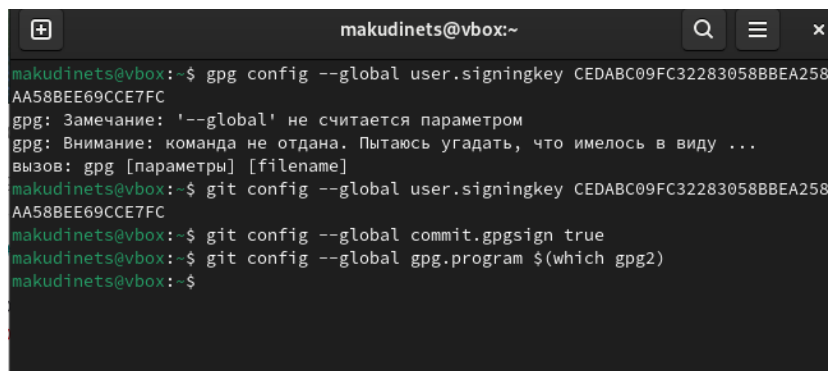


Рис. 4.6: Новый ключ PGP

4.4.2 Настройка автоматических подписей коммитов git

При помощи команд `git config --global user.signingkey`, `git config --global commit.gpgsign true` и `git config --global gpg.program $(which gpg2)` самостоятельно выбираем подписи коммитов в git. (рис. 4.7)

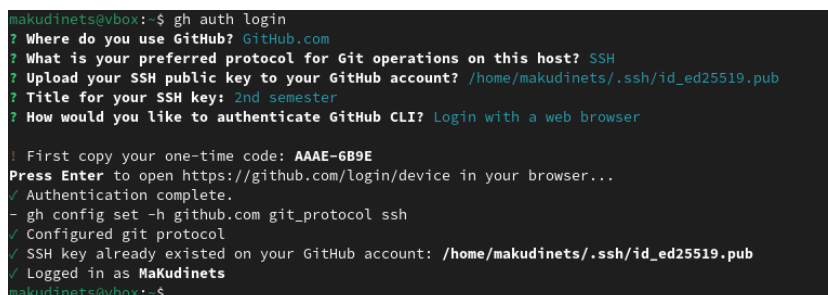


```
makudinets@vbox:~$ gpg config --global user.signingkey CEDABC09FC32283058BBEA258
AA58BEE69CCE7FC
gpg: Замечание: '--global' не считается параметром
gpg: Внимание: команда не отдана. Пытаюсь угадать, что имелось в виду ...
вызов: gpg [параметры] [filename]
makudinets@vbox:~$ git config --global user.signingkey CEDABC09FC32283058BBEA258
AA58BEE69CCE7FC
makudinets@vbox:~$ git config --global commit.gpgsign true
makudinets@vbox:~$ git config --global gpg.program $(which gpg2)
makudinets@vbox:~$
```

Рис. 4.7: Настройка необходимых подписей коммитов

4.5 Настройка gh

Введя в терминал команду `gh auth login`, ответим на необходимые в терминале вопросы, после чего авторизуемся через браузер. (рис. 4.8)



```
makudinets@vbox:~$ gh auth login
? Where do you use GitHub? GitHub.com
? What is your preferred protocol for Git operations on this host? SSH
? Upload your SSH public key to your GitHub account? /home/makudinets/.ssh/id_ed25519.pub
? Title for your SSH key: 2nd semester
? How would you like to authenticate GitHub CLI? Login with a web browser

! First copy your one-time code: AAAE-6B9E
Press Enter to open https://github.com/login/device in your browser...
✓ Authentication complete.
- gh config set -h github.com git_protocol ssh
✓ Configured git protocol
✓ SSH key already existed on your GitHub account: /home/makudinets/.ssh/id_ed25519.pub
✓ Logged in as MaKudinets
makudinets@vbox:~$
```

Рис. 4.8: Настройка gh и авторизация в браузере

4.6 Создание и настройка репозитория курса.

Используя команды `mkdir`, `gh repo`, `create study` и `git clone` создаем репозиторий курса. (рис. 4.9)

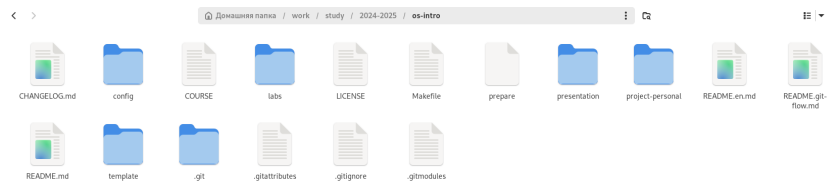


Рис. 4.9: Созданный репозиторий, папка первой лабораторной работы

Отправляем файлы первой лабораторной работы на сервер. (рис. 4.10)

```
makudinets@vbox:~/work/study/2024-2025/os-intro/labs/lab01$ git add .
makudinets@vbox:~/work/study/2024-2025/os-intro/labs/lab01$ git commit -am "Загрузки"
Текущая ветка: master
Эта ветка соответствует «origin/master».

Неотслеживаемые файлы:
(используйте «git add <файл>...», чтобы добавить в то, что будет включено в коммит)
  ../README.md
  ../README.ru.md
  ../lab02/
  ../lab03/
  ../lab04/
  ../lab05/
  ../lab06/
  ../lab07/
  ../lab08/
  ../lab09/
  ../lab10/
  ../lab11/
  ../lab12/
  ../lab13/
  ../lab14/
  ../lab15/
  ../../prepare
  ../../presentation/
  ../../project-personal/

индекс пуст, но есть неотслеживаемые файлы
(используйте «git add», чтобы проиндексировать их)
makudinets@vbox:~/work/study/2024-2025/os-intro/labs/lab01$ git push
Everything up-to-date
makudinets@vbox:~/work/study/2024-2025/os-intro/labs/lab01$
```

Рис. 4.10: Отправка файлов на сервер

5 Контрольные вопросы

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

Система контроля версий — 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. Системы контроля версий (Version Control System, VCS) применяются для:

- Хранение полной истории изменений причин всех производимых изменений
- Откат изменений, если что-то пошло не так
- Поиск причины и ответственного за появления ошибок в программе
- Совместная работа группы над одним проектом
- Возможность изменять код, не мешая работе других пользователей

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

Репозиторий - хранилище версий - в нем хранятся все документы вместе с историей их изменения и другой служебной информацией. Commit — отслеживание изменений Рабочая копия - копия проекта, связанная с репозиторием (текущее состояние файлов проекта, основанное на версии из хранилища (обычно на последней) История хранит все изменения в проекте и позволяет при необходимости обратиться к нужным данным.

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

Централизованные VCS (Subversion; CVS; TFS; VAULT; AccuRev): Одно основное хранилище всего проекта Каждый пользователь копирует себе необходимые ему файлы из этого репозитория, изменяет и, затем, добавляет свои изменения обратно Децентрализованные VCS (Git; Mercurial; Bazaar): У каждого пользователя свой вариант (возможно не один) репозитория Присутствует возможность добавлять и забирать изменения из любого репозитория . В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером. В отличие от классических, в распределённых системах контроля версий центральный репозиторий не является обязательным.

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

Сначала создаем и подключаем удаленный репозиторий. Затем по мере изменения проекта отправлять эти изменения на сервер.

5. Опишите порядок работы с общим хранилищем VCS.

Участник проекта (пользователь) перед началом работы посредством определенных команд получает нужную ему версию файлов. После внесения изменений, пользователь размещает новую версию в хранилище. При этом предыдущие версии не удаляются из центрального хранилища и к ним можно вернуться в любой момент.

6. Каковы основные задачи, решаемые инструментальным средством git?

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.

Наиболее часто используемые команды 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 имена_файлов` • удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории): `git rm имена_файлов` • сохранение добавленных изменений: – 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 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.

`git push --all (push origin master/любой branch)`

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

Ветвление («ветка», branch) — один из параллельных участков истории в одном хранилище, исходящих из одной версии (точки ветвления). [3] • Обычно есть

главная ветка (master), или ствол (trunk). • Между ветками, то есть их концами, возможно слияние. Используются для разработки новых функций.

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

Во время работы над проектом так или иначе могут создаваться файлы, которые не требуется добавлять в последствии в репозиторий. Например, временные файлы, создаваемые редакторами, или объектные файлы, создаваемые компиляторами. Можно прописать шаблоны игнорируемых при добавлении в репозиторий типов файлов в файл .gitignore с помощью сервисов.

6 Выводы

В результате выполнения данной лабораторной работы я приобрел необходимые навыки работы с гит, научился созданию репозитория, gpg и ssh ключей, настроил каталог курса и авторизовался в gh.

Список литературы