

Отчет по лабораторной работе №2

Простейший вариант

Чигладзе Майя Владиславовна

Содержание

1	Цель работы	5
2	Порядок выполнения лабораторной работы	6
2.1	Установка программного обеспечения	6
2.2	Задание 2. Базовая настройка git	7
2.3	Создание ключа ssh	9
2.4	Создание ключа pgr	10
2.5	Настройка github	11
2.6	Добавление PGP ключа в GitHub	11
2.7	Настройка автоматических подписей коммитов git	12
2.8	Настройка gh	13
3	Создание репозитория курса на основе шаблона	14
4	Настройка каталога курса	15
5	Контрольные вопросы	17
5.1	Лист вопросов	17
5.2	Лист ответов	17
6	Выводы	21
	Список литературы	22

Список иллюстраций

2.1	Установим git	6
2.2	Установим gh	7
2.3	Имя и email владельца	8
2.4	Настройка и параметры	9
2.5	Ключ SSH	10
2.6	Генерация ключа	11
2.7	Список ключей	12
2.8	Копирование ключа	12
2.9	Подписи комитов	13
2.10	Авторизовывание	13
3.1	Шаблон рабочего пространства	14
3.2	Клонирование репозитория	14
4.1	Результат	16

Список таблиц

1 Цель работы

Изучить идеологию и применение средств контроля версий и освоить умения по работе с git.

2 Порядок выполнения лабораторной работы

2.1 Установка программного обеспечения

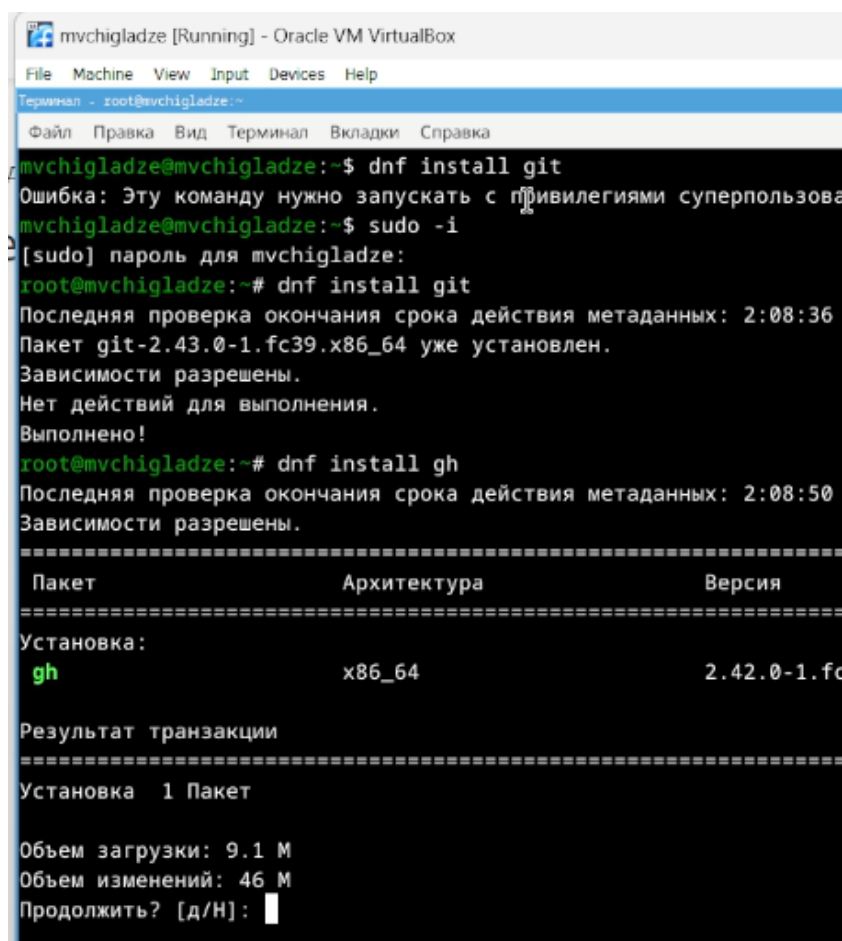
Установила git:

`dnf install git` (рис. [-@fig:001])

Рис. 2.1: Установим git

Установила gh Fedora:

`dnf install gh` (рис. [-@fig:002])



```
mvchigladze@mvchigladze:~$ dnf install git
Ошибка: Эту команду нужно запускать с привилегиями суперпользова
mvchigladze@mvchigladze:~$ sudo -i
[sudo] пароль для mvchigladze:
root@mvchigladze:~# dnf install git
Последняя проверка окончания срока действия метаданных: 2:08:36
Пакет git-2.43.0-1.fc39.x86_64 уже установлен.
Зависимости разрешены.
Нет действий для выполнения.
Выполнено!
root@mvchigladze:~# dnf install gh
Последняя проверка окончания срока действия метаданных: 2:08:50
Зависимости разрешены.
=====
Пакет                Архитектура          Версия
=====
Установка:
gh                    x86_64                2.42.0-1.fc39
=====
Результат транзакции
=====
Установка 1 Пакет

Объем загрузки: 9.1 М
Объем изменений: 46 М
Продолжить? [д/Н]:
```

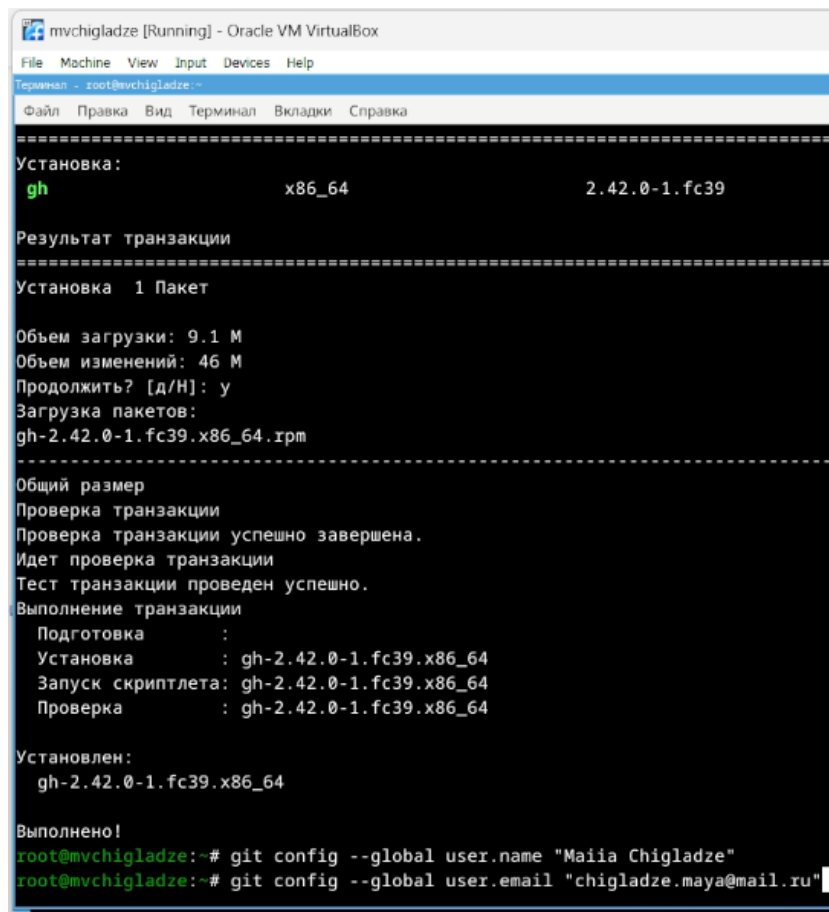
Рис. 2.2: Установим gh

2.2 Задание 2. Базовая настройка git

Зададим имя и email владельца репозитория:

```
git config --global user.name "Name Surname"
```

```
git config --global user.email "work@mail" (рис. [-@fig:003])
```



```
mvchigladze [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Терминал - root@mvchigladze:~
Файл Правка Вид Терминал Вкладки Справка
=====
Установка:
gh x86_64 2.42.0-1.fc39

Результат транзакции
=====
Установка 1 Пакет

Объем загрузки: 9.1 М
Объем изменений: 46 М
Продолжить? [д/Н]: y
Загрузка пакетов:
gh-2.42.0-1.fc39.x86_64.rpm
-----
Общий размер
Проверка транзакции
Проверка транзакции успешно завершена.
Идет проверка транзакции
Тест транзакции проведен успешно.
Выполнение транзакции
Подготовка :
Установка : gh-2.42.0-1.fc39.x86_64
Запуск скрипглета: gh-2.42.0-1.fc39.x86_64
Проверка : gh-2.42.0-1.fc39.x86_64

Установлен:
gh-2.42.0-1.fc39.x86_64

Выполнено!
root@mvchigladze:~# git config --global user.name "Maiia Chigladze"
root@mvchigladze:~# git config --global user.email "chigladze.maya@mail.ru"
```

Рис. 2.3: Имя и email владельца

Настроила utf-8 в выводе сообщений git:

```
git config --global core.quotePath false
```

Настроила верификацию и подписание коммитов git

Задала имя начальной ветки (буду называть её master):

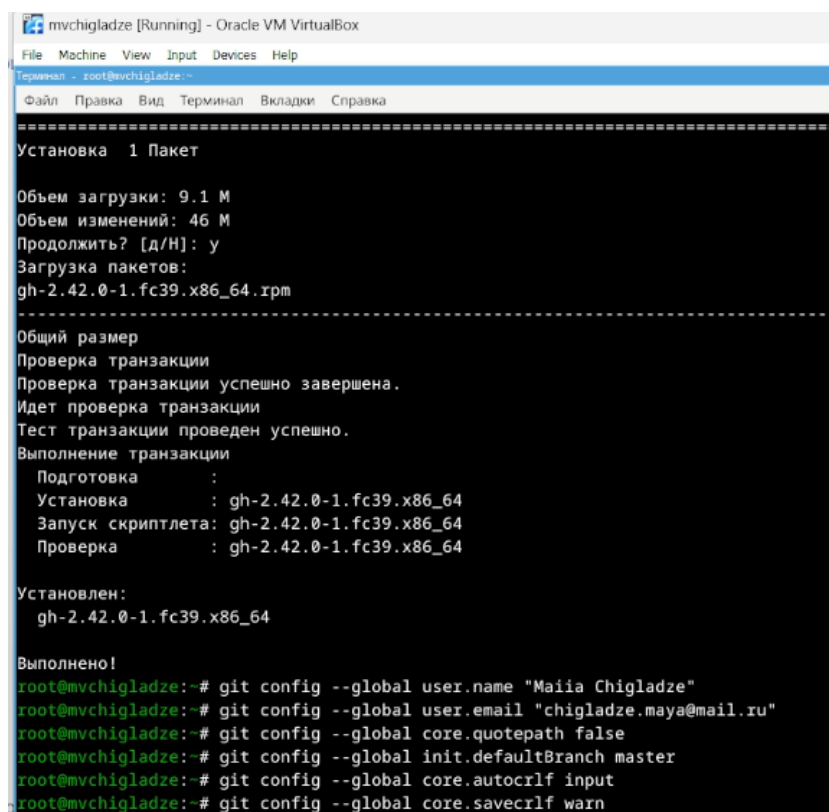
```
git config --global init.defaultBranch master
```

Параметр autocrlf:

```
git config --global core.autocrlf input
```

Параметр safecrlf:


```
git config --global core.safecrlf warn (рис. [-@fig:004])
```



```
mvchigladze [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Терминал - root@mvchigladze:~
Файл Правка Вид Терминал Вкладки Справка
=====
Установка 1 Пакет

Объем загрузки: 9.1 М
Объем изменений: 46 М
Продолжить? [д/н]: у
Загрузка пакетов:
gh-2.42.0-1.fc39.x86_64.rpm
=====
Общий размер
Проверка транзакции
Проверка транзакции успешно завершена.
Идет проверка транзакции
Тест транзакции проведен успешно.
Выполнение транзакции
  Подготовка      :
  Установка       : gh-2.42.0-1.fc39.x86_64
  Запуск скрипта  : gh-2.42.0-1.fc39.x86_64
  Проверка        : gh-2.42.0-1.fc39.x86_64

Установлен:
gh-2.42.0-1.fc39.x86_64

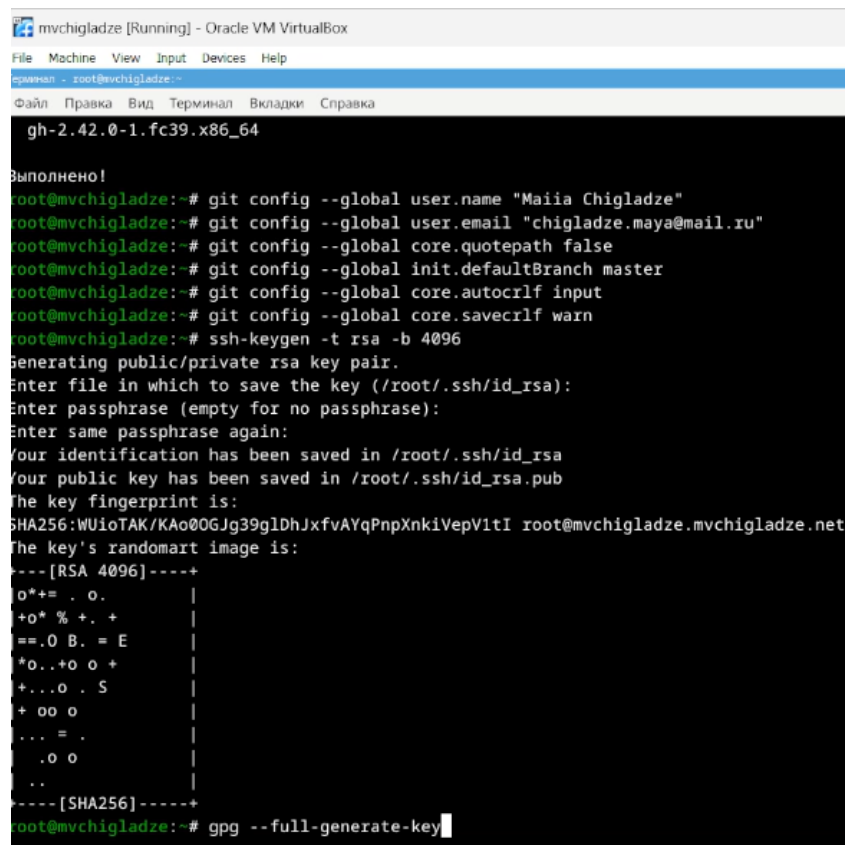
Выполнено!
root@mvchigladze:~# git config --global user.name "Maiia Chigladze"
root@mvchigladze:~# git config --global user.email "chigladze.maya@mail.ru"
root@mvchigladze:~# git config --global core.quotepath false
root@mvchigladze:~# git config --global init.defaultBranch master
root@mvchigladze:~# git config --global core.autocrlf input
root@mvchigladze:~# git config --global core.safecrlf warn
```

Рис. 2.4: Настройка и параметры

2.3 Создание ключа ssh

По алгоритму rsa с ключём размером 4096 бит:

```
ssh-keygen -t rsa -b 4096 (рис. [-@fig:005])
```



```
mvchigladze [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
терминал - root@mvchigladze:~
Файл Правка Вид Терминал Вкладки Справка
gh-2.42.0-1.fc39.x86_64

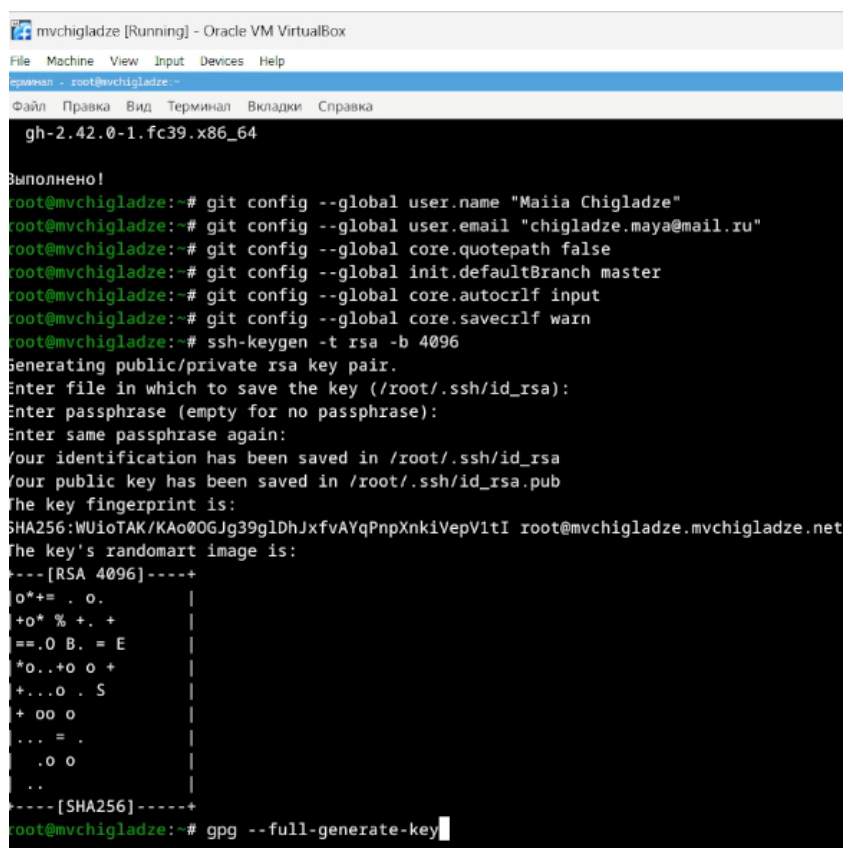
Выполнено!
root@mvchigladze:~# git config --global user.name "Maiia Chigladze"
root@mvchigladze:~# git config --global user.email "chigladze.maya@mail.ru"
root@mvchigladze:~# git config --global core.quotepath false
root@mvchigladze:~# git config --global init.defaultBranch master
root@mvchigladze:~# git config --global core.autocrlf input
root@mvchigladze:~# git config --global core.safecrlf warn
root@mvchigladze:~# ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:WUioTAK/KAo00GJg39gldhJxfvAYqPnpXnkiVepVtI root@mvchigladze.mvchigladze.net
The key's randomart image is:
+--[RSA 4096]-----+
|  o*+ = . o. |
| +o* % +. + |
| ==.O B. = E |
| *O..+o o + |
| +...o . S |
| + oo o |
| ... = . |
| .o o |
| .. |
+-----[SHA256]-----+
root@mvchigladze:~# gpg --full-generate-key
```

Рис. 2.5: Ключ SSH

2.4 Создание ключа ргр

Генерирую ключ

gpg --full-generate-key (рис. [-@fig:006])



```
mvchigladze [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
mvchigladze - root@mvchigladze:~
Файл Правка Вид Терминал Вкладки Справка
gh-2.42.0-1.fc39.x86_64

Выполнено!
root@mvchigladze:~# git config --global user.name "Maiia Chigladze"
root@mvchigladze:~# git config --global user.email "chigladze.maya@mail.ru"
root@mvchigladze:~# git config --global core.quotepath false
root@mvchigladze:~# git config --global init.defaultBranch master
root@mvchigladze:~# git config --global core.autocrlf input
root@mvchigladze:~# git config --global core.safecrlf warn
root@mvchigladze:~# ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:WUioTAK/KAo0OGJg39gldhJxfvAYqPnpXnkiVepVtI root@mvchigladze.mvchigladze.net
The key's randomart image is:
----[RSA 4096]-----+
|
|o*+ . o. |
|+o* % +. + |
|==.O B. = E |
|*O..+o o + |
|+...o . S |
|+ oo o |
|... = . |
|.o o |
|.. |
|-----[SHA256]-----+
root@mvchigladze:~# gpg --full-generate-key
```

Рис. 2.6: Генерация ключа

2.5 Настройка github

Так как аккаунт в гитхаб у меня есть, этот пункт пропускаю

2.6 Добавление PGP ключа в GitHub

Вывожу список ключей и копируем отпечаток приватного ключа:

`gpg --list-secret-keys --keyid-format LONG` (рис. [-@fig:007])

```

root@mvchigladze:~# gpg --list-secret-keys --keyid-format LONG
gpg: проверка таблицы доверия
gpg: marginals needed: 3 completes needed: 1 trust model: pgp
gpg: глубина: 0 достоверных: 1 подписанных: 0 доверие: 0-, 0q, 0n,
[keyboard]
-----
sec   rsa4096/3DB44C0E416EE189 2024-02-12 [SC]
      D5F7ED7F6B7BE27723050F0F3DB44C0E416EE189
uid           [ абсолютно ] Maiia (das) <chigladze.maya@mail.ru>
ssb   rsa4096/148E3203B607B3E7 2024-02-12 [E]

root@mvchigladze:~#

```

Рис. 2.7: Список ключей

Копирую мой сгенерированный PGP ключ в буфер обмена:

tgpg --armor --export <PGP Fingerprint> | xclip -sel clip (рис. [-@fig:008])

```

Установлен:
  xclip-0.13-20.git11cba61.fc39.x86_64

Выполнено!
root@mvchigladze:~# gpg --armor --export <PGP Fingerprint> | xclip -sel clip
-bash: синтаксическая ошибка рядом с неожиданным маркером «|»
root@mvchigladze:~# gpg --armor --export-secret-keys 3DB44C0E416EE189 | xclip
root@mvchigladze:~# gpg --armor --export-secret-keys 3DB44C0E416EE189 | xclip
root@mvchigladze:~# gpg --armor --export <PGP Fingerprint> | xclip -sel clip
-bash: синтаксическая ошибка рядом с неожиданным маркером «|»
root@mvchigladze:~# gpg --armor --export-secret-keys 3DB44C0E416EE189 | xclip -s
root@mvchigladze:~# gpg --armor --export <PGP Fingerprint>
-bash: синтаксическая ошибка рядом с неожиданным маркером «newline»
root@mvchigladze:~# gpg --armor --export 3DB44C0E416EE189 | xclip -sel clip

```

Рис. 2.8: Копирование ключа

Перехожу в настройки GitHub (<https://github.com/settings/keys>), нажмите на кнопку New GPG key и вставьте полученный ключ в поле ввода.

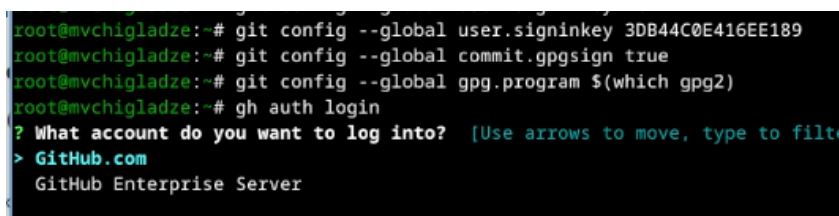
2.7 Настройка автоматических подписей коммитов git

Используя введённый email, указываю Git применять его при подписи коммитов:

```
git config --global user.signingkey <PGP Fingerprint>
```

```
git config --global commit.gpgsign true
```

```
git config --global gpg.program $(which gpg2) (рис. [-@fig:009])
```



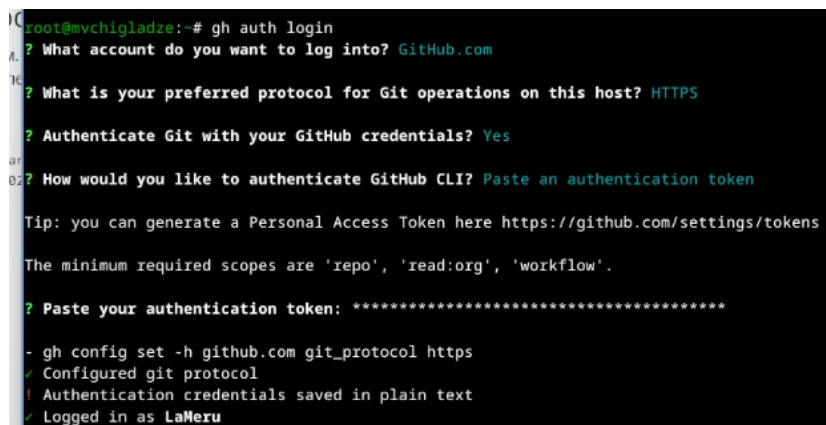
```
root@mvchigladze:~# git config --global user.signinkey 3DB44C0E416EE189
root@mvchigladze:~# git config --global commit.gpgsign true
root@mvchigladze:~# git config --global gpg.program $(which gpg2)
root@mvchigladze:~# gh auth login
? What account do you want to log into? [Use arrows to move, type to filter]
> GitHub.com
  GitHub Enterprise Server
```

Рис. 2.9: Подписи комитов

2.8 Настройка gh

Для начала необходимо авторизоваться

```
gh auth login (рис. [-@fig:010])
```



```
root@mvchigladze:~# gh auth login
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations on this host? HTTPS
? Authenticate Git with your GitHub credentials? Yes
? How would you like to authenticate GitHub CLI? Paste an authentication token

Tip: you can generate a Personal Access Token here https://github.com/settings/tokens
The minimum required scopes are 'repo', 'read:org', 'workflow'.

? Paste your authentication token: *****

- gh config set -h github.com git_protocol https
✓ Configured git protocol
! Authentication credentials saved in plain text
✓ Logged in as LaMeru
```

Рис. 2.10: Авторизовывание

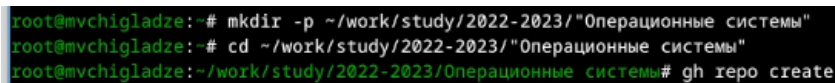
3 Сознание репозитория курса на основе шаблона

Необходимо создать шаблон рабочего пространства.

```
mkdir -p ~/work/study/2022-2023/"Операционные системы"
```

```
cd ~/work/study/2022-2023/"Операционные системы"
```

```
gh repo create study_2022-2023_os-intro --template=yamadharm/course-  
directory-student-template --public (рис. [-@fig:011])
```

A screenshot of a terminal window showing three commands being executed in sequence. The first command creates a directory, the second changes to that directory, and the third creates a new GitHub repository using a template.

```
root@mvchigladze:~# mkdir -p ~/work/study/2022-2023/"Операционные системы"  
root@mvchigladze:~# cd ~/work/study/2022-2023/"Операционные системы"  
root@mvchigladze:~/work/study/2022-2023/Операционные системы# gh repo create
```

Рис. 3.1: Шаблон рабочего пространства

```
git clone --recursive git@github.com:<owner>/study_2022-2023_os-  
intro.git os-intro (рис. [-@fig:012])
```

Клонирование репозитория

Рис. 3.2: Клонирование репозитория

4 Настройка каталога курса

Перешла в каталог курса:

```
cd ~/work/study/2022-2023/"Операционные системы"/os-intro
```

Удалила лишние файлы:

```
rm package.json
```

Создала необходимые каталоги:

```
echo os-intro > COURSE
```

```
make
```

Отправила файлы на сервер:

```
git add .
```

```
git commit -am 'feat(main): make course structure'
```

```
git push
```

Результат манипуляций виден в самом репозитории (рис. [4.1])















 LaMeru Lab1	
 config	Initial commit
 lab1	Lab1
 template	Initial commit
 .gitattributes	Initial commit
 .gitignore	Initial commit
 .gitmodules	Initial commit
 CHANGELOG.md	Initial commit
 COURSE	feat(main): make course s
 LICENSE	Initial commit
 Makefile	Initial commit
 README.en.md	Initial commit
 README.git-flow.md	Initial commit
 README.md	Initial commit

Рис. 4.1: Результат

5 Контрольные вопросы

5.1 Лист вопросов

- 1) Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются?
- 2) Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.
- 3) Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида.
- 4) Опишите действия с VCS при единоличной работе с хранилищем.
- 5) Опишите порядок работы с общим хранилищем VCS.
- 6) Каковы основные задачи, решаемые инструментальным средством git?
- 7) Назовите и дайте краткую характеристику командам git.
- 8) Приведите примеры использования при работе с локальным и удалённым репозиториями.
- 9) Что такое и зачем могут быть нужны ветви (branches)?
- 10) Как и зачем можно игнорировать некоторые файлы при commit?

5.2 Лист ответов

- 1) Система управления версиями (также используется определение «система контроля версий», от англ. version control system, VCS или revision control system) — программное обеспечение для облегчения работы с изменяющей-

ся информацией.

- 2) Основные понятия VCS: Хранилище (repository) - это место, где хранятся все версии файла или проекта. Обычно это сервер, на котором установлен специальный сервер VCS. Commit (фиксация) - это процесс добавления новой версии файла в хранилище. Разработчик делает коммит, когда вносит изменения в файл или набор файлов, и отправляет их на сервер VCS для сохранения. История (history) - это список всех коммитов, которые были сделаны в проекте. Каждый коммит содержит информацию о том, какие изменения были внесены, кем они были внесены и когда. История позволяет разработчикам видеть, какие изменения были сделаны и когда, а также откатываться к предыдущим версиям в случае необходимости. Рабочая копия (working copy) - это локальная копия проекта, с которой разработчик работает на своем компьютере. Рабочая копия синхронизируется с хранилищем, то есть разработчик может делать коммиты, которые затем отправляются
- 3) Централизованные VCS: вся информация о проекте хранится на центральном сервере, и все изменения в проекте делаются через этот сервер. Примеры: Git, Mercurial. Децентрализованные VCS (DVCS): каждый разработчик имеет свою собственную копию проекта, и изменения могут делаться независимо на каждом компьютере. Затем изменения объединяются, и получается итоговая версия проекта. Примеры: Git (с использованием нескольких репозиторий), Mercurial (с использованием концепции "пулл-реквестов").
- 4) При единоличной работе над проектом действия с VCS включают: Создание репозитория на сервере VCS (если это еще не было сделано). Клонирование репозитория на свой компьютер с помощью команды `git clone`. Внесение изменений в проект на своем компьютере (рабочая копия). Коммит изменений в репозиторий с помощью команды `git commit`. Обновление рабочей копии с сервера с помощью команды `git pull`. Просмотр истории изменений с помощью команды `git log`. Эти действия позволяют сохранять историю изменений проекта, возвращаться к предыдущим версиям и

работать над проектом без конфликтов. 5) Для работы с общим хранилищем необходимо выполнить следующие шаги: Создать репозиторий на сервере Отправить запрос на включение в проект (pull request) Принять запрос и выполнить слияние изменений 6) Основные задачи, решаемые git: Управление версиями: позволяет хранить историю изменений файлов и при необходимости возвращаться к предыдущим версиям. Совместная работа: позволяет нескольким разработчикам работать над проектом одновременно, не мешая друг другу. Отслеживание изменений: позволяет видеть, какие изменения были внесены в проект и кем. Управление ветками: позволяет создавать ветки для параллельной разработки и затем объединять их в основную ветку проекта. Мерджинг (слияние) веток: позволяет объединять изменения из разных веток в одну. 7) Основные команды Git: git status: показывает текущее состояние репозитория; git add: добавляет файлы в индекс (stage); git commit: создает коммит (измененный набор файлов) и записывает его в репозиторий; git branch: создает новую ветку или переключается на существующую; git merge: объединяет две ветки в одну; git push: отправляет коммиты на удаленный репозиторий; git pull: обновляет рабочую копию репозитория с удаленного сервера; git diff: показывает различия между двумя коммитами; git log: выводит историю коммитов; git reset: сбрасывает состояние индекса или рабочего каталога до определенного состояния. 8) При работе с локальным репозиторием: Создание репозитория: git init. Клонирование репозитория: git clone . Добавление файлов в индекс: git add . Коммит изменений: git commit -m "message". Просмотр истории коммитов: git log При работе с удаленным репозиторием (на примере GitHub): Регистрация на сервисе: <https://github.com>. Создание нового репозитория или клонирование существующего: https://github.com/your_username/your_repo_name. Отправка запроса на включение в проект: https://github.com/your_username/your_repo_name/compare. Принять запрос и выполнить слияние: [19](https://help.github.com/articles/merging-</p></div><div data-bbox=)

a-pull-request/ 9) Ветви в системе контроля версий позволяют разработчикам работать параллельно над различными частями проекта без риска повредить основную ветку проекта. Они могут создаваться для реализации новых функций, исправления ошибок или проведения рефакторинга кода. Когда разработчик создает ветку, он получает копию проекта со всеми его файлами и папками. Затем он может вносить изменения в эту ветку, не затрагивая основную ветку проекта. После того, как работа над веткой завершена, разработчик может объединить свои изменения с основной веткой проекта или предложить другим разработчикам оценить и принять эти изменения.

- 5) Можно использовать команду `.gitignore` чтобы игнорировать файлы при коммите. Это может быть полезно если вы не хотите чтобы в репозитории были видны файлы например с настройками вашего редактора или временные файлы. Вы можете создать файл `.gitignore` и добавить в него названия файлов или папок которые нужно игнорировать. Затем нужно добавить этот файл в репозиторий что бы он учитывался при коммите.

6 Выводы

В ходе лабораторной работы, я изучила идеологию и применение средств контроля версий и освоить умения по работе с git.

Список литературы