

# **Отчёт по лабораторной работе №2**

**Первоначальна настройка git**

Сагдеров Камал

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>8</b>
<b>5</b>	<b>Выводы</b>	<b>19</b>

## Список иллюстраций

4.1	Создание учетной записи . . . . .	9
4.2	установка git . . . . .	9
4.3	установка gh . . . . .	10
4.4	Задаем имя и email владельца репозитория, настраивая utf-8 . .	10
4.5	Настройка верификацию и подписание коммитов git . . . . .	10
4.6	параметры autocrlf и safecrlf . . . . .	10
4.7	по алгоритму rsa . . . . .	11
4.8	по алгоритму ed25519 . . . . .	11
4.9	Генерация ключей . . . . .	12
4.10	Вывод списка ключей и копирование отпечатка приватного ключа	12
4.11	Добавление PGP ключа в GitHub . . . . .	12
4.12	Создание GPG ключа . . . . .	13
4.13	Настройка автоматических подписей коммитов git . . . . .	13
4.14	Настройка gh . . . . .	13
4.15	создание репозитория курса . . . . .	14
4.16	создание репозитория курса . . . . .	14
4.17	Создание каталога курса . . . . .	15

## **Список таблиц**

# 1 Цель работы

1. Изучить идеологию и применение средств контроля версий.
2. Освоить 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

В результате выполнения лабораторной работы №2, мы познакомились с системой контроля версий. 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. Системы контроля версий имеют возможность отслеживания и разрешения конфликтов, которые могут возникнуть при работе нескольких человек над одним файлом. Системы контроля версий также могут обеспечивать дополнительные, более гибкие функциональные возможности. 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

№1 Установка программного обеспечения Создаем учетную запись Github и заполняем основные данные. После устанавливаем программное обеспечение, устанавливаем gh. Затем переходим к базовой настройке git, где задаю свои данные (имя и email) репозитория. Настраиваю utf-8 в выводе сообщений git. Задаем имя начальной ветки, параметры (autocrlf, safecrlf) (рис. 4.1).



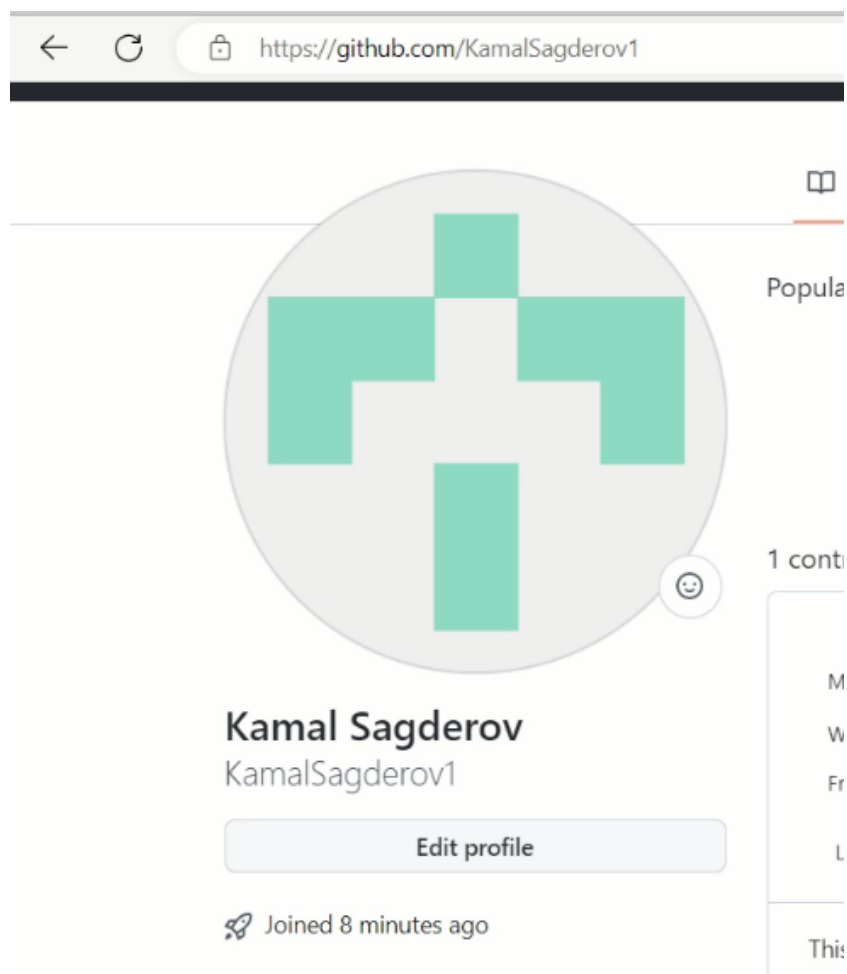


Рис. 4.1: Создание учетной записи

```
[root@fedora ~]# dnf install git
Последняя проверка окончания срока действия метаданных: 1:44:39 назад, Сб 18 фев 2023 03:35:52.
Пакет git-2.39.2-1.fc37.x86_64 уже установлен.
Зависимости разрешены.
Отсутствуют действия для выполнения.
Выполнено!
```

Рис. 4.2: установка git

```
[root@fedora ~]# dnf install gh
Последняя проверка окончания срока действия метаданных: 1:47:24 назад, Сб 18 фев
2023 03:35:52.
Зависимости разрешены.
=====
Пакет      Архитектура  Версия      Резепозиторий  Размер
=====
Установка:
gh         x86_64       2.22.1-1.fc37  updates       8.3 М
=====
Результат транзакции
=====
Установка 1 Пакет
Объем загрузки: 8.3 М
Объем изменений: 42 М
```

Рис. 4.3: установка gh

№2 Базовая настройка git Задаем данные владельца, настраиваем utf-8 в выводе сообщений git, верификацию и подписание коммитов git и задаем имя начальной ветки. (рис. 4.4).

```
[root@fedora ~]# git config --global user.name "Kamal Sagderov"
[root@fedora ~]# git config --global user.email "sagderovk@gmail.com"
[root@fedora ~]# git config --global core.quotePath false
```

Рис. 4.4: Задаем имя и email владельца репозитория, настраиваем utf-8

```
[root@fedora ~]# git config --global init.defaultBranch master
[root@fedora ~]# git config --global core.autocrlf input
[root@fedora ~]# git config --global core.safecrlf warn
[root@fedora ~]#
```

Рис. 4.5: Настройка верификацию и подписание коммитов git

```
[root@fedora ~]# git config --global core.autocrlf input
[root@fedora ~]# git config --global core.safecrlf warn
[root@fedora ~]# git config --global core.safecrlf warn
[root@fedora ~]#
```

Рис. 4.6: параметры autocrlf и safecrlf

№3 Создание ключей ssh и pgr

```

[root@fedora ~]# ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Created directory '/root/.ssh'.
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:jxrUb53U2cI310DfQf/YCRYoqFBg0uqLRp8Fbn5v0uw root@fedora
The key's randomart image is:
+---[RSA 4096]-----+
| ..o      . |
| oo.      .o|
| o ..     .o|
| o +.. .. .o+|
| o *.o ..S. .o=|=|
| + +.o. +.o .ooo|
| .  o.o . + oo . |
|   o = . . .oo|
|   .E=   .o|
+-----[SHA256]-----+
[root@fedora ~]#

```

Рис. 4.7: по алгоритму rsa

```

[root@fedora ~]# ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/root/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_ed25519
Your public key has been saved in /root/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:/sziyn4aUKs6aq/lRc+9hH31GcAUcLM2MfhPmaELqBw root@fedora
The key's randomart image is:
+---[ED25519 256]---+
|      .oB. |
|      .+ =. |
|      . . *. +|
|      . E . ..oo+|
|      ..o S  ..+. |
|      .oo++   ....o|
|      ....+.+ .  o |
|      .+...  .++o  |
|      oo++  .==.o+  |
+-----[SHA256]-----+
[root@fedora ~]#

```

Рис. 4.8: по алгоритму ed25519

```

открытый и секретный ключи созданы и подписаны.

pub  rsa4096 2023-02-18 [SC]
    8D5950CFEDC4589F1097309A2BCD5AE3D096C713
uid                               Kamal <sagderovk@gmail.com>
sub   rsa4096 2023-02-18 [E]

[root@fedora ~]#

```

Рис. 4.9: Генерация ключей

№4 Экспорт ключа Выводим список ключей и копируем отпечаток приватного ключа, а также экспортируем ключ в формате ASCII по его отпечатку (рис. 4.10).

```

[root@fedora ~]# 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, 0m, 0f
, 1u
/root/.gnupg/pubring.kbx
-----
sec  rsa4096/2BCD5AE3D096C713 2023-02-18 [SC]
    8D5950CFEDC4589F1097309A2BCD5AE3D096C713
uid      [ абсолютно ] Kamal <sagderovk@gmail.com>
ssb  rsa4096/4128D9CE2EC2629E 2023-02-18 [E]

```

Рис. 4.10: Вывод списка ключей и копирование отпечатка приватного ключа

№5 Добавление PGP ключа в GitHub

```

[root@fedora ~]# gpg --armor --export PGP Fingerprint | xclip -sel clip
gpg: Внимание: нечего экспортировать
bash: xclip: команда не найдена...
Установить пакет «xclip», предоставляющий команду «xclip»? [N/y] y

* Ожидание в очереди...
Следующие пакеты должны быть установлены:
xclip-0.13-18.git11cba61.fc37.x86_64 Command line clipboard grabber
Продолжить с этими изменениями? [N/y] y

```

Рис. 4.11: Добавление PGP ключа в GitHub

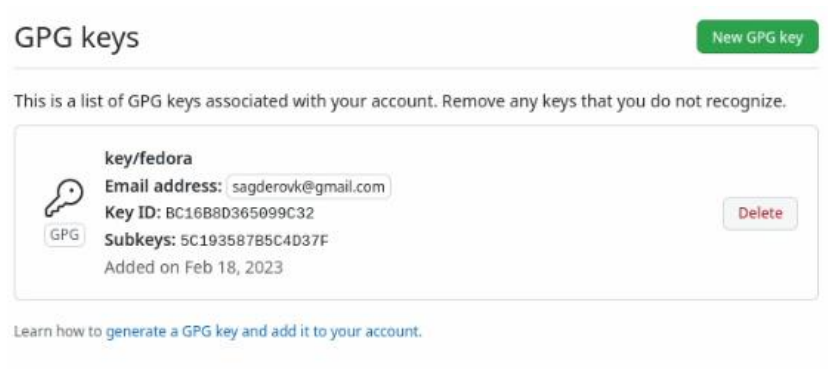


Рис. 4.12: Создание GPG ключа

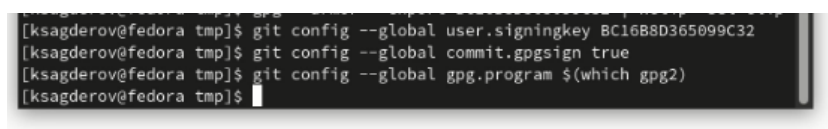


Рис. 4.13: Настройка автоматических подписей коммитов git

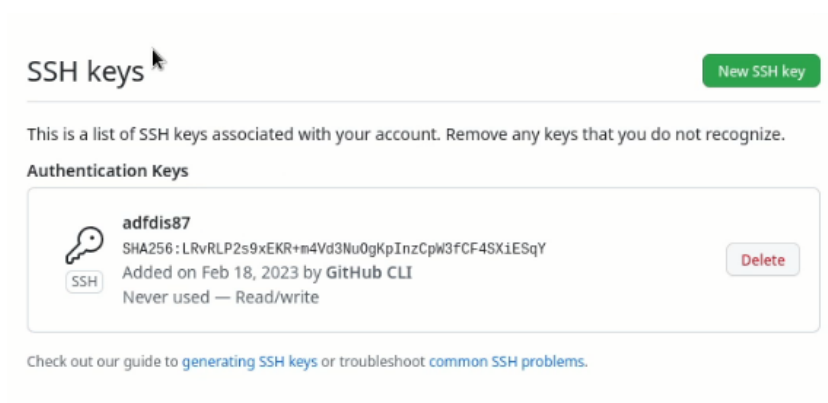


Рис. 4.14: Настройка gh

## №6 Создание репозитория курса на основе шаблона

```
Logged in as KamalSagderov1
[ksagderov@fedora tmp]$ mkdir -p ~/work/study/2022-2023/"Операционные системы"
[ksagderov@fedora tmp]$ d ~/work/study/2022-2023/"Операционные системы"
bash: d: команда не найдена...
[ksagderov@fedora tmp]$ mkdir -p ~/work/study/2022-2023/"Операционные системы"
[ksagderov@fedora tmp]$ cd ~/work/study/2022-2023/"Операционные системы"
[ksagderov@fedora Операционные системы]$ gh repo create study_2022-2023_os-intro
--template=yamadharma/course-directory-student-template --public
✓ Created repository KamalSagderov1/study_2022-2023_os-intro on GitHub
[ksagderov@fedora Операционные системы]$ git clone --recursive git@github.com:Ka
wner>/study_2022-2023_os-intro.git os-intro
bash: owner: Нет такого файла или каталога
[ksagderov@fedora Операционные системы]$
[ksagderov@fedora Операционные системы]$ git clone --recursive git@github.com:Ka
malSagderov1/study_2022-2023_os-intro.git os-intro
Клонирование в «os-intro»...
The authenticity of host 'github.com (140.82.121.3)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvCoQu.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
```

Рис. 4.15: создание репозитория курса

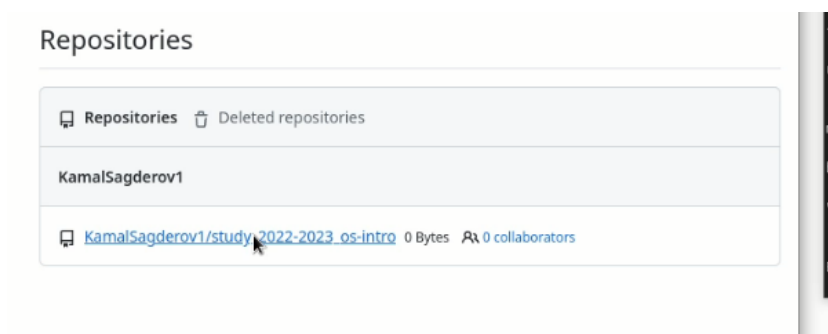


Рис. 4.16: создание репозитория курса

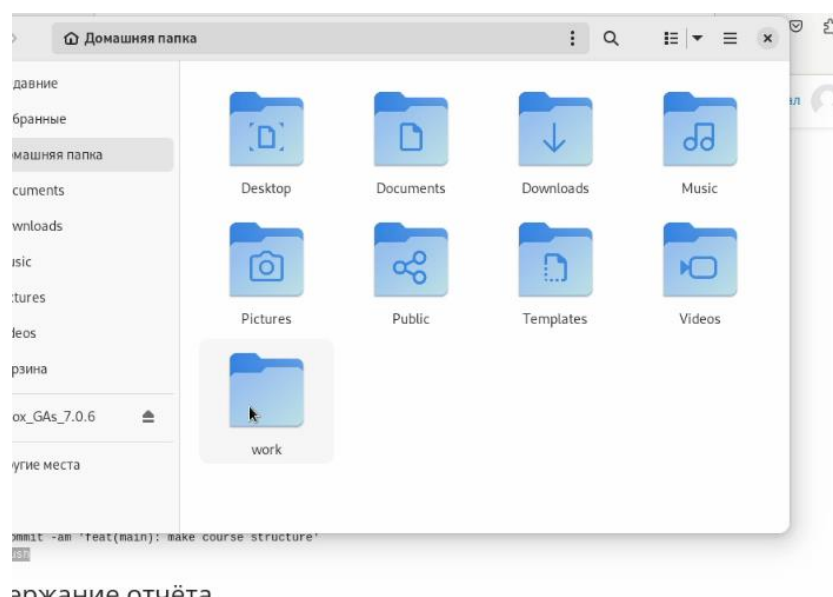


Рис. 4.17: Создание каталога курса

№7 Контрольные вопросы 1.Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются? Система управления версиями — 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.

Система контроля версий применяется при работе : - нескольких человек над одним проектом. - 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

2.Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия. repository - хранилище файлов, ссылок на изменения в файлах, в нем хранятся все документы вместе с историей их изменения и другой служебной информацией. commit - отслеживание изменений, сохраняет разницу в изменениях История - хранит все изменения в проекте и позволяет

при необходимости обратиться к нужным данным. Рабочая копия- «снимок» содержимого репозитория, плюс некоторая служебная информация. Изменив содержимое рабочей копии, разработчик фиксирует сделанные изменения в репозитории. Как правило, фиксация сопровождается небольшим текстовым комментарием, описывающим сделанные изменения.

3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида. Централизованные VCS - это системы контроля версий, которые имеют один основной удаленный репозиторий, к которому все пользователи имеют доступ. Они базируются на предположении, что репозиторий является истинным и достоверным. Примерами таких VCS являются Subversion и CVS.

Децентрализованные VCS - это системы контроля версий, которые не имеют основного удаленного репозитория. Вместо этого каждый пользователь имеет локальную копию репозитория, которая может быть синхронизирована с копиями

1. Централизованные:

- Subversion (SVN)
- CVS
- Perforce

2. Децентрализованные:

- Git
- Mercurial
- Bazaar

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

- Создание локального хранилища: Используя команду `git init`, можно создать локальное хранилище.
- Клонирование хранилища: Если хранилище уже существует в удаленном репозитории, можно использовать команду `git clone` для клонирования хранилища на локальную машину.



- Добавление файлов: Используя команду `git add`, можно добавлять файлы в хранилище.
- Коммитирование изменений: Используя команду `git commit`, можно коммитить изменения в хранилище.
- Просмотр истории коммитов: Используя команду `git log`

5. Опишите порядок работы с общим хранилищем VCS.

6. Начните с создания репозитория VCS. Это может быть на сервере или на локальном компьютере.

7. Добавьте в репозиторий все необходимые файлы.

8. Создайте коммит для хранения изменений.

9. Обменяйтесь изменениями с другими участниками проекта с помощью пулл-реквестов.

10. Объедините изменения в одну версию.

11. Обновите код на локальной машине.

12. Проверьте изменения и подтвердите их.

13. Отправьте изменения в общее хранилище.

6. Каковы основные задачи, решаемые инструментальным средством `git`? 1 - хранить информацию о всех изменениях в вашем коде, начиная с самой первой строчки 2 - обеспечение удобства командной работы над кодом

7. Назовите и дайте краткую характеристику командам `git`.

8. `git init` - Инициализация нового репозитория `Git` в текущем каталоге.

9. `git clone` - Клонирование удаленного репозитория в текущий каталог.

10. `git add` - Добавление файлов в репозиторий для последующего коммита.
11. `git commit` - Сохранение изменений в репозитории.
12. `git push` - Отправка изменений из локального репозитория в удаленный репозиторий.
13. `git pull` - Загрузка изменений из удаленного репозитория в локальный репозиторий.
14. `git status` - Просмотр текущего состояния репозитория.
15. `git diff` - Просмотр изменений, внесенных
16. Приведите примеры использования при работе с локальным и удалённым репозиториями. `git push - all(push origin master/любой branch)` \$ `git fetch` - загрузка изменений с удаленного репозитория \$ `git push` - отправка изменений в удаленный репозиторий
17. что такое и зачем могут быть нужны ветви (branches)? Ветви (branches) — это варианты исходного кода, которые могут быть использованы для разработки различных версий программы. Они позволяют разработчикам и тестировщикам работать над одним проектом одновременно, не пересекаясь друг с другом. Кроме того, они позволяют отслеживать изменения в исходном коде и настраивать процесс разработки.
18. 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.

## 5 Выводы

В результате выполнения лабораторной работы я научился создавать учетную запись GitHub, устанавливать и работать с программным обеспечением. Также научился базовой настройке git и работать с серверами репозиториями, добавлять SSH/PGP ключи в свою учетную запись Github