

Отчет по лабораторной работе №2

Татур Стефан Андреевич

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6

Список иллюстраций

2.1	Название рисунка	6
2.2	Название рисунка	7
2.3	Название рисунка	8
2.4	Название рисунка	9
2.5	Название рисунка	10
2.6	Название рисунка	11
2.7	Название рисунка	12
2.8	Название рисунка	12

Список таблиц

1 Цель работы

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

2 Выполнение лабораторной работы

1. Создаём учётную запись на github.

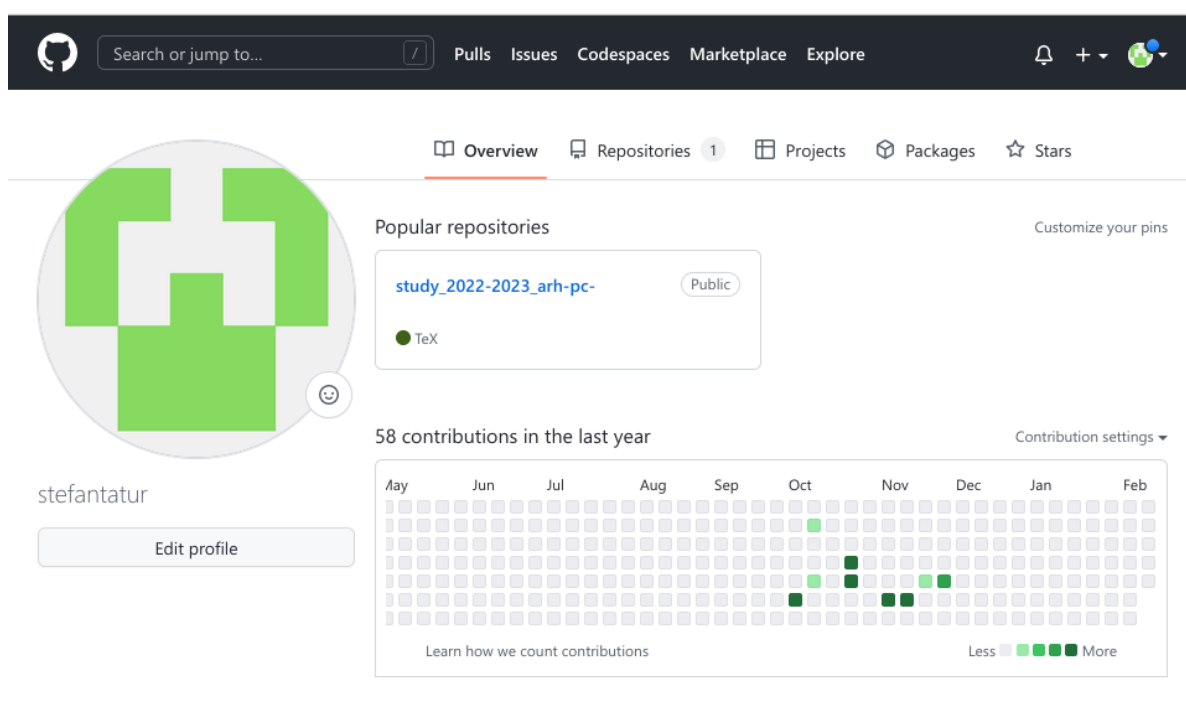


Рис. 2.1: Название рисунка

2. Сделаем предварительную конфигурацию, указав имя и email владельца репозитория с помощью `git config --global user.name "Имя Фамилия"`, `git config --global user.email "work@mail"`.

После этого создаём новый ключ на github (команда `ssh-keygen -C "Stefantatur stephantatur@gmail.com"` и привязываем его к компьютеру через консоль. После

этого, скопировав из локальной консоли ключ в буфер обмена, вставляем ключ в появившееся на сайте поле.

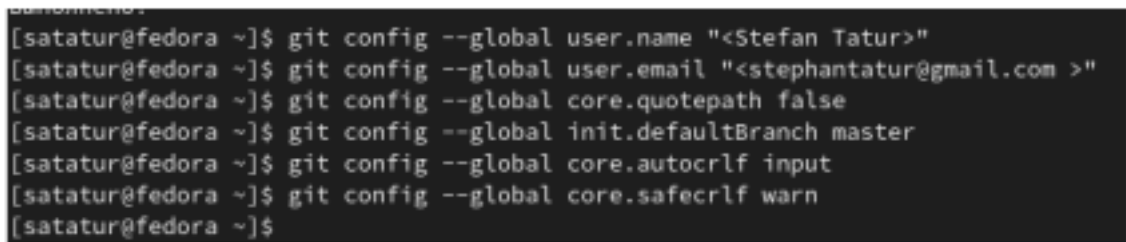
A screenshot of a terminal window with a dark background and light-colored text. The prompt is [satatur@fedora ~]. The commands and their outputs are as follows:
[satatur@fedora ~]\$ git config --global user.name "<Stefan Tatur>"
[satatur@fedora ~]\$ git config --global user.email "<stephantatur@gmail.com >"
[satatur@fedora ~]\$ git config --global core.quotepath false
[satatur@fedora ~]\$ git config --global init.defaultBranch master
[satatur@fedora ~]\$ git config --global core.autocrlf input
[satatur@fedora ~]\$ git config --global core.safecrlf warn
[satatur@fedora ~]\$

Рис. 2.2: Название рисунка

3. Приступаем к базовой настройке git. Зададим имя и email владельца репозитория: `git config --global user.name "Name Surname"`, `git config --global user.email "work@mail"`. Настроим utf-8 в выводе сообщений git: `git config --global core.quotepath false`. Настроим верификацию и подписание коммитов git. Зададим имя начальной ветки: `git config --global init.defaultBranch master`. Параметр `autocrlf`: `git config --global core.autocrlf input`. Параметр `safecrlf`: `git config --global core.safecrlf warn`.

```

[saatur@fedora ~]$ ssh-keygen -C "Стефан Тарп <stephantatur@gmail.com>"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/satur/.ssh/id_rsa): ~/.ssh/
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Saving key "~/.ssh/" failed: No such file or directory
[saatur@fedora ~]$ ssh-keygen -C "Стефан Тарп <stephantatur@gmail.com>"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/satur/.ssh/id_rsa):
Created directory '/home/satur/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/satur/.ssh/id_rsa
Your public key has been saved in /home/satur/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:Jql/zKhIC2afrgVy9NoWx5eu4EItDfcehMDNiQWvAPs Стефан Тарп <stephantatur@gmail.com>
The key's randomart image is:
+----[RSA 3072]-----+
|...*...|
|..+.+|
|o....|
|+.00...|
|.E=00+ S|
|00+000*|
|.=.=....+|
|o.*00.o.+|
|.=*00...|
+----[SHA256]-----+
[saatur@fedora ~]$

```

Рис. 2.3: Название рисунка

4. Готовый Ssh-ключ на я github.

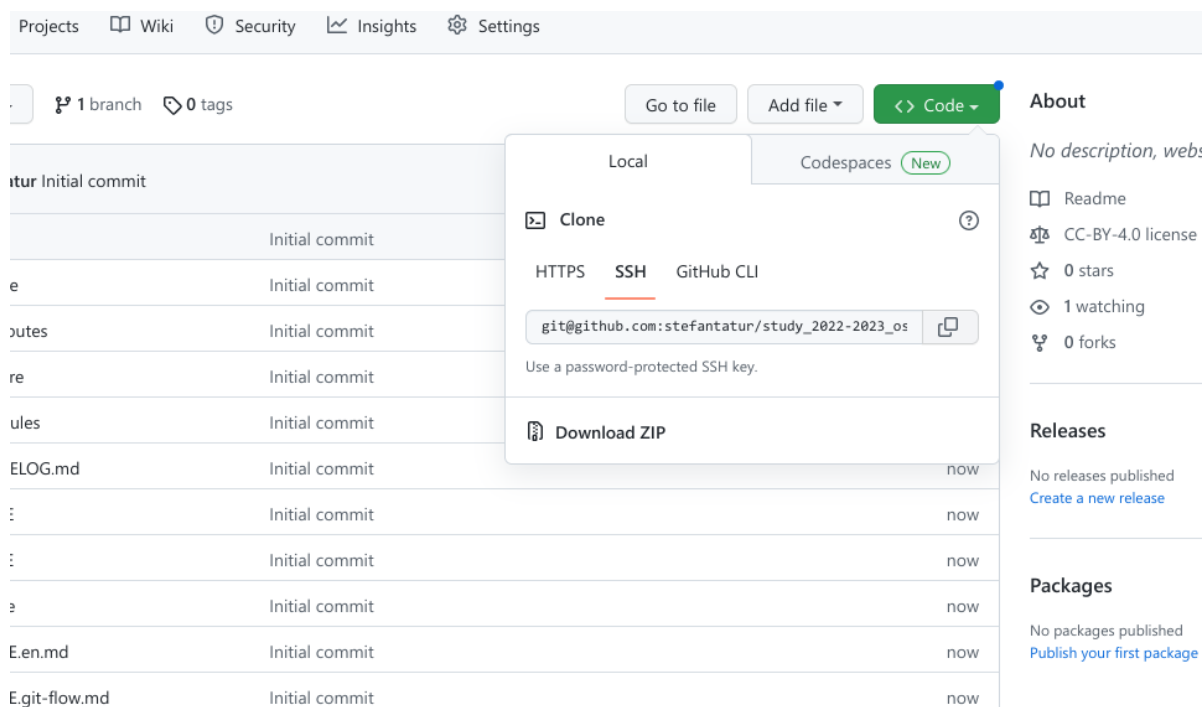


Рис. 2.4: Название рисунка

4. Создаём ключ `gpg`: `gpg --full-generate-key`. Затем настраиваем:
 - Тип RSA and RSA; • размер 4096; • срок действия; значение по умолчанию— 0 (срок действия не истекает никогда). – • Имя • Адрес электронной почты.

```

satatur@dk6n64 ~ $ gpg --full-generate-key
gpg (GnuPG) 2.2.40; Copyright (C) 2022 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 и RSA (по умолчанию)
  (2) DSA и Elgamal
  (3) DSA (только для подписи)
  (4) RSA (только для подписи)
  (14) Имеющийся на карте ключ
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
  <n>w = срок действия ключа - n недель
  <n>m = срок действия ключа - n месяцев
  <n>y = 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: Стефан
Адрес электронной почты: stephantatur@gmail.com
Примечание: Создание gpg ключа
Используется таблица символов 'utf-8'.
Вы выбрали следующий идентификатор пользователя:
  "Стефан (Создание gpg ключа) <stephantatur@gmail.com>"

Сменить (N)Имя, (C)Примечание, (E)Адрес; (O)Принять/(Q)Выход? 0
Сменить (N)Имя, (C)Примечание, (E)Адрес; (O)Принять/(Q)Выход? y
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: создан каталог '/afs/.dk.sci.pfu.edu.ru/home/s/a/satatur/.gnupg/openpgp-revocs.'
gpg: сертификат отзыва записан в '/afs/.dk.sci.pfu.edu.ru/home/s/a/satatur/.gnupg/op
открытый и секретный ключи созданы и подписаны.

pub   rsa4096 2023-02-17 [SC]
      84AFD3448898F14D85DECE50D5ADCA4BC01D3AF6
uid   Стефан (Создание gpg ключа) <stephantatur@gmail.com>

```

Рис. 2.5: Название рисунка

5. Добавлем PGP ключ в GitHub. Используем `gpg --list-secret-keys --keyid-format LONG`. По образцу видим отпечаток моего ключа, вставляем его в следующую конструкцию: `gpg --armor --export | xclip -sel clip`. Затем перешли в настройки github и вставили полученный ключ
6. Готовые SSH и PGP ключи на github.

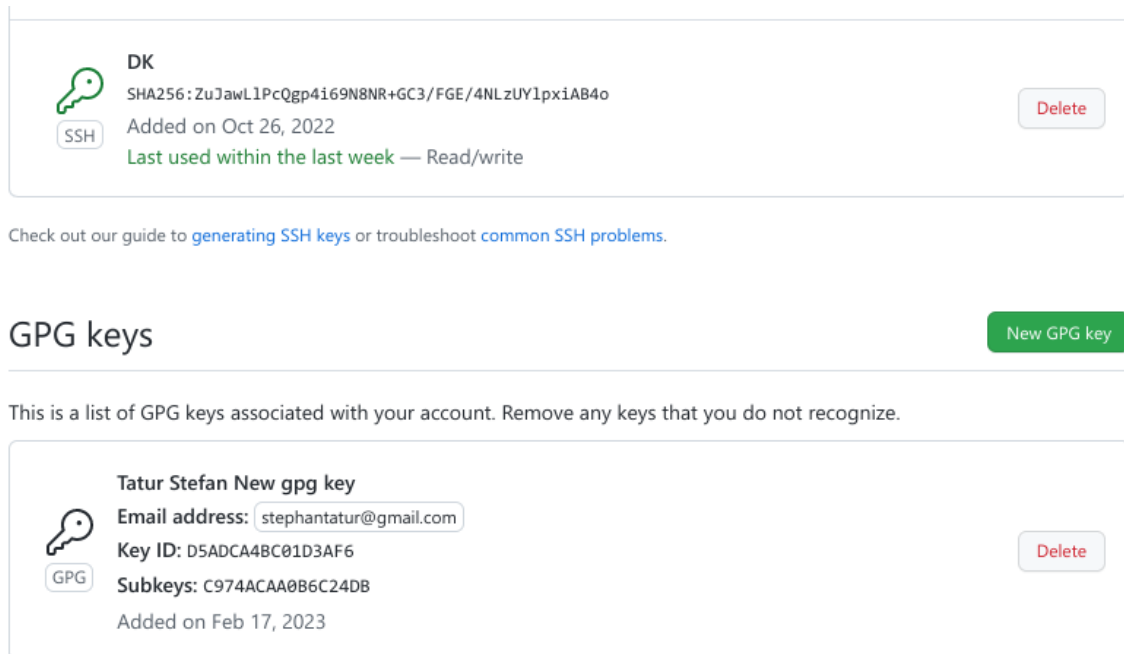


Рис. 2.6: Название рисунка

7. Затем настраиваем автоматические подписи коммитов `git: git config --global user.signingkey`, `git config --global commit.gpgsign true`, `git config --global gpg.program $(which gpg2)`.

Create a new repository from course-directory-student-template

The new repository will start with the same files and folders as yamadharma/course-directory-student-template.

Owner * Repository name *

KseniyaSyachinova / study_2021-2022_OS ✓

Great repository names are short a study_2021-2022_OS is available. in? How about fluffy-succotash?

Description (optional)

☒ Public
Anyone on the internet can see this repository. You choose who can commit.

☐ Private
You choose who can see and commit to this repository.

☐ Include all branches
Copy all branches from yamadharma/course-directory-student-template and not just master.

Create repository from template

Рис. 2.7: Название рисунка

8. После этого создаём репозиторий курса на основе шаблона:

```
satatur@dk6n64 ~ $ git config --global stefantatur.signingkey stephantatur@gmail.com
satatur@dk6n64 ~ $
satatur@dk6n64 ~ $ git config --global commit.gpgsign true
satatur@dk6n64 ~ $ git config --global gpg.program $(which gpg2)
```

Рис. 2.8: Название рисунка

9. Готовый репозиторий на github.

stefantatur feat(lab01)		a951b33 yesterday	🕒 3 commits
📁 config	Initial commit		yesterday
📁 labs	feat(lab01)		yesterday
📁 presentation	feat(main): make new course structure		yesterday
📁 project-personal	feat(main): make new course structure		yesterday
📁 template	Initial commit		yesterday
📄 .gitattributes	Initial commit		yesterday
📄 .gitignore	Initial commit		yesterday
📄 .gitmodules	Initial commit		yesterday
📄 CHANGELOG.md	Initial commit		yesterday
📄 COURSE	feat(main): make new course structure		yesterday
📄 LICENSE	Initial commit		yesterday
📄 Makefile	Initial commit		yesterday
📄 README.en.md	Initial commit		yesterday
📄 README.git-flow.md	Initial commit		yesterday
📄 README.md	Initial commit		yesterday
📄 prepare	feat(main): make new course structure		yesterday

Выводы

Я изучил идеологию и научился применять средства контроля версий.

#Контрольные вопросы

1). Система контроля версий Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды `git` различными опциями. Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом.

2). 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 неполную версию изменённых файлов, а производить так называемую дельта-компрессию — сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных. 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.

3). Централизованные системы — это системы, которые используют архитектуру клиент / сервер, где один или несколько клиентских узлов напрямую подключены к центральному серверу. Пример - Wikipedia. В децентрализованных системах каждый узел принимает свое собственное решение. Конечное поведение системы является совокупностью решений отдельных узлов. Пример — Bitcoin. В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером.

4). Создадим локальный репозиторий. Сначала сделаем предварительную конфигурацию, указав имя и email владельца репозитория: `git config --global user.name "Имя Фамилия"` `git config --global user.email "work@mail"` и настроив utf-8 в выводе сообщений `git: git config --global quotePath false` Для инициализации локального репозитория, расположенного, например, в каталоге `~/tutorial`, необходимо ввести в командной строке: `cd mkdir tutorial cd tutorial git init`

5). Для последующей идентификации пользователя на сервере репозитория необходимо сгенерировать пару ключей (приватный и открытый): `ssh-keygen`

-С"Имя Фамилия work@mail" Ключи хранятся в каталоге ~/.ssh/. Скопировав из локальной консоли ключ в буфер обмена `cat ~/.ssh/id_rsa.pub | xclip -sel clip` вставляем ключ в появившееся на сайте поле.

6). 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). Основные команды 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). Использование git при работе с локальными репозиториями (добавления текстового документа в локальный репозиторий): `git add hello.txt git commit`

-am'Новый файл

9). Проблемы, которые решают ветки git: · нужно постоянно создавать архивы с рабочим кодом · сложно “переключаться” между архивами · сложно перетаскивать изменения между архивами · легко что-то напутать или потерять

10). 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 ервисов. Для этого сначала нужно получить списки имеющихся шаблонов: `curl -L -s https://www.gitignore.io/api/list` Затем скачать шаблон, например, для C и C++ `curl -L -s https://www.gitignore.io/api/c` » `.gitignore` `curl -L -s https://www.gitignore.io/api/c++` » `.gitignore`