

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ»

Факультет физико-математических и естественных наук
Кафедра информационных технологий

«УТВЕРЖДАЮ»
Заведующий кафедрой
информационных технологий
д.ф.-м.н., проф.
Ю.Н. Орлов
«___» _____ 2022г.

ОТЧЕТ
по лабораторной работе

ТЕМА «Управление версиями»
по дисциплине «Компьютерный практикум по ИТ»

Выполнил:

Студент группы НПИбд-01-21

Студенческий билет № 1032211402

Лефтеров Игорь Иванович

(Подпись)

«___» _____ 2022г.

Руководитель:

к.ф.-м.н., доцент кафедры
информационных технологий

Кулябов Дмитрий Сергеевич

(Подпись)

«___» _____ 2022г.

Оценка: _____

Москва, 2022

Цель работы:

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

Ход работы:

- 1) Создаем учетную запись на github.com

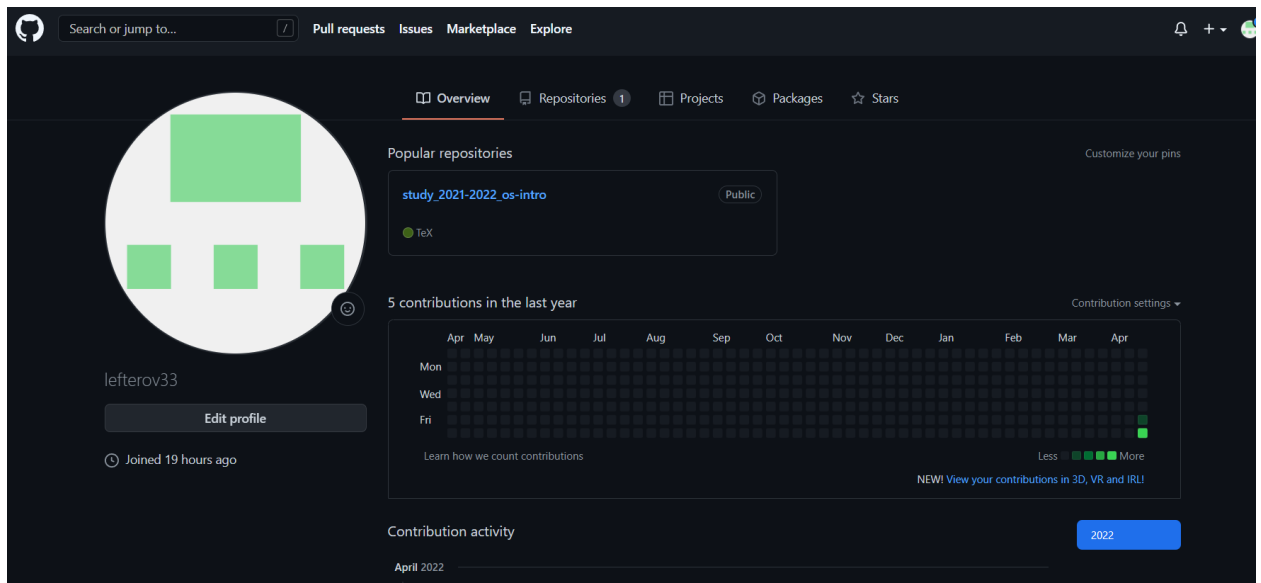


Рис1.

Настраиваем систему контроля версий git. Синхронизируем учётную запись github с компьютером:

```
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 "lefterv33 <lefterv_33@mail>"`) и привязываем его к компьютеру через консоль.

Базовая настройка git

```
[lefterv@lefterv tmp]$ git config --global user.name "lefterv33"
[lefterv@lefterv tmp]$ git config --global user.name "lefterv_33@mail.ru"
[lefterv@lefterv tmp]$ git config --global user.name "lefterv33"
[lefterv@lefterv tmp]$ git config --global user.email "lefterv_33@mail.ru"
[lefterv@lefterv tmp]$
```

Зададим имя и почту

```
[lfterov@lfterov tmp]$ git config --global user.name "lfterov33"
[lfterov@lfterov tmp]$ git config --global user.name "lfterov_33@mail.ru"
[lfterov@lfterov tmp]$ git config --global user.name "lfterov33"
[lfterov@lfterov tmp]$ git config --global user.email "lfterov_33@mail.ru"
[lfterov@lfterov tmp]$ git config --global core.quotePath false
[lfterov@lfterov tmp]$ git config --global init.defaultBranch master
[lfterov@lfterov tmp]$ git config --global core.autocrlf input
[lfterov@lfterov tmp]$ git config --global core.safecrlf warn
[lfterov@lfterov tmp]$
```


Настроим utf-8 в выводе сообщений git, настройка верификации и подписание коммитов, зададим имя начальной ветки(будем называть ее master), параметр auticrlf, параметр safecrlf.

```
[lfterov@lfterov tmp]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/lfterov/.ssh/id_rsa):
Created directory '/home/lfterov/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/lfterov/.ssh/id_rsa
Your public key has been saved in /home/lfterov/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:T1ztv02/d9mrIZMrAEdCMBSNmAzVFW6l9Qs9PRb1nY lfterov@lfterov
The key's randomart image is:
+----[RSA 4096]-----+
|+0+++o.o .          |
|ooo...o.= .  ..     |
|.  .oo + ..o.oE|
|.  .o.o. .+. . |
|.ooS o.  . |
|.. o  .  . |
|.  .+ .  = |
|.  + .oB|
|.. ..+B|
+-----[SHA256]-----+
[lfterov@lfterov tmp]$
```

Создание ключа ssh

SSH keys
New SSH key

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.


key

SHA256:mvV8b3Rs20nYZ/W+w/UTRGmAqzLZ5P7ybDwG/OTkq9E
Added on 23 Apr 2022
Last used within the last week — Read/write

Delete

Check out our guide to [generating SSH keys](#) or troubleshoot [common SSH problems](#).

Добавление ключа на git

```
[lfterov@lfterov tmp]$ gpg --list-secret-keys --keyid-format LONG
gpg: checking the trustdb
gpg: marginals needed: 3 completes needed: 1 trust model: pgp
gpg: depth: 0 valid: 3 signed: 0 trust: 0-, 0q, 0n, 0m, 0f, 3u
/home/lfterov/.gnupg/pubring.kbx
-----
sec   rsa4096/97DAC819134BB123 2022-04-22 [SC]
      A4A8E5B95F3BD2EB71E9FF2097DAC819134BB123
uid           [ultimate] lfterov33 <lfterov_33@mail.ru>
ssb   rsa4096/4C857774398AF2ED 2022-04-22 [E]

sec   rsa4096/E78C03CC607B2F0C 2022-04-22 [SC]
      E86F639C529ESC7E9C34CFC0E78C03CC607B2F0C
uid           [ultimate] lfterov33 <lfterov_33@mail.ru>
ssb   rsa4096/532E8761BA9C1F82 2022-04-22 [E]


sec   rsa4096/68868B62EE012505 2022-04-22 [SC]
      BB89E5C0F10CDF132C405FC468868B62EE012505
uid           [ultimate] lfterov33 <lfterov_33@mail.ru>
ssb   rsa4096/4E959E56B8E9A262 2022-04-22 [E]

[lfterov@lfterov tmp]$ gpg --armor --export | xclip -sel clip
```

Создание ключа pgr

GPG keys

This is a list of GPG keys associated with your account. Remove any keys that you do not recognize.



Email address: lfterov_33@mail.ru
Key ID: 97DAC819134BB123
Subkeys: 4C857774398AF2ED
Added on 23 Apr 2022

Delete

[Learn how to generate a GPG key and add it to your account.](#)

Добавление ключа GPG

```
usage: gpg [options] [filename]
[lfterov@lfterov tmp]$ git config --global user.singingkey 97DAC819134BB123
[lfterov@lfterov tmp]$ git config --global commit.gpgsign true
[lfterov@lfterov tmp]$ git config --global gpg.program $(which gpg2)
```

Настройка автоматических подписей коммитов git

```
lfterov@lfterov/tmp — gh auth login
[lfterov@lfterov tmp]$ gh auth login
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations? [Use arrows to move, type to filter]
? What is your preferred protocol for Git operations? HTTPS
? Authenticate Git with your GitHub credentials? Yes
? How would you like to authenticate GitHub CLI? Login with a web browser

First copy your one-time code: 398F-B19F
Press Enter to open github.com in your browser...
```

Настройка gh

```
[lefterov@lefterov tmp]$ git clone --recursive https://github.com/yamadharma/course-directory-student-template.git
Cloning into 'course-directory-student-template'...
remote: Enumerating objects: 35, done.
remote: Counting objects: 100% (35/35), done.
remote: Compressing objects: 100% (28/28), done.
remote: Total 35 (delta 7), reused 34 (delta 6), pack-reused 0
Receiving objects: 100% (35/35), 15.77 KiB | 15.77 MiB/s, done.
Resolving deltas: 100% (7/7), done.
Submodule 'template/presentation' (https://github.com/yamadharma/academic-presentation-markdown-template.git) registered for path 'template/presentation'
Submodule 'template/report' (https://github.com/yamadharma/academic-laboratory-report-template.git) registered for path 'template/report'
Cloning into '/tmp/course-directory-student-template/template/presentation'...
remote: Enumerating objects: 42, done.
remote: Counting objects: 100% (42/42), done.
remote: Compressing objects: 100% (34/34), done.
remote: Total 42 (delta 9), reused 40 (delta 7), pack-reused 0
Receiving objects: 100% (42/42), 31.19 KiB | 550.00 KiB/s, done.
Resolving deltas: 100% (9/9), done.
Cloning into '/tmp/course-directory-student-template/template/report'...
```

Шаблон для рабочего пространства

<https://github.com/yamadharma/course-directory-student-template>.

```
Cloning into '/home/lefterov/work/study/2021-2022/OC/os-intro/template/presentation'...
remote: Enumerating objects: 42, done.
remote: Counting objects: 100% (42/42), done.
remote: Compressing objects: 100% (34/34), done.
remote: Total 42 (delta 9), reused 40 (delta 7), pack-reused 0
Receiving objects: 100% (42/42), 31.19 KiB | 725.00 KiB/s, done.
Resolving deltas: 100% (9/9), done.
Cloning into '/home/lefterov/work/study/2021-2022/OC/os-intro/template/report'...
remote: Enumerating objects: 78, done.
remote: Counting objects: 100% (78/78), done.
remote: Compressing objects: 100% (52/52), done.
remote: Total 78 (delta 31), reused 69 (delta 22), pack-reused 0
Receiving objects: 100% (78/78), 292.27 KiB | 1.42 MiB/s, done.
Resolving deltas: 100% (31/31), done.
Submodule path 'template/presentation': checked out '3eae8b7586f8a9aded2b506cd1018e625b228b93'
Submodule path 'template/report': checked out 'df7b2ef80f8def3b9a496f8695277469ala7842a'
[lefterov@lefterov OC]$ cd ~/work/study/2021-2022/"OC"/os-intro
[lefterov@lefterov os-intro]$
```

Создание репозитория курса на основе шаблона

```
[lefterov@lefterov OC]$ cd ~/work/study/2021-2022/"OC"/os-intro
[lefterov@lefterov os-intro]$ rm package.json
[lefterov@lefterov os-intro]$ ls
config  Makefile      README.git-flow.md  template
LICENSE README.en.md  README.md
```

```
[lefterov@lefterov os-intro]$ git push
Enumerating objects: 20, done.
Counting objects: 100% (20/20), done.
Compressing objects: 100% (16/16), done.
Writing objects: 100% (19/19), 266.54 KiB | 2.17 MiB/s, done.
Total 19 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To github.com:lefterov33/study_2021-2022_os-intro.git
d863a73..cd4b8f1 master -> master
[lefterov@lefterov os-intro]$
```

Настройка каталога курса

Выводы:

В ходе выполнения данной лабораторной работы были приобретены практические навыки git на виртуальную машину, а также настройка дополнительного для работы сервисов.

1). Система контроля версий Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды git с 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 core.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 -C "Имя Фамилия <work@mail>"
```

Ключи сохраняются 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 впоследствии 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. 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
```