

Лабораторная работа 2

Управление версиями

Лебедева Ольга Андреевна

Содержание

1	Цель работы	5
2	Указания к лабораторной работе	6
3	Последовательность выполнения работы	7
4	Базовая настройка git	8
5	Добавление PGP ключа в GitHub	11
6	Создание репозитория	12
7	Настройка каталога курса	15
8	Настройка автоматических подписей коммитов git	16
9	Вывод	17
10	Контрольные вопросы	18

Список иллюстраций

3.1	Создание профиля на Гитхаб	7
4.1	Имя и email репозитория	8
4.2	Задаём имя ветки и параметры	8
4.3	ключ ssh	9
4.4	Опции ключа	10
5.1	Вывод ключа	11
5.2	Добавление ключа GPG	11
6.1	Создание репозитория	12
6.2	Копируем ссылку на репозиторий	13
6.3	Добавление первых ключей	13
6.4	Клонирование файлов в репозитория	14
7.1	Удаление файла	15
7.2	Обновление ветки (из-за возникновения ошибки)	15
8.1	Нахождение новых файлов в папках	16
8.2	Отправка файлов в репозиторий	16
10.1	Единоличная работа с хранилищем	19
10.2	Работы с общим хранилищем	19

Список таблиц

1 Цель работы

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

2 Указания к лабораторной работе

Система контроля версий Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды `git` с различными опциями.

3 Последовательность выполнения работы

Настройка github

1. Создайте учётную запись на <https://github.com>.
2. Заполните основные данные на <https://github.com>. (рис. 3.1)

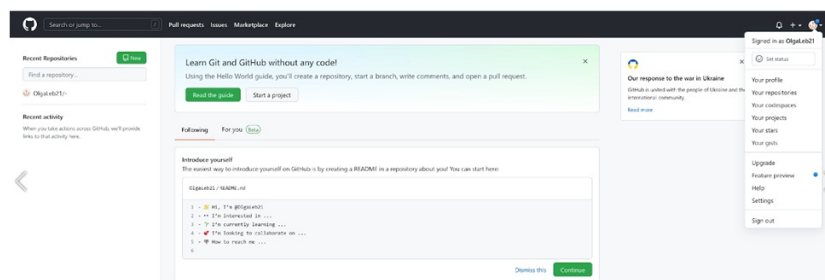


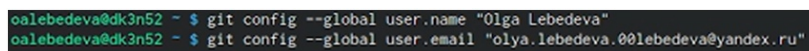
Рис. 3.1: Создание профиля на Гитхаб

4 Базовая настройка git

– Зададим имя и email владельца репозитория: (рис. 4.1)

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

`git config --global user.email "work@mail"`

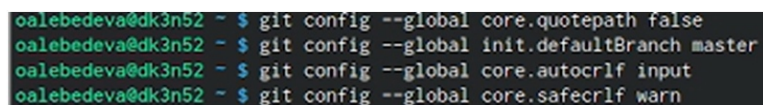


```
oalebedeva@dk3n52 ~ $ git config --global user.name "Olga Lebedeva"
oalebedeva@dk3n52 ~ $ git config --global user.email "olya.lebedeva.00lebedeva@yandex.ru"
```

Рис. 4.1: Имя и email репозитория

Настроим верификацию и подписание коммитов git.

– Зададим имя начальной ветки (будем называть её master):(рис. 4.2)



```
oalebedeva@dk3n52 ~ $ git config --global core.quotepath false
oalebedeva@dk3n52 ~ $ git config --global init.defaultBranch master
oalebedeva@dk3n52 ~ $ git config --global core.autocrlf input
oalebedeva@dk3n52 ~ $ git config --global core.safecrlf warn
```

Рис. 4.2: Задаём имя ветки и параметры

Создание ключей ssh

Генерируем ключ (рис. 4.3)

`gpg --full-generate-key`


```

oalebedeva@dk3n52 ~ $ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/afs/.dk.sci.pfu.edu
Created directory '/afs/.dk.sci.pfu.edu.ru/home/o/a/oaleb
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /afs/.dk.sci.pfu.ed
Your public key has been saved in /afs/.dk.sci.pfu.edu.ru
The key fingerprint is:
SHA256:19+WLrErdEzTe3lVb0GZlMwlyIvSrPrTIuLucgbpD0k oalebe
The key's randomart image is:
+---[RSA 4096]-----+
|          . . +==* |
|           o  == |
|          . . . . + |
|           o..o .+ |
| . .      So. + ..+ |
| o o      .o. = ++ |
| .+ .     o. . +.= |
| E+ + . . + .. o.. |
| B+..+.o ..o. |
+---[SHA256]-----+

```

Рис. 4.3: ключ ssh

- Из предложенных опций выбираем:(рис. 4.4) – тип RSA and RSA;
- размер 4096;
 - выберите срок действия; значение по умолчанию— 0 (срок действия не истекает никогда).
 - GPG запросит личную информацию, которая сохранится в ключе: – Имя (не менее 5 символов).
 - Адрес электронной почты.
 - При вводе email убедитесь, что он соответствует адресу, используемому на GitHub.
 - Комментарий. Можно ввести что угодно или нажать клавишу ввода, чтобы оставить это поле пустым.

```

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
Ваше полное имя: Olga
Имя не должно быть короче 5 символов
Ваше полное имя: Ольга
Адрес электронной почты: olya.lebedeva.001lebedeva@yandex.ru
Примечание:

```

Рис. 4.4: Опции ключа

5 Добавление PGP ключа в GitHub

– Выводим список ключей и копируем отпечаток приватного ключа:(рис. 5.1)

`gpg --list-secret-keys --keyid-format LONG`

```
oalebedeva@dk3n52 ~ $ 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, 1
/afs/.dk.sci.pfu.edu.ru/home/o/a/oalebedeva/.gnupg/pubring.kbx
-----
sec   rsa4096/A1ECE56D051AA5B 2022-04-20 [SC]
      1A453FF652EAB4D6B955EBFCA1ECE56D051AA5B
uid           [ абсолютно ] OlgaLeb <olya.lebedeva.00lebedeva@yandex.ru>
ssb   rsa4096/5198F389650AED98 2022-04-20 [E]
```

Рис. 5.1: Вывод ключа

Перейдём в настройки GitHub (<https://github.com/settings/keys>), нажмём на кнопку New GPG key и вставим полученный ключ в поле ввода.(рис. 5.2)

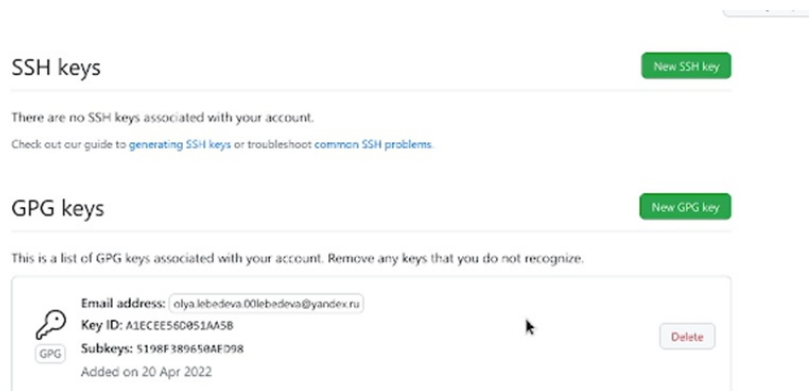


Рис. 5.2: Добавление ключа GPG

6 Создание репозитория

Шаблон для рабочего пространства – Репозиторий: <https://github.com/yamadharma/course-directory-student-template>. (рис. 6.1)

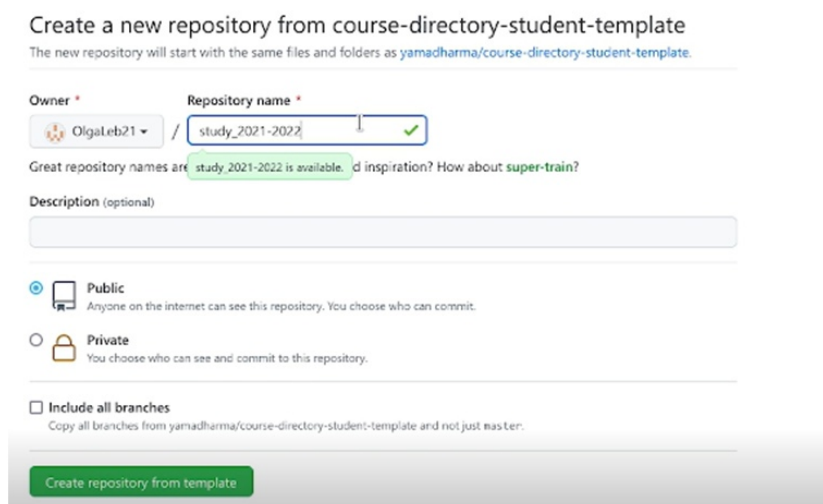


Рис. 6.1: Создание репозитория

Создание репозитория примет следующий вид:

```
mkdir -p ~/work/study/2021-2022/“Операционные системы”  
cd ~/work/study/2021-2022/“Операционные системы”  
gh repo create study_2021-2022_os-intro --template=yamadharma/course-directory-student-template --public
```

```
git clone --recursive git@github.com:/study_2021-2022_os-intro.git os-intro
```

Команда `gh` оказалась недоступной для использования, поэтому мы пошли другим путём: (рис. 6.2)

Клонируем репозиторий, представленный в виде шаблона. (рис. 6.4)

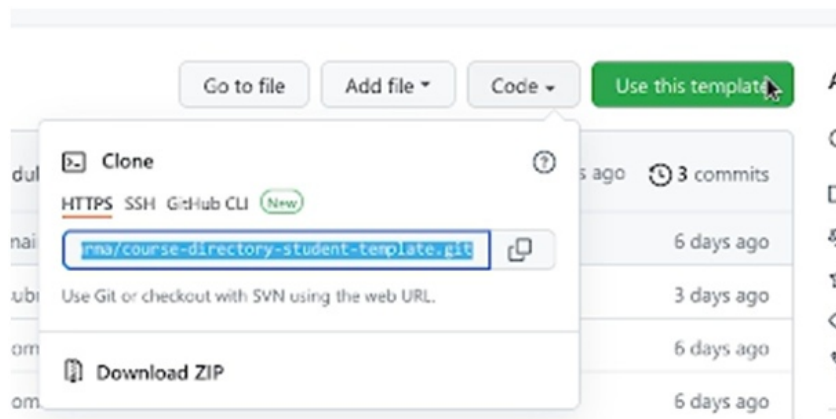


Рис. 6.2: Копируем ссылку на репозиторий

Далее, аналогично ключам GPG впишем ключи SSH:(рис. 6.3)

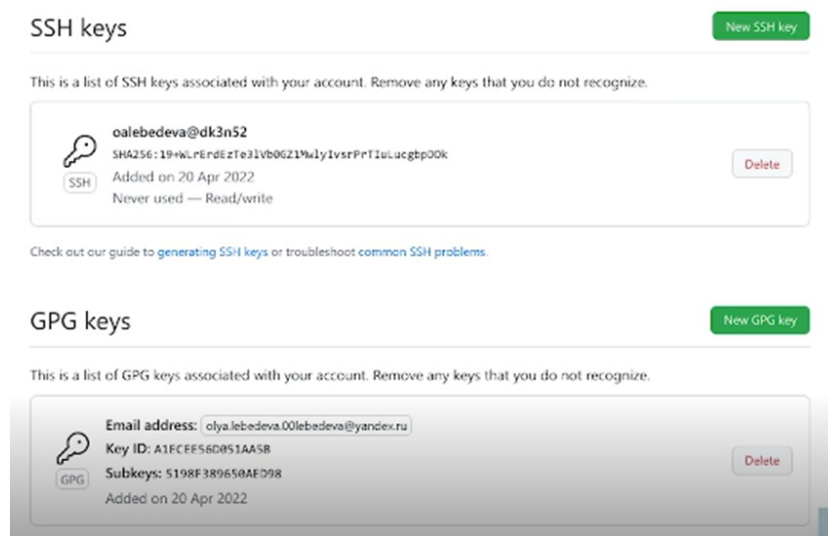


Рис. 6.3: Добавление первых ключей

7 Настройка каталога курса

Перейдем в каталог курса и удалим лишние файлы:(рис. 7.1)

```
oalebedeva@dk3n52 ~/work/study/2021-2022/Операционные системы $ cd study_2021-2022_os-intro/  
oalebedeva@dk3n52 ~/work/study/2021-2022/Операционные системы/study_2021-2022_os-intro $ rm package.json
```

Рис. 7.1: Удаление файла

Отправим файлы на сервер:(рис. 7.2)

git add .

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

git push

```
oalebedeva@dk3n52 ~/work/study/2021-2022/Операционные системы/study_2021-2022_os-intro $ git add .  
oalebedeva@dk3n52 ~/work/study/2021-2022/Операционные системы/study_2021-2022_os-intro $ git status  
На ветке master  
Ваша ветка обновлена в соответствии с «origin/master».  
  
Изменения, которые будут включены в коммит:  
(используйте «git restore --staged <файл>...», чтобы убрать из индекса)  
новый файл:   labs/lab01/presentation/Makefile  
новый файл:   labs/lab01/presentation/presentation.md  
новый файл:   labs/lab01/report/Makefile  
новый файл:   labs/lab01/report/bib/cite.bib  
новый файл:   labs/lab01/report/image/placemg_800_600_tech.jpg  
новый файл:   labs/lab01/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl  
новый файл:   labs/lab01/report/report.md  
новый файл:   labs/lab02/presentation/Makefile  
новый файл:   labs/lab02/presentation/presentation.md  
новый файл:   labs/lab02/report/Makefile  
новый файл:   labs/lab02/report/bib/cite.bib
```

Рис. 7.2: Обновление ветки (из-за возникновения ошибки)

8 Настройка автоматических подписей коммитов git

– Используя введённый email, укажем Git применять его при подписи коммитов:(рис. 8.1) (рис. 8.2)

```
git config --global user.signingkey
```

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

```
git config --global gpg.program $(which gpg2)
```

```
oalebedeva@dk3n52 ~/work/study/2021-2022/Операционные системы/study_2021-2022_os-intro $ git config --global user.signingkey A1ECE56D051AA5B
oalebedeva@dk3n52 ~/work/study/2021-2022/Операционные системы/study_2021-2022_os-intro $ git config --global commit.gpgsign true
oalebedeva@dk3n52 ~/work/study/2021-2022/Операционные системы/study_2021-2022_os-intro $ git config --global gpg.program $(which gpg2)
oalebedeva@dk3n52 ~/work/study/2021-2022/Операционные системы/study_2021-2022_os-intro $ git commit -am 'Feat(main): make course structure'
[master bd8ec02] Feat(main): make course structure
149 files changed, 16590 insertions(+), 14 deletions(-)
create mode 100644 labs/lab01/presentation/Makefile
create mode 100644 labs/lab01/presentation/presentation.md
create mode 100644 labs/lab01/report/Makefile
create mode 100644 labs/lab01/report/bib/cite.bib
```

Рис. 8.1: Нахождение новых файлов в папках

```
oalebedeva@dk3n52 ~/work/study/2021-2022/Операционные системы/study_2021-2022_os-intro $ git push
Перечисление объектов: 20, готово.
Подсчет объектов: 100% (20/20), готово.
При сжатии изменений используется до 6 потоков
Сжатие объектов: 100% (16/16), готово.
Запись объектов: 100% (19/19), 266.53 КиБ | 3.13 МиБ/с, готово.
Всего 19 (изменений 2), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To github.com:Olgaleb21/study_2021-2022_os-intro.git
6d187f2..bd8ec02 master -> master
```

Рис. 8.2: Отправка файлов в репозиторий

9 Вывод

- Изучили идеологию и применение средств контроля версий.
- Освоили умения по работе с git.

10 Контрольные вопросы

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

Система контроля версий — это система, записывающая изменения в файл или набор файлов в течение времени и позволяющая вернуться позже к определённой версии. Для контроля версий файлов в этой книге в качестве примера будет использоваться исходный код программного обеспечения, хотя на самом деле вы можете использовать контроль версий практически для любых типов файлов.

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 («[трудовой] вклад», не переводится) — процесс создания новой версии Рабочая копия (working copy) — текущее состояние файлов проекта, основанное на версии, загруженной из хранилища (обычно на последней). Версия (revision), или ревизия, — состояние всех файлов на определенный момент времени, сохраненное в репозитории, с дополнительной информацией

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

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)

4. Опишите действия с VCS при единоличной работе с хранилищем.(рис. 10.1)

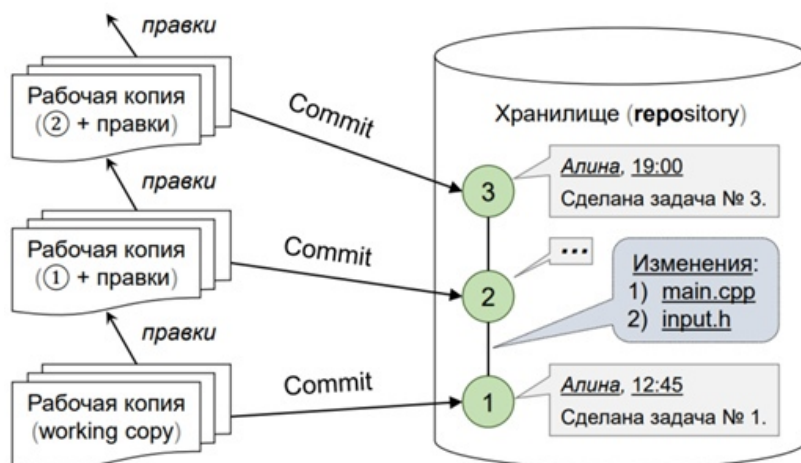


Рис. 10.1: Единоличная работа с хранилищем

5. Опишите порядок работы с общим хранилищем VCS. (рис. 10.2)

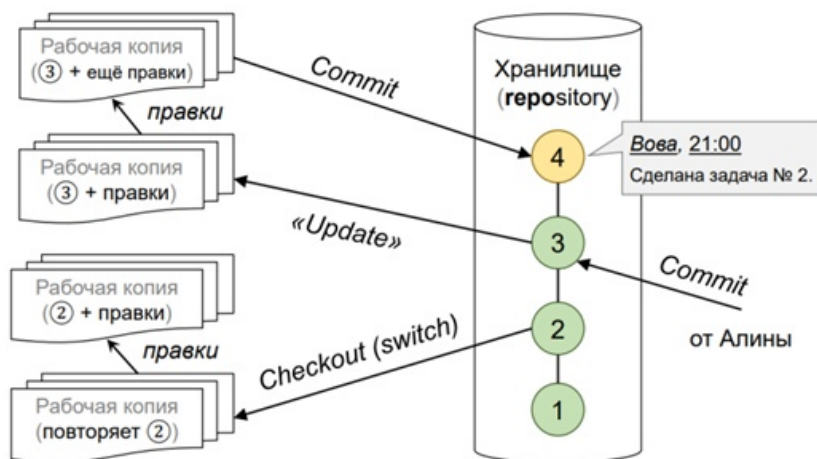


Рис. 10.2: Работы с общим хранилищем

6. Каковы основные задачи, решаемые инструментальным средством git?

У Git есть две основные задачи: хранить информацию обо всех изменениях в коде, начиная с самой первой строчки, и обеспечить удобства командной работы над кодом.

7. Назовите и дайте краткую характеристику командам 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`

8. Приведите примеры использования при работе с локальным и удалённым репозиториями.

Локальный репозиторий — она же директория “.git”. В ней хранятся коммиты и другие объекты. Удаленный репозиторий – тот самый репозиторий который считается общим, в который вы можете передать свои коммиты из локального репозитория, что бы остальные программисты могли их увидеть. Удаленных репозиторий может быть несколько, но обычно он бывает один.

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

‘Git branch’ – это команда для управления ветками в репозитории Git. Ветка – это просто «скользящий» указатель на один из коммитов. Когда мы создаём новые

коммиты, указатель ветки автоматически сдвигается вперёд, к вновь созданному коммиту. Ветки используются для разработки одной части функционала изолированно от других. Каждая ветка представляет собой отдельную копию кода проекта. Ветки позволяют одновременно работать над разными версиями проекта. Ветвление («ветка», branch) — один из параллельных участков истории в одном хранилище, исходящих из одной версии (точки ветвления). Ветки нужны для того, чтобы программисты могли вести совместную работу над проектом и не мешать друг другу при этом.

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

Игнорируемые файлы обычно представляют собой файлы, специфичные для платформы, или автоматически созданные из сборочных систем. Временно игнорировать изменения в файле можно командой: `git update-index —assume-unchanged`