

Отчёт по лабораторной работе № 2

Королёв Иван Андреевич

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
4.1	Установка программного обеспечения	9
4.2	Базовая настройка git	10
4.3	Создайте ключи ssh	11
4.4	Создайте ключи pgr	11
4.5	Настройка github	12
4.6	Добавление PGP ключа в GitHub	13
4.7	Настройка автоматических подписей коммитов git	14
4.8	Настройка gh	14
4.9	Создание репозитория курса на основе шаблона	14
4.10	Настройка каталога курса	15
5	Выводы	16

Список иллюстраций

4.1	git	9
4.2	gh	9
4.3	name and email	10
4.4	utf-8	10
4.5	master	10
4.6	autocrlf	10
4.7	safecrlf	10
4.8	ssh	11
4.9	pgp	11
4.10	pgp	12
4.11	github	12
4.12	key pgp	13
4.13	copy pgp	13
4.14	pgp	13
4.15	подписи коммитов	14
4.16	gh	14
4.17	gh	14
4.18	Файлы	15
4.19	Файлы	15

Список таблиц

1 Цель работы

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

2 Задание

- Создать базовую конфигурацию для работы с git.
- Создать ключ SSH.
- Создать ключ PGP.
- Настроить подписи git.
- Зарегистрироваться на Github.
- Создать локальный каталог для выполнения заданий по предмету.

3 Теоретическое введение

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.

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

Системы контроля версий поддерживают возможность отслеживания и разрешения конфликтов, которые могут возникнуть при работе нескольких человек над одним файлом. Можно объединить (слить) изменения, сделанные разными участниками (автоматически или вручную), вручную выбрать нужную версию,

отменить изменения вовсе или заблокировать файлы для изменения. В зависимости от настроек блокировка не позволяет другим пользователям получить рабочую копию или препятствует изменению рабочей копии файла средствами файловой системы ОС, обеспечивая таким образом, привилегированный доступ только одному пользователю, работающему с файлом.

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.

В отличие от классических, в распределённых системах контроля версий центральный репозиторий не является обязательным.

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

4.1 Установка программного обеспечения

- Установка git (рис. 4.1).

```
[root@iakorolyov ~]# dnf install git
Ожидание завершения процесса с PID 10589.
Fedora 36 - x86_64 - Updates          914 B/s | 8.4 kB    00:09
Fedora 36 - x86_64 - Updates        376 kB/s | 3.9 MB    00:10
Fedora Modular 36 - x86_64 - Updates  16 kB/s | 18 kB     00:01
Пакет git-2.39.1-1.fc36.x86_64 уже установлен.
Зависимости разрешены.
Отсутствуют действия для выполнения.
Выполнено!
[root@iakorolyov ~]#
```

Рис. 4.1: git

- Установка gh (рис. 4.2).

```
[root@iakorolyov ~]# dnf install git
Fedora 36 - x86_64 - Updates          10 kB/s | 9.9 kB     00:00
Fedora 36 - x86_64 - Updates        306 kB/s | 3.3 MB     00:11
Fedora Modular 36 - x86_64 - Updates  15 kB/s | 18 kB      00:01
Пакет git-2.39.1-1.fc36.x86_64 уже установлен.
Зависимости разрешены.
Отсутствуют действия для выполнения.
Выполнено!
```

Рис. 4.2: gh

4.2 Базовая настройка git

- Зададим имя и email владельца репозитория (рис. 4.3).

```
[iakorolyov@fedora ~]$ git config --global user.name "<iakorolyov>"  
[iakorolyov@fedora ~]$ git config --global user.email "<1032225751@pfur.ru>"
```

Рис. 4.3: name and email

- Настроим utf-8 в выводе сообщений git (рис. 4.4).

```
[iakorolyov@fedora ~]$ git config --global core.quotepath false
```

Рис. 4.4: utf-8

- Зададим имя начальной ветки (будем называть её master) (рис. 4.5).

```
[iakorolyov@fedora ~]$ git config --global init.defaultBranch master
```

Рис. 4.5: master

- Параметр autocrlf (рис. 4.6).

```
[iakorolyov@fedora ~]$ git config --global core.autocrlf input
```

Рис. 4.6: autocrlf

- Параметр safecrlf (рис. 4.7).

```
[iakorolyov@fedora ~]$ git config --global core.safecrlf warn
```

Рис. 4.7: safecrlf

4.3 Создайте ключи ssh

- Мы уже умеем создавать ssh из прошлого курса Архитектуры компьютеров. (рис. 4.8).

```
[iakorolyov@fedora ~]$ ssh-keygen -C "Ivan Korolev <1032225751@pfur.ru>"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/iakorolyov/.ssh/id_rsa):
/home/iakorolyov/.ssh/id_rsa already exists.
```

Рис. 4.8: ssh

4.4 Создайте ключи pgp

- Генерируем ключ (рис. 4.9), (рис. 4.10)

```
[root@iakorolyov ~]# gpg --full-generate-key
gpg (GnuPG) 2.3.7; Copyright (C) 2021 Free Software Foundation, Inc.
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 and RSA
  (2) DSA and Elgamal
  (3) DSA (sign only)
  (4) RSA (sign only)
  (9) ECC (sign and encrypt) *default*
 (10) ECC (только для подписи)
 (14) Existing key from card
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
```

Рис. 4.9: pgp

```

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: iakorolyov
Адрес электронной почты: 1032225751@pfur.ru
Примечание:
Вы выбрали следующий идентификатор пользователя:
    "iakorolyov <1032225751@pfur.ru>"

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 генерато
случайных чисел больше возможностей получить достаточное количество
[root@iakorolyov ~]# █16 [E]orolyov <1032225751@pfur.ru>evocs.d/1E1D

```

Рис. 4.10: gpg

4.5 Настройка github

- У меня создан репозиторий. (рис. 4.11)

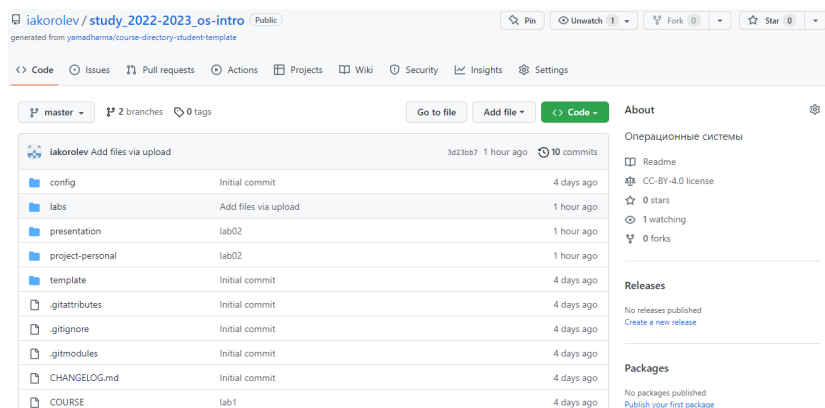


Рис. 4.11: github

4.6 Добавление PGP ключа в GitHub

- Выводим список ключей и копируем отпечаток приватного ключа (рис. 4.12)

```
[root@iakorolyov ~]# 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: 2 подписанных: 0 доверие: 0-, 0q, 0n, 0m,
0f, 2u
/root/.gnupg/pubring.kbx
-----
sec   rsa4096/3D72B9DFE43E3647 2023-02-15 [SC]
      D2C83B1219ADA7EF167982FE3D72B9DFE43E3647
uid   [ абсолютно ] iakorolyov <1032225751@pfur.ru>
ssb   rsa4096/37CDF9C08D0A40C6 2023-02-15 [E]
```

Рис. 4.12: key gpg

- Скопируйте ваш сгенерированный PGP ключ в буфер обмена (рис. 4.13)

```
[root@iakorolyov ~]# gpg --armor --export 3D72B9DFE43E3647 | xclip -sel clip
[root@iakorolyov ~]#
```

Рис. 4.13: copy gpg

- Перейдите в настройки GitHub (<https://github.com/settings/keys>), нажмите на кнопку New GPG key и вставьте полученный ключ в поле ввода (рис. 4.14)

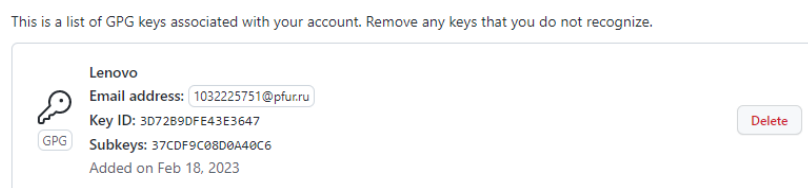


Рис. 4.14: gpg

4.7 Настройка автоматических подписей коммитов git

Используя введённый email, укажите Git применять его при подписи коммитов (рис. 4.15)

```
[root@iakorolyov ~]# git config --global user.signingkey 3D72B9DFE43E3647
[root@iakorolyov ~]# git config --global commit.gpgsign true
[root@iakorolyov ~]# git config --global gpg.program $(which gpg2)
```

Рис. 4.15: подписи коммитов

4.8 Настройка gh

- Авторизоваться в gh (рис. 4.16)

```
[root@iakorolyov ~]# gh auth login
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations? SSH
? Generate a new SSH key to add to your GitHub account? No
? 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'.
? Paste your authentication token: *****
- gh config set -h github.com git_protocol ssh
✓ Configured git protocol
✓ Logged in as iakorolev
[root@iakorolyov ~]#
```

Рис. 4.16: gh

4.9 Создание репозитория курса на основе шаблона

- Создать шаблон рабочего пространства (рис. 4.17)

```
[root@iakorolyov ~]# gh repo create study_2022-2023_os-intro --template=yamadhar/ma/course-directory-student-template --public
```

Рис. 4.17: gh

- Репозиторий “Операционные системы” создан.

4.10 Настройка каталога курса

- Файлы на сервере (рис. 4.18), (рис. 4.19)

```
[iakorolyov@iakorolyov os-intro]$ rm package.json
rm: невозможно удалить 'package.json': Нет такого файла или каталога
[iakorolyov@iakorolyov os-intro]$ ls
CHANGELOG.md  labs      prepare      README.en.md  template
config        LICENSE   presentation  README.git-flow.md
COURSE        Makefile  project-personal  README.md
```

Рис. 4.18: Файлы

The screenshot displays the GitHub interface for the repository 'iakorolev / study_2022-2023_os-intro'. The main area shows a file browser with a table of repository contents. The 'labs' directory is selected, showing its sub-items: 'lab02', 'project-personal', and 'template'. The right sidebar provides additional information, including the repository's license (CC-BY-4.0), star/fork counts, and release status.

File/Folder	Commit Message	Commit Date
config	Initial commit	4 days ago
labs	Add files via upload	1 hour ago
presentation	lab02	1 hour ago
project-personal	lab02	1 hour ago
template	Initial commit	4 days ago
.gitattributes	Initial commit	4 days ago
.gitignore	Initial commit	4 days ago
.gitmodules	Initial commit	4 days ago
CHANGELOG.md	Initial commit	4 days ago
COURSE	lab1	4 days ago

Рис. 4.19: Файлы

5 Выводы

Я освоил умения использования git.