

# **Отчет по лабораторной работе №2**

**Первоначальная настройка git**

Ангелина Павловна Ким

# Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Выводы	16
	Список литературы	17

## Список иллюстраций

3.1	Установка git . . . . .	7
3.2	Имя пользователя и почта . . . . .	7
3.3	Настройка верификации и подписание коммитов git . . . . .	8
3.4	Создание SSH ключа . . . . .	8
3.5	Генерируем ключ pgr . . . . .	8
3.6	Опции . . . . .	9
3.7	Копирование приватного ключа . . . . .	9
3.8	Готовый ключ . . . . .	9
3.9	Подписи коммитов . . . . .	10
3.10	Авторизация . . . . .	10
3.11	Подсоединение . . . . .	10
3.12	Создание репозитория (1) . . . . .	11
3.13	Создание репозитория (2) . . . . .	11
3.14	Настройка каталога курса . . . . .	11
3.15	Отправка файлов (1) . . . . .	11
3.16	Отправка файлов (2) . . . . .	12

## Список таблиц

# 1 Цель работы

Изучить идеологию и применение средств контроля версий. 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 Выполнение лабораторной работы

Для начала установим git (рис. 3.1).

```
[apkim@fedora ~]$ sudo -i
[root@fedora ~]# dnf install gh
Последняя проверка окончания срока действия метаданных: 0:57:32 назад, Чт 16 фев 2023 20:21:18.
Зависимости разрешены.
=====
Пакет      Архитектура  Версия      Репозиторий  Размер
=====
Установка:
gh          x86_64       2.5.2-1.fc34 updates      6.7 М
=====
Результат транзакции
=====
Установка 1 Пакет
=====
Объем загрузки: 6.7 М
Объем изменений: 31 М
Продолжить? [д/Н]: д
Загрузка пакетов:
gh-2.5.2-1.fc34.x86_64.rpm                1.1 MB/s | 6.7 MB  00:05
-----
Общий размер                               972 kB/s | 6.7 MB  00:07
Проверка транзакции
Проверка транзакции успешно завершена.
Идет проверка транзакции
```

Рис. 3.1: Установка git

Далее зададим имя и почту владельца репозитория, а затем настроим utf-8 в выводе сообщений git (рис. 3.2).

```
Установлен:
gh-2.5.2-1.fc34.x86_64

Выполнено!
[root@fedora ~]# git config --global user.name "angelinagata"
[root@fedora ~]# git config --global user.name "kimangelyusha@mail.ru"
[root@fedora ~]# git config --global core.quotepath false
[root@fedora ~]#
```

Рис. 3.2: Имя пользователя и почта

Следующим шагом настроим верификацию и подписание коммитов git. Сначала зададим имя начальной ветки, потом параметр autocrlf, а затем параметр

safecrlf (рис. 3.3).

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

Рис. 3.3: Настройка верификации и подписание коммитов git

Далее нам нужно создать SSH ключи. По алгоритму rsa с ключем размером 4096 бит, по алгоритму ed25519 (рис. 3.4).

```
[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): ssh-keygen -t ed25519
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in ssh-keygen -t ed25519
Your public key has been saved in ssh-keygen -t ed25519.pub
The key fingerprint is:
SHA256:yv0JDN++2Xdo6dzXbqQIpuIncU4NeUnDkViQuGwvvs root@fedora
The key's randomart image is:
+----[RSA 4096]-----+
|          ..*oo|
|          . =  =|
|          + o.o |
|          . oo. |
|      . S      .+|
|      . + .o .o o|
|      o =+...+ =|
|      .+..+oB.=|
|      .B+..+E+|
+----[SHA256]-----+
[root@fedora ~]#
```

Рис. 3.4: Создание SSH ключа

Теперь создаем ключи pgr. Сначала генерируем ключ. Затем из предложенных опций выбираем то, что нужно. (рис. 3.5).

```
[root@fedora ~]# gpg --full-generate-key
gpg (GnuPG) 2.2.27; 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.

gpg: создан каталог '/root/.gnupg'
gpg: создан щит с ключами '/root/.gnupg/pubring.kbx'
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
```

Рис. 3.5: Генерируем ключ pgr

Продолжение опций (рис. 3.6).



```

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: angelinagata
Адрес электронной почты: kimangelyusha@mail.ru
Примечание:
Вы выбрали следующий идентификатор пользователя:
"angelinagata <kimangelyusha@mail.ru>"

```

Рис. 3.6: Опции

Далее выводим список ключей и копируем отпечаток приватного ключа. Затем копируем сгенерированный pgr ключ и вставляем его на github. (рис. 3.7).

```

[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   rsa4096/483B2FCC6E6224F5 2023-02-16 [SC]
      CA4302A15E8C7ED62A6780CF483B2FCC6E6224F5
uid   [ а6солютно ] angelinagata <kimangelyusha@mail.ru>
ssb   rsa4096/EF230CE92943A342 2023-02-16 [E]
[root@fedora ~]#

```


Рис. 3.7: Копирование приватного ключа

Ключ присоединился (рис. 3.8).

GPG keys

New GPG key

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



ангелина

Email address: kimangelyusha@mail.ru

Key ID: 8D9012DBCCBD3B5E

Subkeys: 5257D827FDC09392

Added on Feb 16, 2023

Delete

Рис. 3.8: Готовый ключ

Следующим шагом нужно, используя введенную почту, указать Git и применять его при подписи коммитов (рис. 3.9).

```
[root@fedora ~]# git config --global user.signingkey 8D9012D8CCBD3B5E
[root@fedora ~]# git config --global commit.gpgsign true
[root@fedora ~]# git config --global gpg.program $(which gpg2)
[root@fedora ~]# gh auth login
? What account do you want to log into? [Use arrows to move, type to filter]
> GitHub.com
  GitHub Enterprise Server
```

Рис. 3.9: Подписи коммитов

Далее необходимо авторизоваться, отвечая на вопросы (рис. 3.10).

```
[root@fedora ~]# 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? Login with a web browser

! First copy your one-time code: 2120-EC08
Press Enter to open github.com in your browser...
```

Рис. 3.10: Авторизация

Успешное подключение (рис. 3.11).

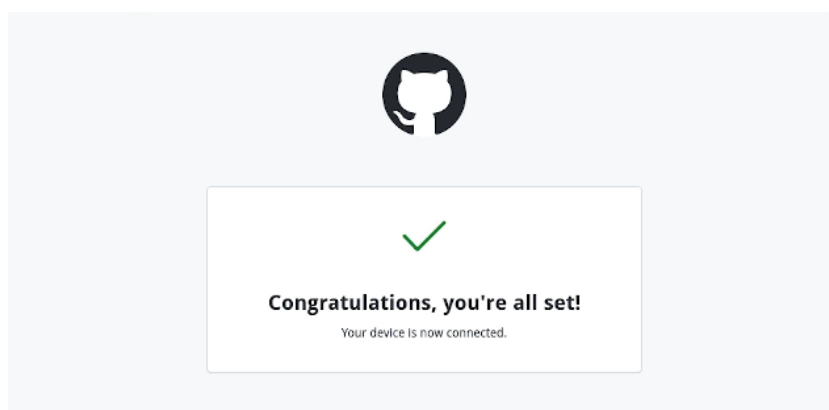


Рис. 3.11: Подсоединение

Теперь нам нужно создать репозиторий на основе шаблона (рис. 3.12).

```
[apkim@fedora ~]$ mkdir -p ~/work/study/2022-2023/"Операционные системы"
[apkim@fedora ~]$ cd ~/work/study/2022-2023/"Операционные системы"
[apkim@fedora Операционные системы]$ gh repo create study_2022-2023_os-intro --temp
late=yamadharma/course-directory-student-template --public
Welcome to GitHub CLI!
To authenticate, please run 'gh auth login'.
```

Рис. 3.12: Создание репозитория (1)

Продолжение (рис. 3.13).

```
[apkim@fedora Операционные системы]$ git clone --recursive git@github.com:angelinagata/study_2022-2023_os-intro.git os-intro
Клонирование в «os-intro»...
remote: Enumerating objects: 27, done.
remote: Counting objects: 100% (27/27), done.
remote: Compressing objects: 100% (26/26), done.
remote: Total 27 (delta 1), reused 11 (delta 0), pack-reused 0
Получение объектов: 100% (27/27), 16.93 КиБ | 3.39 МБ/с, готово.
Определение изменений: 100% (1/1), готово.
Подмодуль «template/presentation» (https://github.com/yamadharma/academic-presentation-markdown-template.git) зарегистрирован по пути «template/presentation»
Подмодуль «template/report» (https://github.com/yamadharma/academic-laboratory-report-template.git) зарегистрирован по пути «template/report»
Клонирование в «/home/apkim/work/study/2022-2023/Операционные системы/os-intro/template/presentation»...
remote: Enumerating objects: 82, done.
remote: Counting objects: 100% (82/82), done.
remote: Compressing objects: 100% (57/57), done.
remote: Total 82 (delta 28), reused 77 (delta 23), pack-reused 0
Получение объектов: 100% (82/82), 92.90 КиБ | 184.00 КиБ/с, готово.
Определение изменений: 100% (28/28), готово.
Клонирование в «/home/apkim/work/study/2022-2023/Операционные системы/os-intro/template/report»...
remote: Enumerating objects: 101, done.
remote: Counting objects: 100% (101/101), done.
remote: Compressing objects: 100% (78/78), done.
remote: Total 101 (delta 40), reused 88 (delta 27), pack-reused 0
Получение объектов: 100% (101/101), 327.25 КиБ | 726.00 КиБ/с, готово.
Определение изменений: 100% (40/40), готово.
Submodule path 'template/presentation': checked out 'b1be3809ee91f5809264cb755d316174540b753e'
Submodule path 'template/report': checked out '1d1b61dcac9c287a83917b82e3aeffa33b1e3b2'
[apkim@fedora Операционные системы]$
```

Рис. 3.13: Создание репозитория (2)

Теперь нам нужно настроить каталог курса. Сначала переходим в каталог курса, затем удаляем лишние файлы и создаем необходимые каталоги (рис. 3.14).

```
[apkim@fedora Операционные системы]$ cd ~/work/study/2022-2023/"Операционные системы"/os-intro
[apkim@fedora os-intro]$ rm package.json
[apkim@fedora os-intro]$ echo os-intro > COURSE
```

Рис. 3.14: Настройка каталога курса

Отправляем файлы на сервер (рис. 3.15).

```
[apkim@fedora os-intro]$ make
[apkim@fedora os-intro]$ git add .
[apkim@fedora os-intro]$ git commit -am 'feat(main): make course str'
```

Рис. 3.15: Отправка файлов (1)

Продолжение (рис. 3.16).

```
[apkim@fedora os-intro]$ git push
Перечисление объектов: 40, готово.
Подсчет объектов: 100% (40/40), готово.
Сжатие объектов: 100% (38/38), готово.
Запись объектов: 100% (38/38), 342.40 КиБ | 1.30 МБ/с, готово.
Всего 38 (изменений 4), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (4/4), completed with 1 local object.
To github.com:angelinagata/study_2022-2023_os-intro.git
   7d2bd78..c0a3edc  master -> master
[apkim@fedora os-intro]$
```

Рис. 3.16: Отправка файлов (2)

Ответы на контрольные вопросы: 1. 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 поддерживают 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. Например, они могут поддерживать работу с несколькими версиями одного файла, сохраняя общую историю изменений до точки ветвления версий и собственные истории изменений каждой ветви.

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 не полную версию изменённых файлов, а производить так называемую дельта-компрессию — со-

хранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных. Сохранить добавленные изменения с внесением комментария через встроенный редактор: `git commit` 3. Централизованные системы контроля версий представляют собой приложения типа клиент-сервер, когда репозиторий проекта существует в единственном экземпляре и хранится на сервере. Доступ к нему осуществлялся через специальное клиентское приложение. В качестве примеров таких программных продуктов можно привести CVS, Subversion. Распределенные системы контроля версий (Distributed Version Control System, DVCS) позволяют хранить репозиторий (его копию) у каждого разработчика, работающего с данной системой. При этом можно выделить центральный репозиторий (условно), в который будут отправляться изменения из локальных и, с ним же эти локальные репозитории будут синхронизироваться. При работе с такой системой, пользователи периодически синхронизируют свои локальные репозитории с центральным и работают непосредственно со своей локальной копией. После внесения достаточного количества изменений в локальную копию они (изменения) отправляются на сервер. При этом сервер, чаще всего, выбирается условно, т.к. в большинстве DVCS нет такого понятия как “выделенный сервер с центральным репозиторием”. 4. 5. 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 Перечислим 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.

9. Ветка (англ. branch) — это последовательность коммитов, в которой ведётся параллельная разработка какого-либо функционала Основная ветка – master Ветки в GIT. Показать все ветки, существующие в репозитории git branch. Создать ветку git branch имя. Ветки нужны, чтобы несколько программистов могли вести работу над одним и тем же проектом или даже файлом одновременно, при этом не мешая друг другу. Кроме того, ветки используются для тестирования экспериментальных функций: чтобы не повредить

основному проекту, создается новая ветка специально для экспериментов.

## 4 Выводы

В ходе данной лабораторной работы мы изучили идеологию и применение средств контроля версий, а также освоили умения по работе с git.



## **Список литературы**