

# **Отчёт по лабораторной работе №2**

**Управление версиями**

Буянбадрах Тогтохжав

# Содержание

1	Цель работы	4
2	Выполнение лабораторной работы	5
3	Вывод	10
4	Контрольные вопросы	11
	Список литературы	15

## Список иллюстраций

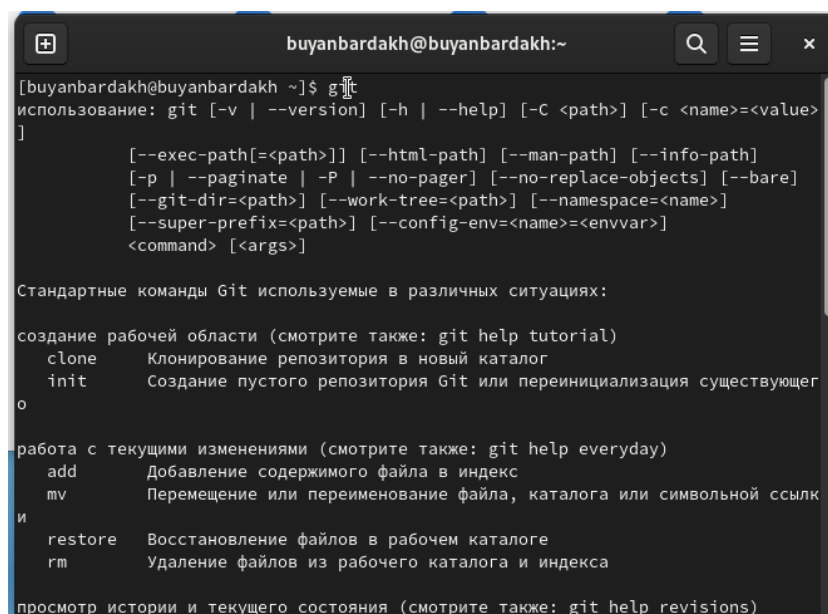
2.1	Загрузка пакетов . . . . .	5
2.2	Параметры репозитория . . . . .	5
2.3	rsa-4096 . . . . .	6
2.4	ed25519 . . . . .	6
2.5	GPG ключ . . . . .	7
2.6	GPG ключ . . . . .	7
2.7	Параметры репозитория . . . . .	8
2.8	Связь репозитория с аккаунтом . . . . .	8
2.9	Загрузка шаблона . . . . .	9
2.10	Первый коммит . . . . .	9

# 1 Цель работы

Целью данной работы является изучение идеологии и применения средств контроля версий и освоение умений работать с git.

## 2 Выполнение лабораторной работы

Устанавливаем git, git-flow и gh.



```
[buyanbardakh@buyanbardakh ~]$ git
использование: git [-v | --version] [-h | --help] [-C <path>] [-c <name>=<value>]
[  

    [--exec-path=<path>] [--html-path] [--man-path] [--info-path]
    [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
    [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
    [--super-prefix=<path>] [--config-env=<name>=<envvar>]
    <command> [<args>]

Стандартные команды Git используемые в различных ситуациях:

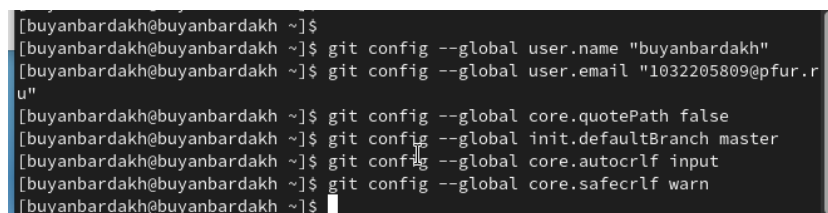
создание рабочей области (смотрите также: git help tutorial)
    clone      Клонирование репозитория в новый каталог
    init       Создание пустого репозитория Git или переинициализация существующег
о

работа с текущими изменениями (смотрите также: git help everyday)
    add        Добавление содержимого файла в индекс
    mv         Перемещение или переименование файла, каталога или символической ссылк
и
    restore    Восстановление файлов в рабочем каталоге
    rm         Удаление файлов из рабочего каталога и индекса

просмотр истории и текущего состояния (смотрите также: git help revisions)
```

Рис. 2.1: Загрузка пакетов

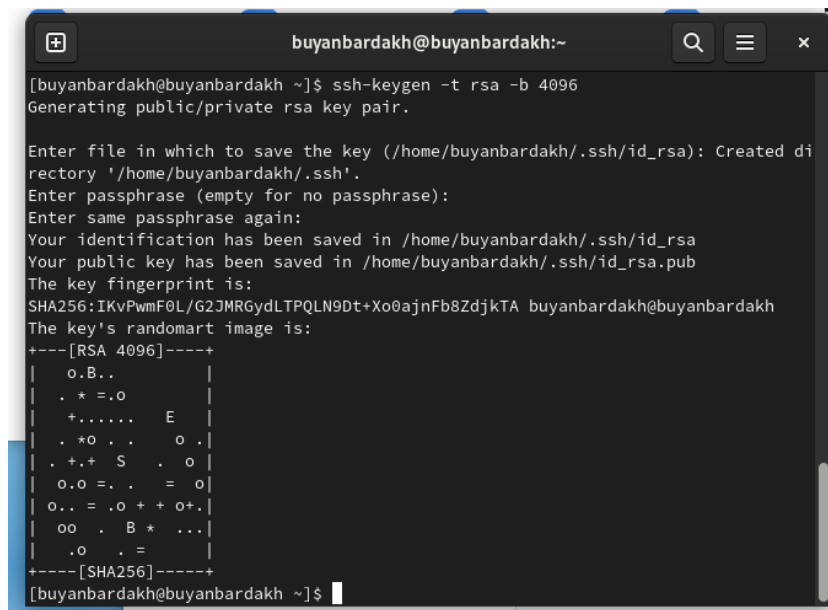
Зададим имя и email владельца репозитория, кодировку и прочие параметры.



```
[buyanbardakh@buyanbardakh ~]$
[buyanbardakh@buyanbardakh ~]$ git config --global user.name "buyanbardakh"
[buyanbardakh@buyanbardakh ~]$ git config --global user.email "1032205809@pfur.ru"
[buyanbardakh@buyanbardakh ~]$ git config --global core.quotePath false
[buyanbardakh@buyanbardakh ~]$ git config --global init.defaultBranch master
[buyanbardakh@buyanbardakh ~]$ git config --global core.autocrlf input
[buyanbardakh@buyanbardakh ~]$ git config --global core.safecrlf warn
[buyanbardakh@buyanbardakh ~]$
```

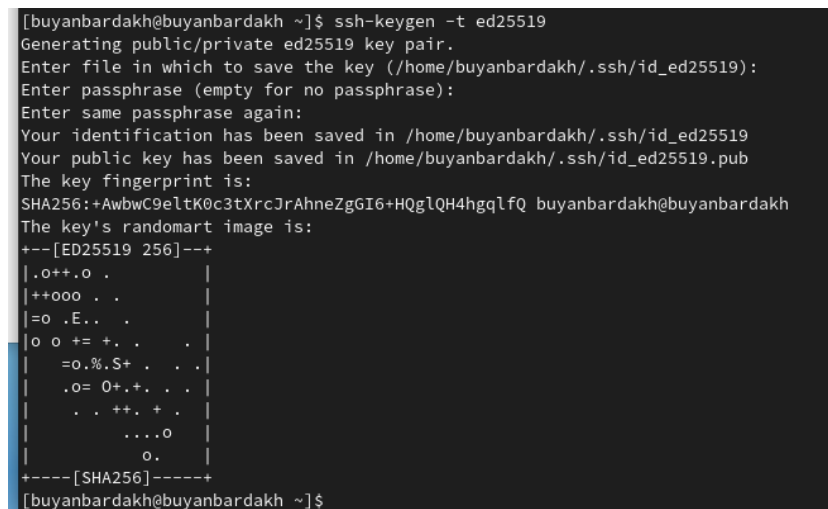
Рис. 2.2: Параметры репозитория

## Создаем SSH ключи



```
buyanbardakh@buyanbardakh:~  
[buyanbardakh@buyanbardakh ~]$ ssh-keygen -t rsa -b 4096  
Generating public/private rsa key pair.  
  
Enter file in which to save the key (/home/buyanbardakh/.ssh/id_rsa): Created directory  
'/home/buyanbardakh/.ssh'.  
Enter passphrase (empty for no passphrase):  
Enter same passphrase again:  
Your identification has been saved in /home/buyanbardakh/.ssh/id_rsa  
Your public key has been saved in /home/buyanbardakh/.ssh/id_rsa.pub  
The key fingerprint is:  
SHA256:IKvPwmF0L/G2JMRGydLTPQLN9Dt+Xo0ajnFb8ZdjKTA buyanbardakh@buyanbardakh  
The key's randomart image is:  
+---[RSA 4096]-----+  
|  o.B.. |  
|  . * =.o |  
|  +..... E |  
|  . *O . . o . |  
|  . .+ S . o |  
|  o.o =. . = o |  
|  o.. = .o + + o+. |  
|  oo . B * ... |  
|  .o . = |  
+---[SHA256]-----+  
[buyanbardakh@buyanbardakh ~]$
```

Рис. 2.3: rsa-4096



```
[buyanbardakh@buyanbardakh ~]$ ssh-keygen -t ed25519  
Generating public/private ed25519 key pair.  
Enter file in which to save the key (/home/buyanbardakh/.ssh/id_ed25519):  
Enter passphrase (empty for no passphrase):  
Enter same passphrase again:  
Your identification has been saved in /home/buyanbardakh/.ssh/id_ed25519  
Your public key has been saved in /home/buyanbardakh/.ssh/id_ed25519.pub  
The key fingerprint is:  
SHA256:+AwbwC9eltK0c3tXrcJrAhneZgGI6+HQg1QH4hgqlfQ buyanbardakh@buyanbardakh  
The key's randomart image is:  
+---[ED25519 256]---+  
|.o++o . |  
|++ooo . . |  
|=o .E.. . |  
|o o += +. . . |  
|=o.%.S+ . . . |  
|.o= 0+.+. . . |  
|. . ++. + . |  
|. . . .o |  
|. . . .o |  
+---[SHA256]-----+  
[buyanbardakh@buyanbardakh ~]$
```

Рис. 2.4: ed25519

## Создаем GPG ключ

```
buyanbardakh@buyanbardakh:~  
Вы выбрали следующий идентификатор пользователя:  
"buyanbardakh <1032205809@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 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: /home/buyanbardakh/.gnupg/trustdb.gpg: создана таблица доверия  
gpg: создан каталог '/home/buyanbardakh/.gnupg/openpgp-revocs.d'  
gpg: сертификат отзыва записан в '/home/buyanbardakh/.gnupg/openpgp-revocs.d/984  
F971D7FE58F04BFE7ADD038170D7268EA93A8.rev'.  
открытый и секретный ключи созданы и подписаны.  
  
pub   rsa4096 2023-03-01 [SC]  
      984F971D7FE58F04BFE7ADD038170D7268EA93A8  
uid    buyanbardakh <1032205809@pfur.ru>  
sub    rsa4096 2023-03-01 [E]  
  
[buyanbardakh@buyanbardakh ~]$
```

Рис. 2.5: GPG ключ

## Добавляем GPG ключ в аккаунт

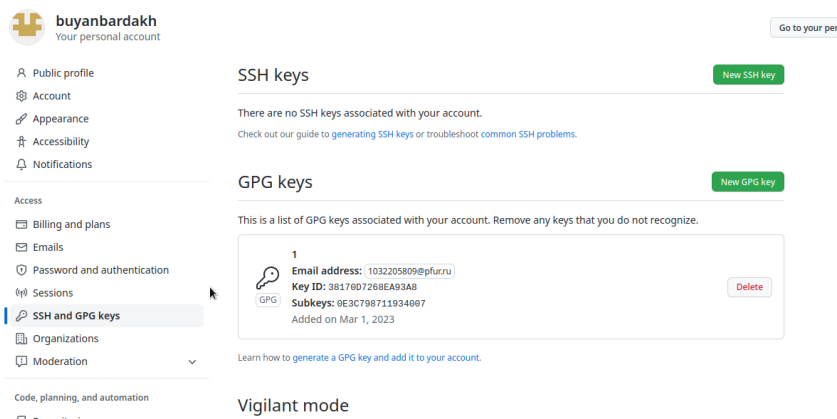


Рис. 2.6: GPG ключ

## Настройка автоматических подписей коммитов git

```
[buyanbardakh@buyanbardakh ~]$  
[buyanbardakh@buyanbardakh ~]$ gpg --list-secret-keys --keyid-format LONG  
/home/buyanbardakh/.gnupg/pubring.kbx  
-----  
sec   rsa4096/38170D7268EA93A8 2023-03-01 [SC]  
      984F971D7FE58F04BFE7ADD038170D7268EA93A8  
uid    [ абсолютно ] buyanbardakh <1032205809@pfur.ru>  
ssb    rsa4096/0E3C798711934007 2023-03-01 [E]  
  
[buyanbardakh@buyanbardakh ~]$ git config --global user.signingKey 38170D7268EA93A8  
[buyanbardakh@buyanbardakh ~]$ git config --global commit.gpgSign true  
[buyanbardakh@buyanbardakh ~]$ git config --global gpg.program $(which gpg2)  
[buyanbardakh@buyanbardakh ~]$
```

Рис. 2.7: Параметры репозитория

## Настройка gh

```
[buyanbardakh@buyanbardakh ~]$ git config --global gpg.program $(which gpg2)  
[buyanbardakh@buyanbardakh ~]$ gh auth login  
? What account do you want to log into? GitHub.com  
? What is your preferred protocol for Git operations? SSH  
? Upload your SSH public key to your GitHub account? /home/buyanbardakh/.ssh/id_rsa.pub  
? Title for your SSH key: GitHub CLI  
? How would you like to authenticate GitHub CLI? Login with a web browser  
  
! First copy your one-time code: 2D8C-E94E  
Press Enter to open github.com in your browser...  
✓ Authentication complete.  
- gh config set -h github.com git_protocol ssh  
✓ Configured git protocol  
✓ Uploaded the SSH key to your GitHub account: /home/buyanbardakh/.ssh/id_rsa.pub  
✓ Logged in as buyanbardakh  
[buyanbardakh@buyanbardakh ~]$
```

Рис. 2.8: Связь репозитория с аккаунтом

## Загрузка шаблона репозитория и синхронизация



```
buyanbardakh@buyanbardakh:~/work/study/2022-2023/Опер...
tation-markdown-template.git) зарегистрирован по пути «template/presentation»
Подмодуль «template/report» (https://github.com/yamadharma/academic-laboratory-r
report-template.git) зарегистрирован по пути «template/report»
Клонирование в «/home/buyanbardakh/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 | 2.51 МиБ/с, готово.
Определение изменений: 100% (28/28), готово.
Клонирование в «/home/buyanbardakh/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% (70/70), done.
remote: Total 101 (delta 40), reused 88 (delta 27), pack-reused 0
Получение объектов: 100% (101/101), 327.25 КиБ | 481.00 КиБ/с, готово.
Определение изменений: 100% (40/40), готово.
Submodule path 'template/presentation': checked out 'b1be380ee91f5809264cb755d3
16174540b753e'
Submodule path 'template/report': checked out '1d1b61dcac9c287a83917b82e3aef11a3
3b1e3b2'
[buyanbardakh@buyanbardakh Операционные системы]$
```

Рис. 2.9: Загрузка шаблона

## Подготовка репозитория и коммит изменений

```
buyanbardakh@buyanbardakh:~/work/study/2022-2023/Опер...
create mode 100755 project-personal/stage6/report/pandoc/filters/pandoc_secnos.
py
create mode 100755 project-personal/stage6/report/pandoc/filters/pandoc_tableno
s.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/__i
nit__.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/cor
e.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/mai
n.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/pan
docattributes.py
create mode 100644 project-personal/stage6/report/report.md
Перечисление объектов: 38, готово.
Подсчет объектов: 100% (38/38), готово.
При сжатии изменений используется до 4 потоков
Сжатие объектов: 100% (30/30), готово.
Запись объектов: 100% (37/37), 343.00 КиБ | 2.74 МиБ/с, готово.
Всего 37 (изменений 4), повторно использовано 0 (изменений 0), повторно использо
вано пакетов 0
remote: Resolving deltas: 100% (4/4), completed with 1 local object.
To github.com:buyanbardakh/os-intro.git
f092ff1..073876d master -> master
[buyanbardakh@buyanbardakh os-intro]$
```

Рис. 2.10: Первый коммит

## **3 Вывод**

Мы приобрели практические навыки работы с сервисом github.

## 4 Контрольные вопросы

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

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

2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.

- хранилище - пространство на накопителе где расположен репозиторий
- commit - сохранение состояния хранилища
- история - список изменений хранилища (коммитов)
- рабочая копия - локальная копия сетевого репозитория, в которой работает программист. Текущее состояние файлов проекта, основанное на версии, загруженной из хранилища (обычно на последней)

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

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. Опишите действия с VCS при единоличной работе с хранилищем.

Один пользователь работает над проектом и по мере необходимости делает коммиты, сохраняя определенные этапы.

#### 5. Опишите порядок работы с общим хранилищем VCS.

Несколько пользователей работают каждый над своей частью проекта. При этом каждый должен работать в своей ветки. При завершении работы ветка пользователя сливается с основной веткой проекта.

#### 6. Каковы основные задачи, решаемые инструментальным средством git?

- Ведение истории версий проекта: журнал (log), метки (tags), ветвления (branches).

- Работа с изменениями: выявление (diff), слияние (patch, merge).
- Обеспечение совместной работы: получение версии с сервера, загрузка обновлений на сервер.

7. Назовите и дайте краткую характеристику командам git.

- git config - установка параметров
- git status - полный список изменений файлов, ожидающих коммита
- git add . - сделать все измененные файлы готовыми для коммита.
- git commit -m "[descriptive message]" - записать изменения с заданным сообщением.
- git branch - список всех локальных веток в текущей директории.
- git checkout [branch-name] - переключиться на указанную ветку и обновить рабочую директорию.
- git merge [branch] — соединить изменения в текущей ветке с изменениями из заданной.
- git push - запустить текущую ветку в удаленную ветку.
- git pull - загрузить историю и изменения удаленной ветки и произвести слияние с текущей веткой.

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

- git remote add [имя] [url] — добавляет удалённый репозиторий с заданным именем;
- git remote remove [имя] — удаляет удалённый репозиторий с заданным именем;
- git remote rename [старое имя] [новое имя] — переименовывает удалённый репозиторий;
- git remote set-url [имя] [url] — присваивает репозиторию с именем новый адрес;

- `git remote show [имя]` — показывает информацию о репозитории.

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

Ветвление — это возможность работать над разными версиями проекта: вместо одного списка с упорядоченными коммитами история будет расходиться в определённых точках. Каждая ветвь содержит легковесный указатель HEAD на последний коммит, что позволяет без лишних затрат создать много веток. Ветка по умолчанию называется `master`, но лучше назвать её в соответствии с разрабатываемой в ней функциональностью.

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

Зачастую нам не нужно, чтобы Git отслеживал все файлы в репозитории, потому что в их число могут входить:

# Список литературы

1. Лекция Системы контроля версий
2. GitHub для начинающих