

# **Отчёт по лабораторной работе №2**

**Управление версиями**

Ахмаров Роман Рафаильевич

# Содержание

1	Цель работы	4
2	Выполнение лабораторной работы	5
3	Вывод	9
4	Контрольные вопросы	10

# List of Figures

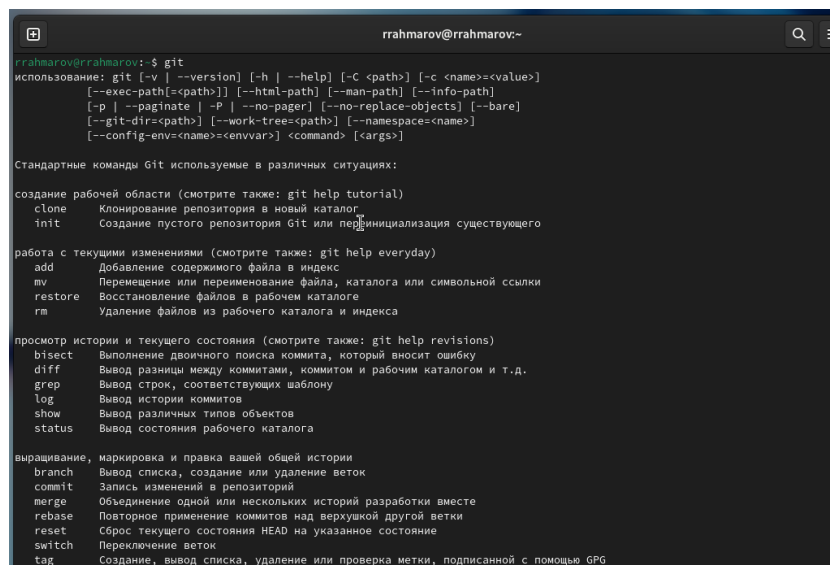
2.1	Загрузка пакетов . . . . .	5
2.2	Параметры репозитория . . . . .	5
2.3	rsa-4096 . . . . .	6
2.4	ed25519 . . . . .	6
2.5	GPG ключ . . . . .	7
2.6	GPG ключ . . . . .	7
2.7	Параметры репозитория . . . . .	7
2.8	Связь репозитория с аккаунтом . . . . .	8
2.9	Загрузка шаблона . . . . .	8
2.10	Первый коммит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

# 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.



```
rrahmarov@rrahmarov:~$ 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>]
               [--config-env=<name>=<envvar>] <command> [<args>]

Стандартные команды Git используются в различных ситуациях:

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

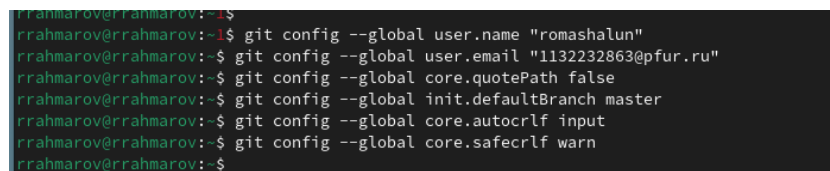
работа с текущими изменениями (смотрите также: git help everyday)
  add    Добавление содержимого файла в индекс
  mv     Перемещение или переименование файла, каталога или символической ссылки
  restore Восстановление файлов в рабочем каталоге
  rm     Удаление файлов из рабочего каталога и индекса

просмотр истории и текущего состояния (смотрите также: git help revisions)
  bisect  Выполнение двоичного поиска коммита, который вносит ошибку
  diff    Вывод разницы между коммитами, коммитом и рабочим каталогом и т.д.
  grep    Вывод строк, соответствующих шаблону
  log     Вывод истории коммитов
  show    Вывод различных типов объектов
  status  Вывод состояния рабочего каталога

выращивание, маркировка и правка вашей общей истории
  branch  Вывод списка, создание или удаление веток
  commit  Запись изменений в репозиторий
  merge   Объединение одной или нескольких историй разработки вместе
  rebase  Повторное применение коммитов над вершущей другой ветки
  reset   Сброс текущего состояния HEAD на указанное состояние
  switch  Переключение веток
  tag     Создание, вывод списка, удаление или проверка метки, подписанной с помощью GPG
```

Figure 2.1: Загрузка пакетов

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



```
rrahmarov@rrahmarov:~$ git config --global user.name "romashalun"
rrahmarov@rrahmarov:~$ git config --global user.email "1132232863@pfur.ru"
rrahmarov@rrahmarov:~$ git config --global core.quotePath false
rrahmarov@rrahmarov:~$ git config --global init.defaultBranch master
rrahmarov@rrahmarov:~$ git config --global core.autocrlf input
rrahmarov@rrahmarov:~$ git config --global core.safecrlf warn
rrahmarov@rrahmarov:~$
```

Figure 2.2: Параметры репозитория

Создаем SSH ключи

```

rrahmarov@rrahmarov:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/rrahmarov/.ssh/id_rsa):
Created directory '/home/rrahmarov/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/rrahmarov/.ssh/id_rsa
Your public key has been saved in /home/rrahmarov/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:H6CHlrvZ+BP3pRp+XgHN4woAKmIhJNEQnLSu8Qm4y0 rrahmarov@rrahmarov
The key's randomart image is:
+---[RSA 4096]-----+
|  =+oo. .          |
| 0=+.+. . = o     |
| +*o*.  O o o + .  |
| . E..  + * o + +  |
|   o o S = . =    |
|   . . = = .      |
|   . + +          |
|   o              |
|                   |
+-----[SHA256]-----+
rrahmarov@rrahmarov:~$

```

Figure 2.3: rsa-4096

```

rrahmarov@rrahmarov:~$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/rrahmarov/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/rrahmarov/.ssh/id_ed25519
Your public key has been saved in /home/rrahmarov/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:8fVZ9w12/fcqxInGEV1eyze24CNBexXvKfFcgj+DEzA rrahmarov@rrahmarov
The key's randomart image is:
+--[ED25519 256]--+
|      Eo ..oo|
|      ooooo.+|
|      .  +++=0|
|      o..**B@|
|      S..o=**|
|      + =.ooo|
|      . . .|
|      . . |
|      .. |
+-----[SHA256]-----+
rrahmarov@rrahmarov:~$

```

Figure 2.4: ed25519

Создаем GPG ключ

```

Случайных чисел больше, возмозможности получить достаточное количество энтропии.
gpg: /home/rrahmarov/.gnupg/trustdb.gpg: создана таблица доверия
gpg: создан каталог '/home/rrahmarov/.gnupg/openpgp-revocs.d'
gpg: серийный отпечаток записан в '/home/rrahmarov/.gnupg/openpgp-revocs.d/0FF1CAB8F3ED04AF3F09127BEC0FB6D726C9476.rev'.
открытый и секретный ключи созданы и подписаны.

pub   rsa4096 2024-02-20 [SC]
      0FF1CAB8F3ED04AF3F09127BEC0FB6D726C9476
uid          romashalun <1132232863@pfur.ru>
sub     rsa4096 2024-02-20 [E]

rrahmarov@rrahmarov:~$ 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
[keybox] I
-----
sec   rsa4096/CC0FB6D726C9476 2024-02-20 [SC]
      0FF1CAB8F3ED04AF3F09127BEC0FB6D726C9476
uid          [ а6солотно ] romashalun <1132232863@pfur.ru>
ssb     rsa4096/2ED0BCF0904DE197 2024-02-20 [E]

rrahmarov@rrahmarov:~$

```

Figure 2.5: GPG ключ

## Добавляем GPG ключ в аккаунт

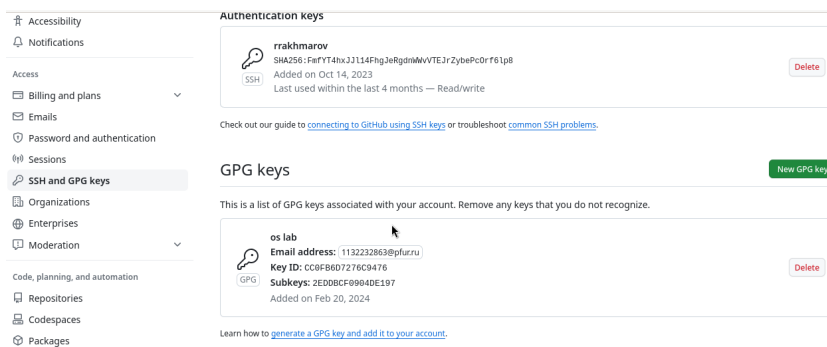


Figure 2.6: GPG ключ

## Настройка автоматических подписей коммитов git

```
ecZ9g5Uixh8ZPGPaApz6u0GWrAOcJMIohV+NdLvG9XZH7eQhUUAZEtnQ+1f0RK
Hg9eh6IuIWdfoLuuHTv6FekVwsHDMs2f0bwfX0wxCrqEF6WncL3Nv049yfXI13cc
e0swxJp8012Dq0jyp2sK6/Cef0FPJrzfg4MdmnhTJL/w2a4B9Z91luaOGPINhKA
B22PdbhzR0JMeTreD9azyQDtFSMUxNmR3G0DIGXBGVDiLoDU0l0K0mcRJB9G+YqG
311fwAGtVe/kn8dySfpWn4P6r0M2uEdPKsrxiAxxKnv3TXVm41RVL94uu7pC7j3j
qVqoPv7ZyZf4pAbHZYxCutFGp7QRkDI/4Jfpr04Bmswue3Sb8QzofufqcA37xzE
qhWdd84r5B52ytffj8JIF6Fu3S34YU9aa02YfJ0NFNVhVn9LPD3i2D6HxjMRW3g==
=FB97
-----END PGP PUBLIC KEY BLOCK-----
rrahmarov@rrahmarov:~$
rrahmarov@rrahmarov:~$
rrahmarov@rrahmarov:~$ git config --global user.signingkey CC0FB6D7276C9476
rrahmarov@rrahmarov:~$ git config --global commit.gpgsign true
rrahmarov@rrahmarov:~$ git config --global gpg.program $(which gpg2)
rrahmarov@rrahmarov:~$
```

Figure 2.7: Параметры репозитория

## Настройка gh

```

rrahmarov@rrahmarov:~$
rrahmarov@rrahmarov:~$ gh auth login
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations on this host? SSH
? Upload your SSH public key to your GitHub account? /home/rrahmarov/.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: 9067-6BD0
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/rrahmarov/.ssh/id_rsa.pub
✓ Logged in as romashalun
rrahmarov@rrahmarov:~$ mkdir -p ~/work/study/2023-2024/"Операционные системы"
rrahmarov@rrahmarov:~$ cd ~/work/study/2023-2024/"Операционные системы"
rrahmarov@rrahmarov:~/work/study/2023-2024/Операционные системы$ gh repo create os-intro --template=yamadharma/course-directory-student-template --public
✓ Created repository romashalun/os-intro on GitHub
rrahmarov@rrahmarov:~/work/study/2023-2024/Операционные системы$

```

Figure 2.8: Связь репозитория с аккаунтом

## Загрузка шаблона репозитория и синхронизация

```

Клонирование в «/home/rrahmarov/work/study/2023-2024/Операционные системы/os-intro/template/report»...
remote: Enumerating objects: 126, done.
remote: Counting objects: 100% (126/126), done.
remote: Compressing objects: 100% (87/87), done.
remote: Total 126 (delta 52), reused 108 (delta 34), pack-reused 0
Получение объектов: 100% (126/126), 335.80 Кб | 2.82 Мб/с, готово.
Определение изменений: 100% (52/52), готово.
Submodule path 'template/presentation': checked out '40a1761813e197d00e8443ff1ca72c68a304f24c'
Submodule path 'template/report': checked out '7c31ab8e5dfa8cbb2d67caeb8a19ef8028ced88e'
rrahmarov@rrahmarov:~/work/study/2023-2024/Операционные системы$ cd ~/work/study/2023-2024/"Операционные системы"/os-intro
rrahmarov@rrahmarov:~/work/study/2023-2024/Операционные системы/os-intro$ rm package.json
rrahmarov@rrahmarov:~/work/study/2023-2024/Операционные системы/os-intro$ make COURSE=os-intro prepare
rrahmarov@rrahmarov:~/work/study/2023-2024/Операционные системы/os-intro$ ls
CHANGELOG.md  COURSE  LICENSE  prepare  project-personal  README.git-flow.md  template
config         labs   Makefile  presentation  README.en.md      README.md
rrahmarov@rrahmarov:~/work/study/2023-2024/Операционные системы/os-intro$

```

Figure 2.9: Загрузка шаблона

## Подготовка репозитория и коммит изменений

```

create mode 100755 project-personal/stage6/report/pandoc/filters/pandoc_eqnos.py
create mode 100755 project-personal/stage6/report/pandoc/filters/pandoc_fignos.py
create mode 100755 project-personal/stage6/report/pandoc/filters/pandoc_secnos.py
create mode 100755 project-personal/stage6/report/pandoc/filters/pandoc_tablenos.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/__init__.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/core.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/main.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/pandocattributes.py
create mode 100644 project-personal/stage6/report/report.md
rrahmarov@rrahmarov:~/work/study/2023-2024/Операционные системы/os-intro$ git push
Перечисление объектов: 38, готово.
Подсчет объектов: 100% (38/38), готово.
При сжатии изменений используется до 6 потоков
Сжатие объектов: 100% (30/30), готово.
Запись объектов: 100% (37/37), 342.06 Кб | 3.23 Мб/с, готово.
Всего 37 (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:romashalun/os-intro.git
e84e279..01b1749 master -> master
rrahmarov@rrahmarov:~/work/study/2023-2024/Операционные системы/os-intro$

```

Figure 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: