

Отчёт по лабораторной работе №2

Управление версиями

Когенгар Ришард

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Вывод	17
4	Контрольные вопросы	18

Список иллюстраций

2.1	Загрузка пакетов	7
2.2	Параметры репозитория	8
2.3	rsa-4096	9
2.4	ed25519	10
2.5	GPG ключ	11
2.6	GPG ключ	12
2.7	Параметры репозитория	13
2.8	Связь репозитория с аккаунтом	14
2.9	Загрузка шаблона	15
2.10	Первый коммит	16

Список таблиц

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.

```
rishard@rishardkogengar:~$ 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] [--no-lazy-fetch]
                [--no-optional-locks] [--no-advice] [--bare] [--git-dir=<path>]
                [--work-tree=<path>] [--namespace=<name>] [--config-env=<name>=<envvar>]
                <command> [<args>]
```

Стандартные команды Git используемые в различных ситуациях:

создание рабочей области (смотрите также: `git help tutorial`)

<code>clone</code>	Клонирование репозитория в новый каталог
<code>init</code>	Создание пустого репозитория Git или переинициализация существующего

работа с текущими изменениями (смотрите также: `git help everyday`)

<code>add</code>	Добавление содержимого файла в индекс
<code>mv</code>	Перемещение или переименование файла, каталога или символической ссылки
<code>restore</code>	Восстановление файлов в рабочем каталоге
<code>rm</code>	Удаление файлов из рабочего каталога и индекса

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

<code>bisect</code>	Выполнение двоичного поиска коммита, который вносит ошибку
<code>diff</code>	Вывод разницы между коммитами, коммитом и рабочим каталогом и т.д.
<code>grep</code>	Вывод строк, соответствующих шаблону
<code>log</code>	Вывод истории коммитов
<code>show</code>	Вывод различных типов объектов
<code>status</code>	Вывод состояния рабочего каталога

Рис. 2.1: Загрузка пакетов

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

```
rishard@rishardkogengar:~$  
rishard@rishardkogengar:~$ git config --global user.name "RishardKogengar"  
rishard@rishardkogengar:~$ git config --global user.email "1032239631@rudn.university"  
rishard@rishardkogengar:~$ git config --global core.quotepath false  
rishard@rishardkogengar:~$ git config --global init.defaultBranch master  
rishard@rishardkogengar:~$ git config --global core.autocrlf input  
rishard@rishardkogengar:~$ git config --global core.safecrlf warn  
rishard@rishardkogengar:~$
```

Рис. 2.2: Параметры репозитория

Создаем SSH ключи


```

rishard@rishardkogengar:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/rishard/.ssh/id_rsa):
Created directory '/home/rishard/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/rishard/.ssh/id_rsa
Your public key has been saved in /home/rishard/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:YoxqKK66bHcdbP6IuWXVUiqjpFPXvvr10txid6AW7ac rishard@rishardkogengar
The key's randomart image is:
+---[RSA 4096]-----+
|
|
|
|
| 0 . +
| .0=+S= 0.
| . .+.0=+ +..0
| 0 00 .+0....=..
| +0 ...=0. .Bo+ 0
| B+. .+...0+0+E+
+---[SHA256]-----+
rishard@rishardkogengar:~$

```

Рис. 2.3: rsa-4096

```

rishard@rishardkogengar:~$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/rishard/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/rishard/.ssh/id_ed25519
Your public key has been saved in /home/rishard/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:oo7R3Th1we0xGcExEuyGzcX86fWhfoRb4n1si3Y+0Us rishard@rishardkogengar
The key's randomart image is:
+--[ED25519 256]--+
|      .00=0      |
|      ..0.+      |
|      0o =.      |
|      . 0ooo..   |
|      . S..0..0..|
|      . 0 = . .+ 0.|
|      . 0 + . 0 *E.|
|      + .      =0B+|
|      . .      ..+*=|
+----[SHA256]-----+
rishard@rishardkogengar:~$

```

Рис. 2.4: ed25519

Создаем GPG ключ

```

rishard@rishardkogengar:~$ gpg --full-generate-key
gpg (GnuPG) 2.4.5; Copyright (C) 2024 g10 Code GmbH
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 '/home/rishard/.gnupg'
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
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: 

```

Рис. 2.5: GPG ключ

Добавляем GPG ключ в аккаунт

```

rishard@rishardkogengar:~$ 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
[keyboard]
-----
sec   rsa4096/D2D1B59D7C8319D6 2025-02-17 [SC]
      962D1E5283BD00B8ADAE14FED2D1B59D7C8319D6
uid           [ абсолютно ] RishardKogengar <1032239631@rudn.university>
ssb   rsa4096/8A94112420B6DFEE 2025-02-17 [E]

rishard@rishardkogengar:~$ gpg --armor --export D2D1B59D7C8319D6
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBGezIX8BEACsjXIvJuRr8h5prBwN0qosUcboVBPEShgRXFMDlVPO/bn0r2J4
iFWKL/+LhtCzf5wyhq0y3lcUJDlbANH2+JmJiL933K5rGRHwmsZIVuRw9oQTjAhU
RXBSUE+JGpIfdlFVgt/mrQjNR4boq5R9vy/cSJtwr6dbiIEG2ZCnYdNHE46HVV3Z
dAG6J0c/gW+Vvlt5eGLjDqTYz1ew0PVs878v3NgiR1HBpJS1vvyWvFqtgPHFP2oD
Ch3K//8d29FJps5eI1JQB4Iwhm29iahjPRWinBq1Vr+xi4yQICZA/y2eB6XoPqQQ
OWMTo/mQZHMieH9sBUE3pGGNGqqEQ5yuQFg3kCMfWFFjbushHPxdx6l99JjulwB5s
fsV8CBzycovEy3ieXyA79SL7N/B0V42jVEGN9uImump3srtETg22Fiao7NI11KLE
eDMiH837h01B6gjkV+A3iJQq/u9Ergned0XU/nEsD5zNySwNxr6Lud68tqVL4z/7
YJk7eyl8VbFWi47YuS1pFIhat2gr3wipfzSmAr9vyfft2B4Ct/lWazz8GEebrSFk
hw+YhdDHBiBku9kM2vWl7ZLqKX18mmLnz6X9vN1RmaU5hRNvzP1+XkC7Loefdosi
5zDsVHXn07I06cwRSEFCovVW7z4Vn/nlUW7eFEpSkTaKDRGf99aQw7tie0ARA0AB

```

Рис. 2.6: GPG ключ

Настройка автоматических подписей коммитов git

```

47M1TfC7WdH5KJ0L0p0aKa13Ttq142Q0yXQ4+g/7VZIKKKKQ00Q52p012D701
mjrf1Lq3M4sywMTqqkbHzNLe3PuM+UMhPHRE5AEhJRFCppBc1Np5XQCqPqQJVjx
DDT00tgGbVBcCudee+DMUwH5/oVV9KRpy0g8nmS002DZLjKAm21WTgHkXkHVVCfV
Q26ENeiYgM9Yu9oR6i0kjIq91m4RoSTl98APts2jTOLdXqWE/nEB/K0suIzkLDWZ
Yhw11CRLRttELCB5nShXb6iLeS+IavNlgOR7IpuN8py9DjsanphHwWp7i327awbL
jso4112j5ezDiorq0svRez2iLynpjCMn4tcRWUavZc/GY6l5WXWHc5jZf1Wp8gkD
aNo/V3/VTgoquxeiGORcCMKEVOq0TgCCRCcGRwHRVZ0gx29xYSvjR8Qa4UmOP3BB
/IM8CX9Y7C+0qMA=
=kafG
-----END PGP PUBLIC KEY BLOCK-----

rishard@rishardkogengar:~$ git config --global user.signingkey D2D1B59D7C8319D6
rishard@rishardkogengar:~$ git config --global commit.gpgsign true
rishard@rishardkogengar:~$ git config --global gpg.program $(which gpg2)
rishard@rishardkogengar:~$

```

Рис. 2.7: Параметры репозитория

Настройка gh

```
rishard@rishardkogengar:~$ gh auth login
? Where do you use GitHub? GitHub.com
? What is your preferred protocol for Git operations on this host? SSH
? Upload your SSH public key to your GitHub account? /home/rishard/.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: 09EF-C224
Press Enter to open https://github.com/login/device 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/rishard/.ssh/id_rsa.pub
✓ Logged in as RishardKogengar
rishard@rishardkogengar:~$
```

Рис. 2.8: Связь репозитория с аккаунтом

Загрузка шаблона репозитория и синхронизация

```
rishard@rishardkogengar:~$  
rishard@rishardkogengar:~$ mkdir -p ~/work/study/2024-2025/"Операционные системы"  
rishard@rishardkogengar:~$ cd ~/work/study/2024-2025/"Операционные системы"  
rishard@rishardkogengar:~/work/study/2024-2025/Операционные системы$ gh repo create os-intro --template=y  
amadharma/course-directory-student-template --public  
✓ Created repository RishardKogengar/os-intro on GitHub  
https://github.com/RishardKogengar/os-intro  
rishard@rishardkogengar:~/work/study/2024-2025/Операционные системы$ git clone --recursive git@github.com  
:RishardKogengar/os-intro.git os-intro  
Клонирование в «os-intro»...  
The authenticity of host 'github.com (140.82.121.3)' can't be established.  
ED25519 key fingerprint is SHA256:+DiY3wvV6TuJJhbpZisF/zLDA0zPMsvHdkr4UvCOqU.  
This key is not known by any other names.  
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

Рис. 2.9: Загрузка шаблона

Подготовка репозитория и коммит изменений

```

create mode 100644 project-personal/stage6/report/image/placeimg_800_600_tech.jpg
create mode 100644 project-personal/stage6/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
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
rishard@rishardkogengar:~/work/study/2024-2025/Операционные системы/os-intro$ git push
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), 342.28 КБ | 2.80 МБ/с, готово.
Total 37 (delta 4), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (4/4), completed with 1 local object.
To github.com:RishardKogengar/os-intro.git
   6c03515..050e197  master -> master
rishard@rishardkogengar:~/work/study/2024-2025/Операционные системы/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: