

Отчёт по лабораторной работе №2

Управление версиями

Касканте Родригес Альберто

Содержание

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

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	8
2.8	Связь репозитория с аккаунтом	8
2.9	Загрузка шаблона	8
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.

```
albertokaskante@albertokaskante:~$ 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)
clone      Клонирование репозитория в новый каталог
init       Создание пустого репозитория Git или переинициализация 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 верхушкой другой ветки
reset      Сброс текущего состояния HEAD на указанное состояние
switch     Переключение ветки
```

Figure 2.1: Загрузка пакетов

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

```
albertokaskante@albertokaskante:~$ git config --global user.name "AlbertoRUDN"
albertokaskante@albertokaskante:~$ git config --global user.email "1132215059@rudn.university"
albertokaskante@albertokaskante:~$ git config --global core.quotepath false
albertokaskante@albertokaskante:~$ git config --global init.defaultBranch master
albertokaskante@albertokaskante:~$ git config --global core.autocrlf input
albertokaskante@albertokaskante:~$ git config --global core.safecrlf warn
albertokaskante@albertokaskante:~$
```

Figure 2.2: Параметры репозитория

Создаем SSH ключи

```
albertokaskante@albertokaskante:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/albertokaskante/.ssh/id_rsa):
Created directory '/home/albertokaskante/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/albertokaskante/.ssh/id_rsa
Your public key has been saved in /home/albertokaskante/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:eYxKSJK+IqJjoXBAsIpMkX2dNorRbimwM1cL0o0VEZY albertokaskante@albertokaskante
The key's randomart image is:
+---[RSA 4096]-----+
|
| ..E
| * * .
|.* 0 * = +
|o.* 0 * S o
|.0 = B . .
|X B o .
|0= .
|+..
+---[SHA256]-----+
albertokaskante@albertokaskante:~$
```

Figure 2.3: rsa-4096

```
albertokaskante@albertokaskante:~$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/albertokaskante/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/albertokaskante/.ssh/id_ed25519
Your public key has been saved in /home/albertokaskante/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:eI45QN20TuurRyrNOPA+ljoSP5owSHTzYl4J/CN4ae10 albertokaskante@albertokaskante
The key's randomart image is:
+--[ED25519 256]--+
|
| . o .
| = ... +
|+ *.o + o
|.+. . S
|.o+.+ E .
| .o # o
|. = = B
|oo= o. +
+---[SHA256]-----+
albertokaskante@albertokaskante:~$
```

Figure 2.4: ed25519

Создаем GPG ключ

```
albertokaskante@albertokaskante:~$
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: AlbertoRUDN
Адрес электронной почты: 1132215059@rudn.university
Примечание:
Вы выбрали следующий идентификатор пользователя:
  "AlbertoRUDN <1132215059@rudn.university>"

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/albertokaskante/.gnupg/trustdb.gpg: создана таблица доверия
gpg: создан каталог '/home/albertokaskante/.gnupg/openpgp-revocs.d'
gpg: сертификат отзыва записан в '/home/albertokaskante/.gnupg/openpgp-revocs.d/665F4AFF00
ED2EC708A22529AF4EF576A10BACC7.rev'.
открытый и секретный ключи созданы и подписаны.

pub   rsa4096 2024-12-25 [SC]
       665F4AFF00ED2EC708A22529AF4EF576A10BACC7
uid     [ абсолютно ] AlbertoRUDN <1132215059@rudn.university>
sub     rsa4096 2024-12-25 [E]

albertokaskante@albertokaskante:~$
```

Figure 2.5: GPG ключ

Добавляем GPG ключ в аккаунт

```
albertokaskante@albertokaskante:~$ 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/AF4EF576A10BACC7 2024-12-25 [SC]
       665F4AFF00ED2EC708A22529AF4EF576A10BACC7
uid     [ абсолютно ] AlbertoRUDN <1132215059@rudn.university>
ssb     rsa4096/B6691A0E0D210A49 2024-12-25 [E]

albertokaskante@albertokaskante:~$ gpg --armor --export AF4EF576A10BACC7
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBGdsBiMBEACmsKIKWdLMl0l2et0A5zwnRrr8GqmaWkrIPCHRR8AchJAnpRpU
cVyEDcXU0zVsGRCzw0sXyWN6QHJr5kB0r/vzx2yucGoQ09szD3vKAtFW99ZRZeI2
PfcYHqF4vBqQas5G2ovyXn6APCS+QUq0GI/qt2MB09h59FbbrfinQIBkuve4TGF
RJYC5cBeerrZpjyD7ji5dWiySsJytug//wtJpw99yPqyKyx1kyqdYoyfez30pnif
d6rmIQGtjn5c4E4lTTLkYb63oUgyrTet4A+KdwmR53t1eHbvc+oo76Ml3M0kjJH/
XVAqH3WrqEteYf3ze2QGRJFBcnmeSTGwm76Kn37fzDKljMeeqli6NzC44fsV+1ho
wx2QecRZXH/WkD2BKB+5xXHgvcsZOI59eCLjis7u+3nExdaMGvz5VFbpcc52036k
xAToWISQGs2Gk308IAjpMuJnIp9IwX1xQumsBaWEqtpX9DbM7l8dudLenAF0Hyku
bIOgIXcbSUTKwFcksuM0emE8B0yIDynG9HRKNRvviOmmP0l9myXtHAN/KcFD0nW6
```

Figure 2.6: GPG ключ

Настройка автоматических подписей коммитов git

```
albertokaskante@albertokaskante:~$  
albertokaskante@albertokaskante:~$ git config --global user.signingkey 'AF4EF576A10BACC7'  
albertokaskante@albertokaskante:~$ git config --global commit.gpgsign true  
albertokaskante@albertokaskante:~$ git config --global gpg.program $(which gpg2)  
albertokaskante@albertokaskante:~$
```

Figure 2.7: Параметры репозитория

Настройка gh

```
albertokaskante@albertokaskante:~$  
albertokaskante@albertokaskante:~$ 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/albertokaskante/.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: EC50-E446  
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/albertokaskante/.ssh/id_rsa.pub  
✓ Logged in as AlbertoRUDN  
albertokaskante@albertokaskante:~$
```

Figure 2.8: Связь репозитория с аккаунтом

Загрузка шаблона репозитория и синхронизация

```
albertokaskante@albertokaskante:~$ mkdir -p ~/work/study/2023-2024/"Операционные системы"  
albertokaskante@albertokaskante:~$ cd ~/work/study/2023-2024/"Операционные системы"  
albertokaskante@albertokaskante:~/work/study/2023-2024/Операционные системы$ gh repo creat  
e os-intro --template=yamadharma/course-directory-student-template --public  
✓ Created repository AlbertoRUDN/os-intro on GitHub  
https://github.com/AlbertoRUDN/os-intro  
albertokaskante@albertokaskante:~/work/study/2023-2024/Операционные системы$ git clone --r  
ecursive git@github.com:AlbertoRUDN/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:+DiY3wvV6TuJJhpZisF/zLDA0zPMSvHdkr4UvC0qU.  
This key is not known by any other names.  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.  
remote: Enumerating objects: 33, done.  
remote: Counting objects: 100% (33/33), done.  
remote: Compressing objects: 100% (32/32), done.  
remote: Total 33 (delta 1), reused 18 (delta 0), pack-reused 0 (from 0)  
Получение объектов: 100% (33/33), 18.82 Киб | 3.14 Миб/с, готово.  
Определение изменений: 100% (1/1), готово.  
Подмодуль «template/presentation» (https://github.com/yamadharma/academic-presentation-mar
```

Figure 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.cs
l
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/pandocattribu
tes.py
create mode 100644 project-personal/stage6/report/report.md
albertokaskante@albertokaskante:~/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 8 потоков
Сжатие объектов: 100% (30/30), готово.
Запись объектов: 100% (37/37), 342.15 Киб | 2.50 Миб/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:AlbertoRUDN/os-intro.git
ee16062..319c54e master -> master
albertokaskante@albertokaskante:~/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: