

Отчёт по лабораторной работе №2

Управление версиями

Еюбоглу Тимур НПИбд-01-22

Содержание

1	Цель работы	4
2	Выполнение лабораторной работы	5
3	Вывод	11
4	Контрольные вопросы	12
	Список литературы	16

Список иллюстраций

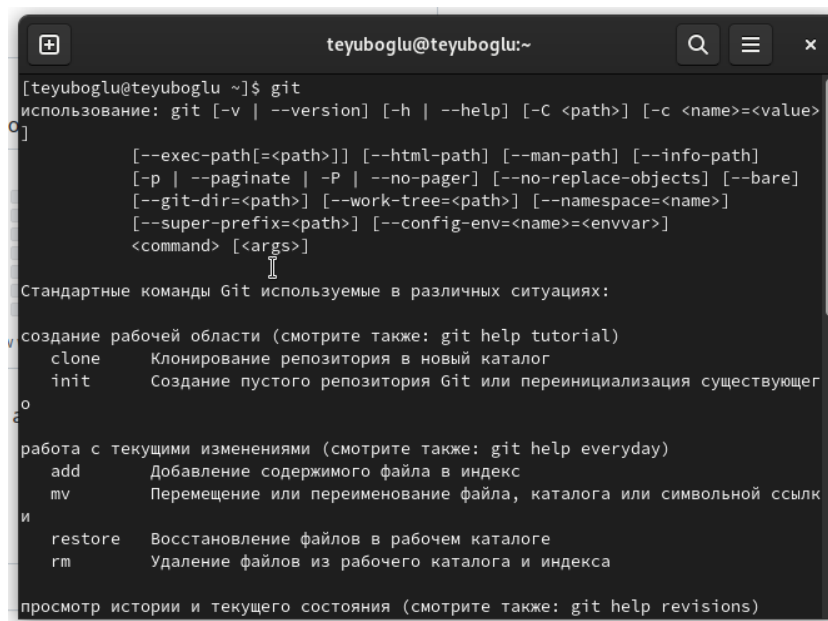
2.1	Загрузка пакетов	5
2.2	Параметры репозитория	6
2.3	rsa-4096	6
2.4	ed25519	7
2.5	GPG ключ	7
2.6	GPG ключ	8
2.7	Параметры репозитория	8
2.8	Связь репозитория с аккаунтом	9
2.9	Загрузка шаблона	9
2.10	Первый коммит	10

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.



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

```
teyuboglu@teyuboglu:~  
tag      Создание, вывод списка, удаление или проверка метки, подписанной с  
помощью GPG  
совместная работа (смотрите также: git help workflows)  
fetch    Загрузка объектов и ссылок из другого репозитория  
pull     Извлечение изменений и объединение с другим репозиториом или локаль  
ной веткой  
push     Обновление внешних ссылок и связанных объектов  
  
«git help -a» и «git help -g» выводит список доступных подкоманд  
и небольшую справку по понятиям. Смотрите «git help <понятие>»  
или «git help <термин>» чтобы узнать больше о конкретной подкоманде  
или понятии.  
Смотрите «git help git» для получения общего обзора системы.  
[teyuboglu@teyuboglu ~]$  
[teyuboglu@teyuboglu ~]$  
[teyuboglu@teyuboglu ~]$  
[teyuboglu@teyuboglu ~]$ git config --global user.name "teyuboglu-rudn"  
[teyuboglu@teyuboglu ~]$ git config --global user.email "1032224357@pfur.ru"  
[teyuboglu@teyuboglu ~]$ git config --global core.quotePath false  
[teyuboglu@teyuboglu ~]$ git config --global init.defaultBranch master  
[teyuboglu@teyuboglu ~]$ git config --global core.autocrlf input  
[teyuboglu@teyuboglu ~]$ git config --global core.safecrlf warn  
[teyuboglu@teyuboglu ~]$
```

Рис. 2.2: Параметры репозитория

Создаем SSH ключи

```
teyuboglu@teyuboglu:~  
[teyuboglu@teyuboglu ~]$ ssh-keygen -t rsa -b 4096  
Generating public/private rsa key pair.  
Enter file in which to save the key (/home/teyuboglu/.ssh/id_rsa):  
/home/teyuboglu/.ssh/id_rsa already exists.  
Overwrite (y/n)? y  
Enter passphrase (empty for no passphrase):  
Enter same passphrase again:  
Your identification has been saved in /home/teyuboglu/.ssh/id_rsa  
Your public key has been saved in /home/teyuboglu/.ssh/id_rsa.pub  
The key fingerprint is:  
SHA256:034fsxpRvuwZnMmc+ax65WEYHEsofpRt2rDjmrrzw00 teyuboglu@teyuboglu  
The key's randomart image is:  
+----[RSA 4096]-----+  
|          +          |  
|      . = =         |  
|    . o O +         |  
|     o = B          |  
|    S + o +         |  
|       o E B O.     |  
|      . * o /o.     |  
|     . = o +.O.     |  
|    o=.. o++.o      |  
+-----[SHA256]-----+  
[teyuboglu@teyuboglu ~]$
```

Рис. 2.3: rsa-4096

```
teyuboglu@teyuboglu:~$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/teyuboglu/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/teyuboglu/.ssh/id_ed25519
Your public key has been saved in /home/teyuboglu/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:HKNE+QaWyHCiVtnebNj0m17R8at01BM5/u0mgSK5+I teyuboglu@teyuboglu
The key's randomart image is:
+---[ED25519 256]---+
| o+..o      o..|
| ..+oo=     +. |
| .. ..*o+   . o |
| .  = @o=   = ..|
|      B.S ..o +..|
|      .....+ oo|
|      . o. ....|
|      .o  . o |
|      .E.. o   |
+---[SHA256]-----+
[teyuboglu@teyuboglu ~]$
```

Рис. 2.4: ed25519

Создаем GPG ключ

```
teyuboglu@teyuboglu:~$ gpg --full-gen-key
Вы выбрали следующий идентификатор пользователя:
"teyuboglu-rudn <1032224357@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/teyuboglu/.gnupg/trustdb.gpg: создана таблица доверия
gpg: создан каталог '/home/teyuboglu/.gnupg/openpgp-revocs.d'
gpg: сертификат отзыва записан в '/home/teyuboglu/.gnupg/openpgp-revocs.d/50BA32
496CD0F6959CE0621A7463C4158668B7E1.rev'.
открытый и секретный ключи созданы и подписаны.

pub  rsa4096 2023-02-17 [SC]
     50BA32496CD0F6959CE0621A7463C4158668B7E1
uid                               teyuboglu-rudn <1032224357@pfur.ru>
sub  rsa4096 2023-02-17 [E]

[teyuboglu@teyuboglu ~]$
```

Рис. 2.5: GPG ключ

Добавляем GPG ключ в аккаунт

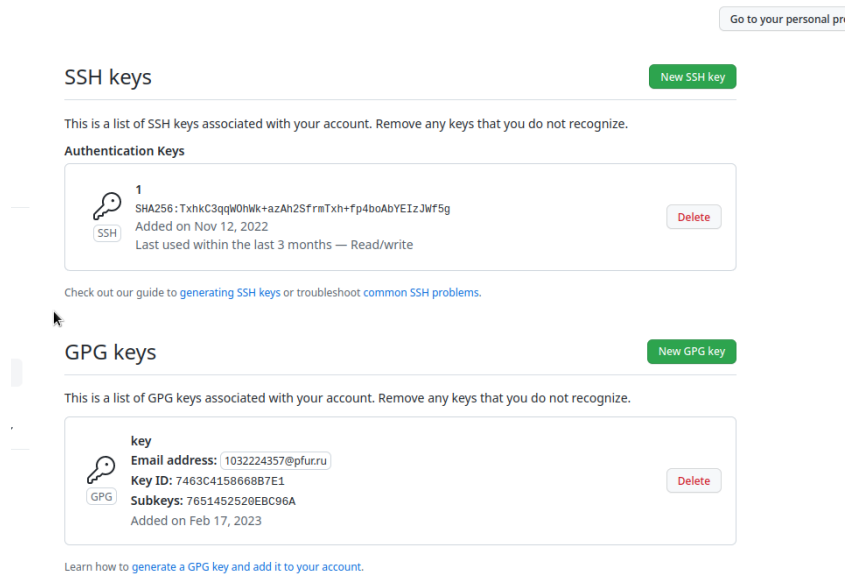


Рис. 2.6: GPG ключ

Настройка автоматических подписей коммитов git

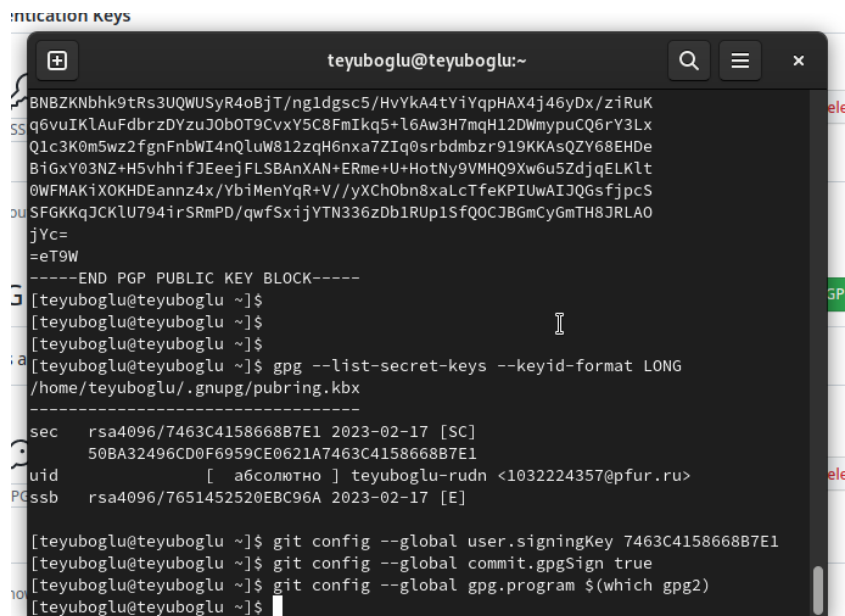


Рис. 2.7: Параметры репозитория

Настройка gh


```
teyuboglu@teyuboglu:~$ git config --global user.signingKey 7463C4158668B7E1
uid [ абсолютно ] teyuboglu-rudn <1032224357@pfur.ru>
ssb rsa4096/7651452520EBC96A 2023-02-17 [E]

[teyuboglu@teyuboglu ~]$ git config --global user.signingKey 7463C4158668B7E1
[teyuboglu@teyuboglu ~]$ git config --global commit.gpgSign true
[teyuboglu@teyuboglu ~]$ git config --global gpg.program $(which gpg2)
[teyuboglu@teyuboglu ~]$
[teyuboglu@teyuboglu ~]$ 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/teyuboglu/.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: 79D6-7917
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/teyuboglu/.ssh/id_rsa.pub
✓ Logged in as teyuboglu-rudn
[teyuboglu@teyuboglu ~]$
```

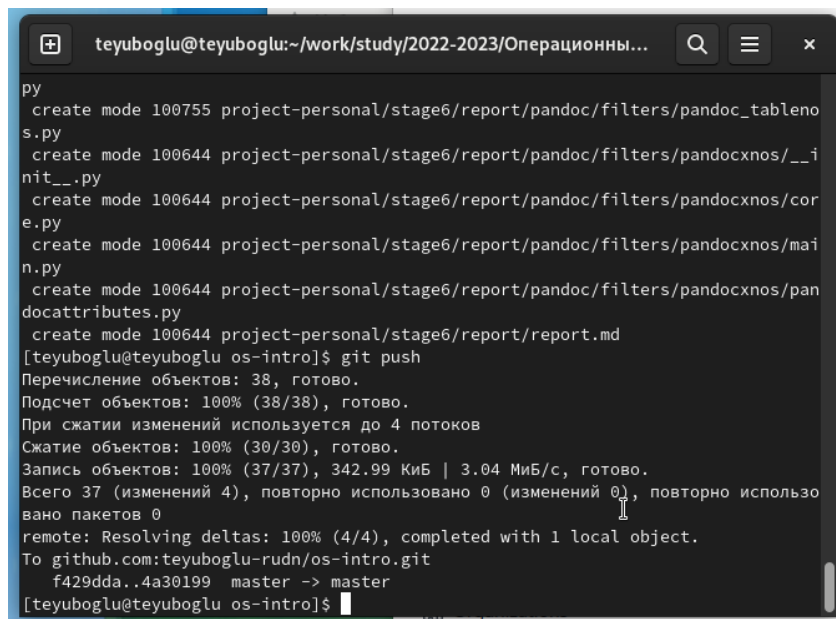
Рис. 2.8: Связь репозитория с аккаунтом

Загрузка шаблона репозитория и синхронизация

```
teyuboglu@teyuboglu:~/work/study/2022-2023/Операционны...$ git clone https://github.com/teyuboglu/os-intro.git
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 | 2.27 МиБ/с, готово.
Определение изменений: 100% (28/28), готово.
Клонирование в «/home/teyuboglu/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 | 3.24 МиБ/с, готово.
Определение изменений: 100% (40/40), готово.
Submodule path 'template/presentation': checked out 'b1be380ee91f5809264cb755d316174540b753e'
Submodule path 'template/report': checked out '1d1b61dcac9c287a83917b82e3aef11a33b1e3b2'
[teyuboglu@teyuboglu Операционные системы]$ cd os-intro/
[teyuboglu@teyuboglu os-intro]$ ls
CHANGELOG.md  COURSE  Makefile  README.en.md  README.md
config        LICENSE  package.json  README.git-flow.md  template
[teyuboglu@teyuboglu os-intro]$
```

Рис. 2.9: Загрузка шаблона

Подготовка репозитория и коммит изменений

A terminal window with a dark background and light text. The title bar shows the user 'teyuboglu' at host 'teyuboglu' in the directory '~/work/study/2022-2023/Операционны...'. The terminal output shows the execution of 'git push' from the 'os-intro' branch. It lists several files being created, including Python scripts and a Markdown report. Progress bars for object counting, compression, and writing are shown, all reaching 100%. The final output shows the commit being pushed to the 'master' branch on GitHub, with a commit hash 'f429dda..4a30199' and a message 'master -> master'.

```
teyuboglu@teyuboglu:~/work/study/2022-2023/Операционны...
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
[teyuboglu@teyuboglu 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), 342.99 КиБ | 3.04 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:teyuboglu-rudn/os-intro.git
f429dda..4a30199 master -> master
[teyuboglu@teyuboglu 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