

# **Отчёт по лабораторной работе №2**

**Управление версиями**

Леонов Алексей НБИбд 01-21

# Содержание

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

# List of Figures

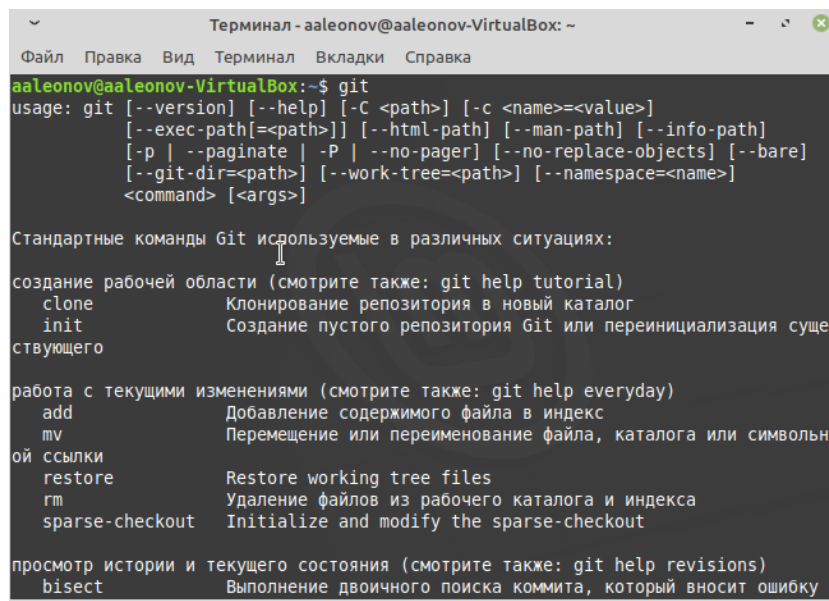
2.1	Загрузка пакетов . . . . .	5
2.2	Параметры репозитория . . . . .	5
2.3	rsa-4096 . . . . .	6
2.4	ed25519 . . . . .	6
2.5	GPG ключ . . . . .	7
2.6	GPG ключ . . . . .	7
2.7	Параметры репозитория . . . . .	8
2.8	Связь репозитория с аккаунтом . . . . .	8
2.9	Загрузка шаблона . . . . .	8
2.10	Первый коммит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



```
Терминал - aaleonov@aaleonov-VirtualBox: ~
Файл  Правка  Вид  Терминал  Вкладки  Справка
aaleonov@aaleonov-VirtualBox:~$ git
usage: git [--version] [--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>]
      <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    Restore working tree files
  rm         Удаление файлов из рабочего каталога и индекса
  sparse-checkout Initialize and modify the sparse-checkout

просмотр истории и текущего состояния (смотрите также: git help revisions)
  bisect     Выполнение двоичного поиска коммита, который вносит ошибку
```

Figure 2.1: Загрузка пакетов

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



```
aaleonov@aaleonov-VirtualBox:~$
aaleonov@aaleonov-VirtualBox:~$ git config --global user.name "1032211396"
aaleonov@aaleonov-VirtualBox:~$ git config --global user.email "1032211396@pfur.ru"
aaleonov@aaleonov-VirtualBox:~$
aaleonov@aaleonov-VirtualBox:~$ git config --global core.quotepath false
aaleonov@aaleonov-VirtualBox:~$ git config --global init.defaultBranch master
aaleonov@aaleonov-VirtualBox:~$ git config --global core.autocrlf input
aaleonov@aaleonov-VirtualBox:~$ git config --global core.safecrlf warn
aaleonov@aaleonov-VirtualBox:~$
```

Figure 2.2: Параметры репозитория

Создаем SSH ключи

```
Терминал - aaleonov@aaleonov-VirtualBox: ~
Файл  Правка  Вид  Терминал  Вкладки  Справка
aaleonov@aaleonov-VirtualBox:~$
aaleonov@aaleonov-VirtualBox:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/aaleonov/.ssh/id_rsa):
Created directory '/home/aaleonov/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/aaleonov/.ssh/id_rsa
Your public key has been saved in /home/aaleonov/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:INkZBate390eIHG8mAkawWHBP+GDtDtayaylsfLCFdk aaleonov@aaleonov-VirtualBox
The key's randomart image is:
+---[RSA 4096]-----+
|.OB*00000|
|. +==+E+.o|
|..+B=+.+.o|
|+.*.+. .|
|o + oS .|
|@|
|*|
|o =|
|o.|
+---[SHA256]-----+
aaleonov@aaleonov-VirtualBox:~$ ssh-keygen -t ed25519
```

Figure 2.3: rsa-4096

```
+---[SHA256]-----+
aaleonov@aaleonov-VirtualBox:~$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/aaleonov/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/aaleonov/.ssh/id_ed25519
Your public key has been saved in /home/aaleonov/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:rd9jh0uHLH/SF2MDD+6qlASZOS/0zqQ6IOYFgA4t5Ws aaleonov@aaleonov-VirtualBox
The key's randomart image is:
+---[ED25519 256]---+
|.O.|
|.. +|
|.. B|
|O . . =. +|
|E .S=. + +|
|+ o B... + *|
|O O . . = +.+=|
|. .. .. o+o..o|
|.. .oo++ o|
+---[SHA256]-----+
aaleonov@aaleonov-VirtualBox:~$
```

Figure 2.4: ed25519

Создаем GPG ключ

```
Терминал - aaleonov@aaleonov-VirtualBox: ~
Файл  Правка  Вид  Терминал  Вкладки  Справка

gpg: ключ 34DE8B4C4682113F помечен как абсолютно доверенный
gpg: создан каталог '/home/aaleonov/.gnupg/openpgp-revocs.d'
gpg: сертификат отзыва записан в '/home/aaleonov/.gnupg/openpgp-revocs.d/F3B7B0D7F1921F4D308E9FDA34DE8B4C4682113F.rev'.
открытый и секретный ключи созданы и подписаны.

pub  rsa4096 2022-09-07 [SC]
    F3B7B0D7F1921F4D308E9FDA34DE8B4C4682113F
uid          aleksey <1032211396@pfur.ru>
sub  rsa4096 2022-09-07 [E]

aaleonov@aaleonov-VirtualBox:~$ 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
/home/aaleonov/.gnupg/pubring.kbx
-----
sec  rsa4096/34DE8B4C4682113F 2022-09-07 [SC]
    F3B7B0D7F1921F4D308E9FDA34DE8B4C4682113F
uid          [ абсолютно ] aleksey <1032211396@pfur.ru>
ssb  rsa4096/9B552FE3092D70C4 2022-09-07 [E]

aaleonov@aaleonov-VirtualBox:~$
```

Figure 2.5: GPG ключ

## Добавляем GPG ключ в аккаунт

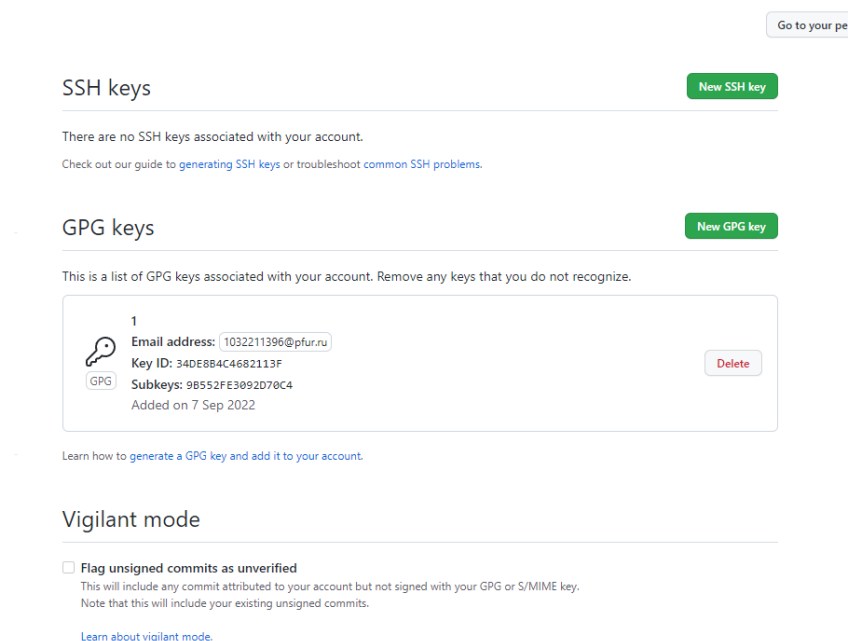


Figure 2.6: GPG ключ

## Настройка автоматических подписей коммитов git

```
aaleonov@aaleonov-VirtualBox:~$ git config --global user.signingkey 34DE8B4C4682113F
aaleonov@aaleonov-VirtualBox:~$ git config --global commit.gpgsign true
aaleonov@aaleonov-VirtualBox:~$ git config --global gpg.program $(which gpg2)
aaleonov@aaleonov-VirtualBox:~$
```

Figure 2.7: Параметры репозитория

## Настройка gh

```
Терминал - aaleonov@aaleonov-VirtualBox: ~
Файл Правка Вид Терминал Вкладки Справка
aaleonov@aaleonov-VirtualBox:~$ git config --global gpg.program $(which gpg2)
aaleonov@aaleonov-VirtualBox:~$ 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/aaleonov/.ssh/id_rsa.pub
? How would you like to authenticate GitHub CLI? Login with a web browser

! First copy your one-time code: E519-926B
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/aaleonov/.ssh/id_rsa.pub
✓ Logged in as 1032211396
```

Figure 2.8: Связь репозитория с аккаунтом

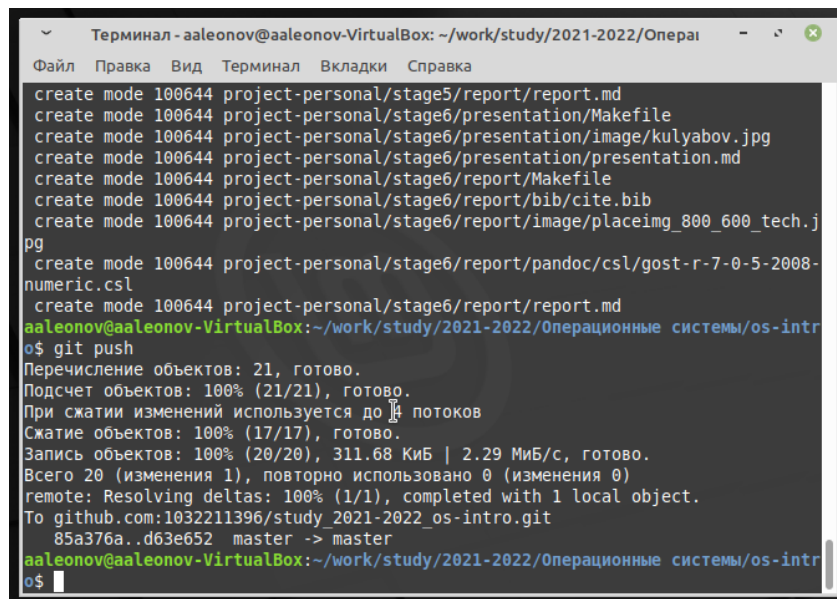
## Загрузка шаблона репозитория и синхронизация

```
Терминал - aaleonov@aaleonov-VirtualBox: ~/work/study/2021-2022/Операционные системы/Операционные системы/os-intro
Файл Правка Вид Терминал Вкладки Справка
Клонирование в «/home/aaleonov/work/study/2021-2022/Операционные системы/os-intro/template/presentation»...
remote: Enumerating objects: 71, done.
remote: Counting objects: 100% (71/71), done.
remote: Compressing objects: 100% (49/49), done.
remote: Total 71 (delta 23), reused 68 (delta 20), pack-reused 0
Клонирование в «/home/aaleonov/work/study/2021-2022/Операционные системы/os-intro/template/report»...
remote: Enumerating objects: 78, done.
remote: Counting objects: 100% (78/78), done.
remote: Compressing objects: 100% (52/52), done.
remote: Total 78 (delta 31), reused 69 (delta 22), pack-reused 0
Подмодуль по пути «template/presentation»: забрано состояние «2703b47423792d472694aaf7555a5626dce51a25»
Подмодуль по пути «template/report»: забрано состояние «df7b2ef80f8def3b9a496f8695277469a1a7842a»
aaleonov@aaleonov-VirtualBox:~/work/study/2021-2022/Операционные системы/os-intro$ cd ~/work/study/2021-2022/Операционные системы/os-intro
aaleonov@aaleonov-VirtualBox:~/work/study/2021-2022/Операционные системы/os-intro$ rm package.json
aaleonov@aaleonov-VirtualBox:~/work/study/2021-2022/Операционные системы/os-intro$ make COURSE=os-intro
aaleonov@aaleonov-VirtualBox:~/work/study/2021-2022/Операционные системы/os-intro$
```

Figure 2.9: Загрузка шаблона



## Подготовка репозитория и коммит изменений



```
Терминал - aaleonov@aaleonov-VirtualBox: ~/work/study/2021-2022/Операционные системы/os-intro
Файл  Правка  Вид  Терминал  Вкладки  Справка

create mode 100644 project-personal/stage5/report/report.md
create mode 100644 project-personal/stage6/presentation/Makefile
create mode 100644 project-personal/stage6/presentation/image/kulyabov.jpg
create mode 100644 project-personal/stage6/presentation/presentation.md
create mode 100644 project-personal/stage6/report/Makefile
create mode 100644 project-personal/stage6/report/bib/cite.bib
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 100644 project-personal/stage6/report/report.md
aaleonov@aaleonov-VirtualBox:~/work/study/2021-2022/Операционные системы/os-intro
o$ git push
Перечисление объектов: 21, готово.
Подсчет объектов: 100% (21/21), готово.
При сжатии изменений используется до 1 потоков
Сжатие объектов: 100% (17/17), готово.
Запись объектов: 100% (20/20), 311.68 КиБ | 2.29 МиБ/с, готово.
Всего 20 (изменения 1), повторно использовано 0 (изменения 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:1032211396/study_2021-2022_os-intro.git
 85a376a..d63e652  master -> master
aaleonov@aaleonov-VirtualBox:~/work/study/2021-2022/Операционные системы/os-intro
o$
```

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:

# Список литературы

1. Лекция Системы контроля версий
2. GitHub для начинающих