

Отчёт по лабораторной работе №2

Управление версиями

Сальников Даниил Александрович НБИбд-02-21

Содержание

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

List of Figures

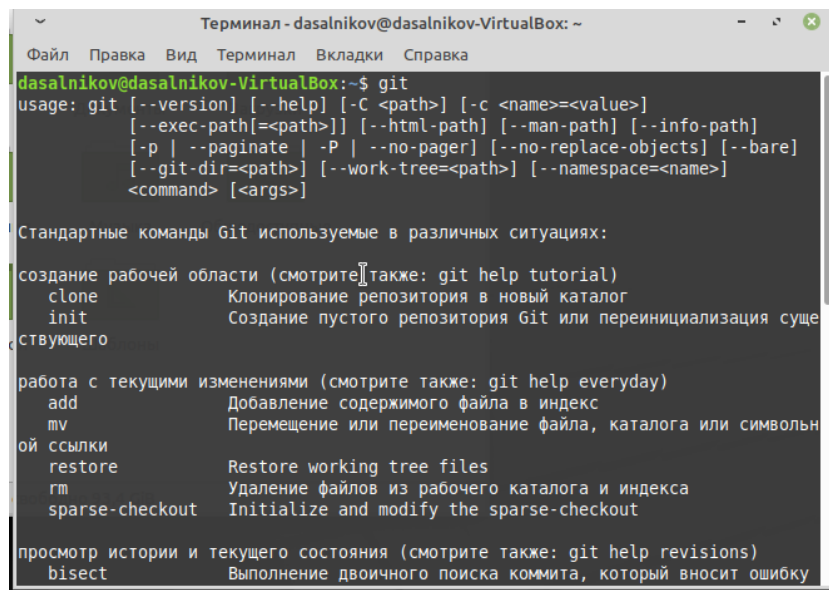
2.1	Загрузка пакетов	5
2.2	Параметры репозитория	6
2.3	rsa-4096	6
2.4	ed25519	7
2.5	GPG ключ	7
2.6	GPG ключ	8
2.7	Параметры репозитория	8
2.8	Связь репозитория с аккаунтом	8
2.9	Загрузка шаблона	9
2.10	Первый коммит	9

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 - dasalnikov@dasalnikov-VirtualBox: ~
Файл  Правка  Вид  Терминал  Вкладки  Справка
dasalnikov@dasalnikov-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.

```
Терминал - dasalnikov@dasalnikov-VirtualBox: ~
Файл  Правка  Вид  Терминал  Вкладки  Справка
tag      Создание, вывод списка, удаление или проверка метки, подпис
анной 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 репозиторием ил
и локальной веткой
  push   Обновление внешних ссылок и связанных объектов

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
See 'git help git' for an overview of the system.
dasalnikov@dasalnikov-VirtualBox:~$
dasalnikov@dasalnikov-VirtualBox:~$
dasalnikov@dasalnikov-VirtualBox:~$ git config --global user.name "dasalnikov"
dasalnikov@dasalnikov-VirtualBox:~$ git config --global user.email "1032209524@p
fur.ru"
dasalnikov@dasalnikov-VirtualBox:~$ git config --global core.quotepath false
dasalnikov@dasalnikov-VirtualBox:~$ git config --global init.defaultBranch maste
r
dasalnikov@dasalnikov-VirtualBox:~$ git config --global core.autocrlf input
dasalnikov@dasalnikov-VirtualBox:~$ git config --global core.safecrlf warn
dasalnikov@dasalnikov-VirtualBox:~$
```

Figure 2.2: Параметры репозитория

Создаем SSH ключи

```
Терминал - dasalnikov@dasalnikov-VirtualBox: ~
Файл  Правка  Вид  Терминал  Вкладки  Справка
dasalnikov@dasalnikov-VirtualBox:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/dasalnikov/.ssh/id_rsa):
Created directory '/home/dasalnikov/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/dasalnikov/.ssh/id_rsa
Your public key has been saved in /home/dasalnikov/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:6ollBfwmBezg0Vuf4rguRUGtxIK0TcZXXhUb04LYVVw dasalnikov@dasalnikov-Virtual
Box
The key's randomart image is:
+----[RSA 4096]-----+
|  . . +00+0=+..E |
|  = 0.=00 =. |
| * . +0*+ * |
| . 0 * 0+ . |
|  . +S.+ |
|  0..+0 |
|  .00.. |
|  .+.0. |
|  .0=. |
+----[SHA256]-----+
dasalnikov@dasalnikov-VirtualBox:~$
```

Figure 2.3: rsa-4096

```
Терминал - dasalnikov@dasalnikov-VirtualBox: ~
Файл Правка Вид Терминал Вкладки Справка
+----[SHA256]-----+
dasalnikov@dasalnikov-VirtualBox:~$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/dasalnikov/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/dasalnikov/.ssh/id_ed25519
Your public key has been saved in /home/dasalnikov/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:CzMnDv4NVgfdgEaBx8VI+w1TsmIek0d2246AemCAzs dasalnikov@dasalnikov-Virtual
Box
The key's randomart image is:
+---[ED25519 256]---+
|      o+*=*=..      |
|      .B++==o       |
|      . .O+*.OO.    |
|      O...+OO .     |
|      E B.S o . o   |
|      . + O + +     |
|      . + . O       |
|      o o           |
|      .             |
+---[SHA256]-----+
dasalnikov@dasalnikov-VirtualBox:~$
```

Figure 2.4: ed25519

Создаем GPG ключ

```
Терминал - dasalnikov@dasalnikov-VirtualBox: ~
Файл Правка Вид Терминал Вкладки Справка
0 = не ограничен
<p> = срок действия ключа - п дней
<p>w = срок действия ключа - п недель
<p>m = срок действия ключа - п месяцев
<p>y = срок действия ключа - п лет
Срок действия ключа? (0)
Срок действия ключа не ограничен
Все верно? (у/Н) у
GnuPG должен составить идентификатор пользователя для идентификации ключа.
Ваше полное имя: dasalnikov
Адрес электронной почты: 1032209524@pfur.ru
Примечание:
Вы выбрали следующий идентификатор пользователя:
"dasalnikov <1032209524@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/dasalnikov/.gnupg/trustdb.gpg: создана таблица доверия
gpg: ключ CDE9BE417C7D967E помечен как абсолютно доверенный
gpg: создан каталог '/home/dasalnikov/.gnupg/openpgp-revocs.d'
gpg: сертификат отзыва записан в '/home/dasalnikov/.gnupg/openpgp-revocs.d/DA8448653651638CD4E88C45CDE9BE417C7D967E.rev',
открытый и секретный ключи созданы и подписаны.

pub   rsa4096 2022-04-30 [SC]
      DA8448653651638CD4E88C45CDE9BE417C7D967E
uid           dasalnikov <1032209524@pfur.ru>
sub    rsa4096 2022-04-30 [E]
dasalnikov@dasalnikov-VirtualBox:~$
```

Figure 2.5: GPG ключ

Добавляем GPG ключ в аккаунт

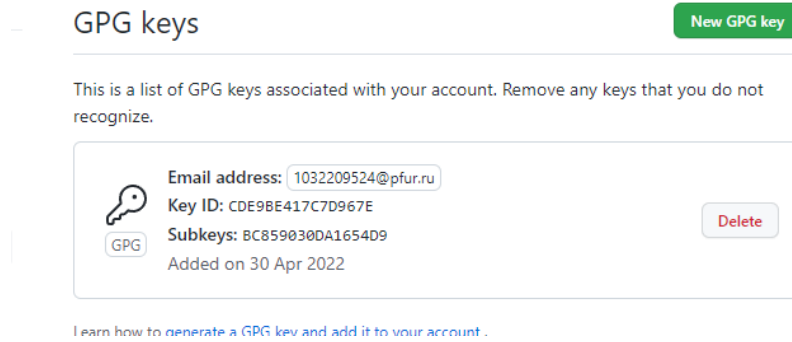


Figure 2.6: GPG ключ

Настройка автоматических подписей коммитов git

```
-----END PGP PUBLIC KEY BLOCK-----
dasalnikov@dasalnikov-VirtualBox:~$
dasalnikov@dasalnikov-VirtualBox:~$
dasalnikov@dasalnikov-VirtualBox:~$
dasalnikov@dasalnikov-VirtualBox:~$ git config --global user.signingkey CDE9BE417C7D967E
dasalnikov@dasalnikov-VirtualBox:~$ git config --global commit.gpgsign true
dasalnikov@dasalnikov-VirtualBox:~$ git config --global gpg.program $(which gpg2)
```

Figure 2.7: Параметры репозитория

Настройка gh

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

First copy your one-time code: 6DC6-BF61
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/dasalnikov/.ssh/id_rsa.pub
Logged in as dasalnikov

A new release of gh is available: 2.8.0 → v2.9.0
https://github.com/cli/cli/releases/tag/v2.9.0
dasalnikov@dasalnikov-VirtualBox:~$
```

Figure 2.8: Связь репозитория с аккаунтом

Загрузка шаблона репозитория и синхронизация


```

dasalnikov@dasalnikov-VirtualBox:~$ mkdir -p ~/work/study/2021-2022/"Операционные системы"
dasalnikov@dasalnikov-VirtualBox:~$ cd ~/work/study/2021-2022/"Операционные системы"
dasalnikov@dasalnikov-VirtualBox:~/work/study/2021-2022/Операционные системы$ gh repo create study_2021-2022_os-intro --t
template=yamadharma/course-directory-student-template --public
Created repository dasalnikov/study_2021-2022_os-intro on GitHub
dasalnikov@dasalnikov-VirtualBox:~/work/study/2021-2022/Операционные системы$ git clone --recursive git@github.com:dasalnikov/study_2021-2022_os-intro.git os-intro
Клонирование в «os-intro»...
The authenticity of host 'github.com [140.82.121.4]' can't be established.
ECDSA key fingerprint is SHA256:p2QAMXNIC1TJYweI0ttrVc98/R1BUFWu3/LiyKgUfQM.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com,140.82.121.4' (ECDSA) to the list of known hosts.
remote: Enumerating objects: 20, done.
remote: Counting objects: 100% (20/20), done.
remote: Compressing objects: 100% (18/18), done.
remote: Total 20 (delta 2), reused 15 (delta 2), pack-reused 0
Получение объектов: 100% (20/20), 12.49 Киб | 4.16 Миб/с, готово.
Определение изменений: 100% (2/2), готово.
Подмодуль «template/presentation» (https://github.com/yamadharma/academic-presentation-markdown-template.git) зарегистрир
ован по пути «template/presentation»
Подмодуль «template/report» (https://github.com/yamadharma/academic-laboratory-report-template.git) зарегистрирован по пу
ти «template/report»
Клонирование в «~/home/dasalnikov/work/study/2021-2022/Операционные системы/os-intro/template/presentation»...
remote: Enumerating objects: 42, done.
remote: Counting objects: 100% (42/42), done.
remote: Compressing objects: 100% (34/34), done.
remote: Total 42 (delta 9), reused 40 (delta 7), pack-reused 0
Клонирование в «~/home/dasalnikov/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 «3eae6b7586f8a9aded2b506cd1018e625b228b93»
Подмодуль по пути «template/report»: забрано состояние «df7b2ef80f8def3b9a496f8695277469a1a7842a»
dasalnikov@dasalnikov-VirtualBox:~/work/study/2021-2022/Операционные системы$

```

Figure 2.9: Загрузка шаблона

Подготовка репозитория и коммит изменений

```

dasalnikov@dasalnikov-VirtualBox:~/work/study/2021-2022/Операционные системы/os-intro$ rm package.json
dasalnikov@dasalnikov-VirtualBox:~/work/study/2021-2022/Операционные системы/os-intro$ make COURSE=os-intro
dasalnikov@dasalnikov-VirtualBox:~/work/study/2021-2022/Операционные системы/os-intro$ git add .
dasalnikov@dasalnikov-VirtualBox:~/work/study/2021-2022/Операционные системы/os-intro$ git commit -am 'feat(main): make c
ourse structure'
[master 373b9ec] feat(main): make course structure
16 files changed, 1580 insertions(+), 14 deletions(-)
create mode 100644 labs/lab{01..15}/presentation/Makefile
create mode 100644 labs/lab{01..15}/presentation/presentation.md
create mode 100644 labs/lab{01..15}/report/Makefile
create mode 100644 labs/lab{01..15}/report/bib/cite.bib
create mode 100644 labs/lab{01..15}/report/image/placeimg_800_600_tech.jpg
create mode 100644 labs/lab{01..15}/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100644 labs/lab{01..15}/report/report.md
delete mode 100644 package.json
create mode 100644 project-personal/stage{1..6}/presentation/Makefile
create mode 100644 project-personal/stage{1..6}/presentation/presentation.md
create mode 100644 project-personal/stage{1..6}/report/Makefile
create mode 100644 project-personal/stage{1..6}/report/bib/cite.bib
create mode 100644 project-personal/stage{1..6}/report/image/placeimg_800_600_tech.jpg
create mode 100644 project-personal/stage{1..6}/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100644 project-personal/stage{1..6}/report/report.md
create mode 100644 structure
dasalnikov@dasalnikov-VirtualBox:~/work/study/2021-2022/Операционные системы/os-intro$ git push
Warning: Permanently added the ECDSA host key for IP address '140.82.121.3' to the list of known hosts.
Перечисление объектов: 20, готово.
Подсчет объектов: 100% (20/20), готово.
При сжатии изменений используется до 4 потоков
Сжатие объектов: 100% (14/14), готово.
Запись объектов: 100% (19/19), 266.46 Киб | 2.54 Миб/с, готово.
Всего 19 (изменения 2), повторно использовано 0 (изменения 0)
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To github.com:dasalnikov/study_2021-2022_os-intro.git
9c23820..373b9ec master -> master
dasalnikov@dasalnikov-VirtualBox:~/work/study/2021-2022/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:

Список литературы

1. Лекция Системы контроля версий
2. GitHub для начинающих