Лабораторная работа №2

Научное программирование

Дарижапов Тимур Андреевич

Содержание

List of Tables

List of Figures

1 Цель работы

Научиться оформлять отчёты с помощью легковесного языка разметки Markdown.

2 Задание

- Сделайте отчёт по предыдущей лабораторной работе в формате Markdown.
- В качестве ответа просьба предоставить отчёты в 3 форматах: pdf, docx и md.

3 Выполнение лабораторной работы

Внизу представлен текст лабораторной работы №1.

Создаём учётную запись на https://github.com.

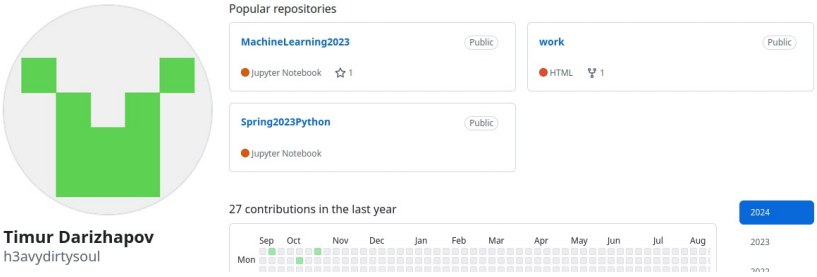

Figure 3.1: Учетная запись

Настроем систему контроля версий git, как это описано в лабораторной работе с использованием сервера репозиториев https://github.com/ начала сделаем предварительную конфигурацию,указав имя и email владельца репозитория: git config –global user.name“Имя Фамилия”git config –global user.email“work@mail”. Настроим верификацию и подписание коммитов git. Зададим имя начальной ветки. Для последующей идентификации пользователя на сервере репозиториев необходимо сгенерировать пару ключей (приватный и открытый):ssh-keygen -C“Имя Фамилия work@mail”Ключи сохраняться в каталоге~/.ssh/. Генерируем ключ двумя способами.

```
timur@tadarizhapov:~/InfoBez$ git config --global user.name 'Timur Darizhapov'
timur@tadarizhapov:~/InfoBez$ git config --global user.email 'timaanddar@mail.ru'
timur@tadarizhapov:~/InfoBez$ git config --global core.quotepath false
timur@tadarizhapov:~/InfoBez$ git config --global init.defaultbranch master
timur@tadarizhapov:~/InfoBez$ git config --global core.autocrlf input
timur@tadarizhapov:~/InfoBez$ git config --global core.safecrlf warn
timur@tadarizhapov:~/InfoBez$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.

Enter file in which to save the key (/home/timur/.ssh/id_rsa): /home/timur/.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/timur/.ssh/id_rsa
Your public key has been saved in /home/timur/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:Dh3n3FkuJk1ZdPi5eVm+8/bhDzrClVQLQtC4kGPoea0 timur@tadarizhapov
The key's randomart image is:
+---[RSA 4096]----+
|     . ..=.  .o..|
|    . = . o .oo. |
|   . o +....oo.o.|
|    o ..o= +.+..o|
|     ...S +.*...=|
|      Eo   oo. +o|
|        .. .  ..o|
|          o ...+o|
|           ... oO|
+----[SHA256]-----+
```

Figure 3.2: Идентификация и создание ключа

```
timur@tadarizhapov:~/InfoBez$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/timur/.ssh/id_ed25519):
/home/timur/.ssh/id_ed25519 already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/timur/.ssh/id_ed25519
Your public key has been saved in /home/timur/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:oXYwq6pLVq7F4sQy7zgKtMSTj+raiPl4ZqxG1XNl9Wo timur@tadarizhapov
The key's randomart image is:
+--[ED25519 256]--+
|           ..    |
|          o  .   |
|     . o +    .  |
|. .. o * .  .    |
| *..  * S  E     |
|+.O  o .  .      |
|=X =.            |
|@XO.             |
|^#*              |
+----[SHA256]-----+
```

Figure 3.3: Создание ключа

Генерируем ключ pqp командой gpg –full-generate-key и следуя инструкциям из лабораторной работы создаём ключ.

```
timur@tadarizhapov:~/InfoBez$ gpg --full-generate-key
gpg (GnuPG) 2.4.4; 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.

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
```

Figure 3.4: Создание ключа pgp

```
GnuPG должен составить идентификатор пользователя для идентификации ключа.

Ваше полное имя: tadarizhapov
Адрес электронной почты: timaanddar@mail.ru
Примечание:
Вы выбрали следующий идентификатор пользователя:
    "tadarizhapov <timaanddar@mail.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: сертификат отзыва записан в '/home/timur/.gnupg/openpgp-revocs.d/3D143658FB71129C1C77EEE4F3917C7F999515CA.rev'.
открытый и секретный ключи созданы и подписаны.

pub   rsa4096 2024-09-14 [SC]
      3D143658FB71129C1C77EEE4F3917C7F999515CA
uid                      tadarizhapov <timaanddar@mail.ru>
sub   rsa4096 2024-09-14 [E]
```

Figure 3.5: Настройка ключа

Выводим список ключей и копируем отпечаток приватного ключа.

```
timur@tadarizhapov:~/InfoBez$ 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:   2  подписанных:   0  доверие: 0-, 0q, 0n, 0m, 0f, 2u
/home/timur/.gnupg/pubring.kbx
------------------------------
sec   rsa1024/2B339CA12873B53B 2024-09-14 [SC]
      63E035958755E9DEE474A9112B339CA12873B53B
uid                 [  абсолютно ] Timur <tidaan@mail.ru>
ssb   rsa1024/40D9E73D131421C4 2024-09-14 [E]

sec   rsa4096/F3917C7F999515CA 2024-09-14 [SC]
      3D143658FB71129C1C77EEE4F3917C7F999515CA
uid                 [  абсолютно ] tadarizhapov <timaanddar@mail.ru>
ssb   rsa4096/D88B56BBEB627544 2024-09-14 [E]

timur@tadarizhapov:~/InfoBez$
```

Figure 3.6: Список и отпечаток

Скопируем наш сгенерированный PGP ключ в буфер обмена.

Figure 3.7: Демонстрация ключа

Перейдем в настройки GitHub (https://github.com/settings/keys), нажмем на кнопку New GPG key и вставим полученный ключ в поле ввода.

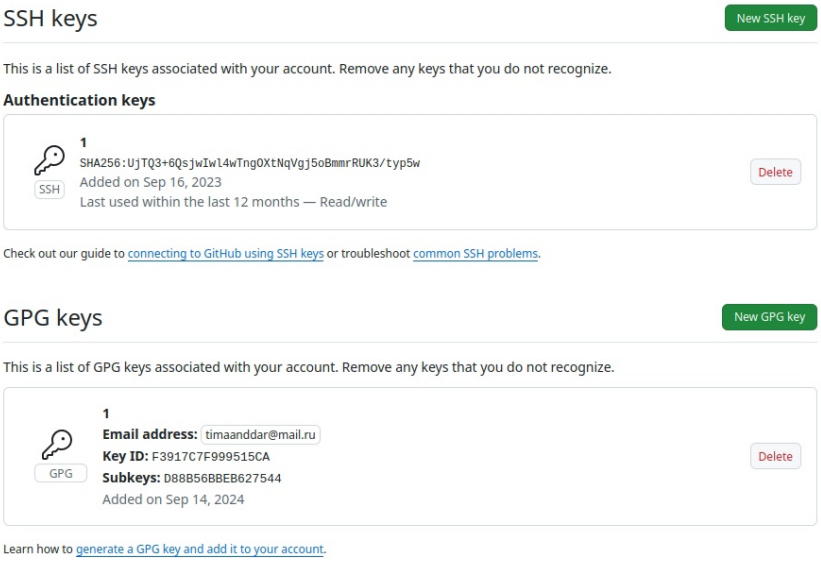

Figure 3.8: Отображение ключа

Осталось настроить автоматические подписи с помощью команд:

```
timur@tadarizhapov:~/InfoBez$ git config --global user.signingkey F3917C7F999515CA
timur@tadarizhapov:~/InfoBez$ git config --global commit.gpgsign true
timur@tadarizhapov:~/InfoBez$ git config --global gpg.program $[which gpg2]
bash: which gpg2: синтаксическая ошибка в выражении (неверный маркер «gpg2»)
timur@tadarizhapov:~/InfoBez$ git config --global gpg.program $(which gpg2)
timur@tadarizhapov:~/InfoBez$ mkdir -p ~/work/2024-2025/'Научное программирование'
timur@tadarizhapov:~/InfoBez$ cd ~/work/2024-2025/'Научное программирование'
timur@tadarizhapov:~/work/2024-2025/Научное программирование$
```

Figure 3.9: Отображение ключа

Далее мы создаём репозиторий на основе шаблона и отправляем все имеющиеся данные на GitHub.

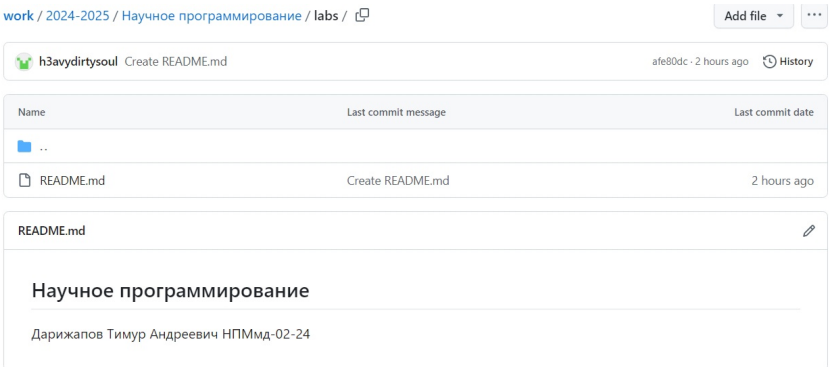

Figure 3.10: Отображение ключа

4 Выводы

В ходе выполнения лабораторной работы я научился оформлять отчёты с помощью легковесного языка разметки Markdown.

5 Ответы на контрольные вопросы

1) Системы контроля версий (Version Control System,VCS) применяются при работе нескольких человек над одним проектом.

2)В классических системах контроля версий используется централизованная модель,предполагающая наличие единого репозитория для хранения файлов.Выполнение боль-шинства функций по управлению версиями осуществляется специальным сервером.Участник проекта (пользователь) перед началом работы посредством определённыхкоманд получает нужную ему версию файлов.После внесения изменений,пользовательразмещает новую версию в хранилище. При этом предыдущие версии не удаляютсяиз центрального хранилища и к ним можно вернуться в любой момент. Сервер можетсохранятьнеполнуюверсиюизменённыхфайлов,апроизводитьтакназываемуюдельта-компрессию—сохранятьтолько изменения между последовательными версиями,чтопозволяет уменьшить объём хранимых данных. Системы контроля версийтакже могутобеспечиватьдополнительные,более гибкиефункциональные возможности.Например,они могут поддерживать работу с несколь-ми версиями одного файла,сохраняя общую историю изменений до точки ветвленияверсий и собственные истории изменений каждой ветви.Крометого,обычно доступнаинформация отом,кто из участников,когда и какие изменения вносил.Обычнотакогорода информация хранится в журнале изменений,доступ к которому можно ограничить.В отличие от классических,в распределённых системах контроля версий центральныйрепозиторий не является обязательным.

3)Среди классических VCS наиболее известны CVS, Subversion, а среди

распределён-ных — Git, Bazaar, Mercurial. Принципы их работы схожи, отличаются они в основномсинтаксисом используемых в работе команд.

4)Сначала сделаем предварительную конфигурацию,указав имя и email владельца репозитория:git config –global user.name“Имя Фамилия”git config –global user.email“work@mail”и настроив utf-8 в выводе сообщенийgit:git config –global quotepathfalseДля инициализации локального репозитория,расположенного,например,в каталоге~/tutorial,необходимо ввести в командной строке:cdmkdir tutorialcd tutorialgit initПосле это в каталог etutorial появится каталог.git, в котором будет хранитьсяистория изменений.Создадим тестовый текстовый файлhello.txt и добавим его в локальный репозиторий. echo’hello world’> hello.txtgit add hello.txtgit commit -am’Новый файл’Воспользуемся командойstatusдля просмотра изменений в рабочем каталоге,сде-ланных с момента последней ревизии:git status. 5) Для последующей идентификации пользователя на сервере репозиториев необходимосгенерировать пару ключей (приватный и открытый):ssh-keygen -C“Имя Фамилия work@mail”Ключи сохраняться в каталоге~/.ssh/.Существуетнесколькодоступных серверов репозиториев с возможностью бесплатного размещения данных.Например,https://github.com/.Для работы с ним необходимо сначала завести на сайтеhttps://github.com/учёт-ную запись.Затем необходимо загрузить сгенерённый нами ранее открытый ключ.Для этого зайти на сайтhttps://github.com/под своей учётной записью и перейти в ме-нюGitHub setting.После этого выбрать в боковом менюGitHub settingSSH-ключии нажатькнопкуДобавить клюTimes New Romanч.Скопировав из локальной консоли ключ в буфер обменаcat ~/.ssh/id_rsa.pub | xclip -sel clipвставляем ключ в появившееся на сайте поле.После этого можно создать на сайте репозиторий, выбрав в меню Репозитории Создать репозиторий,дать ему название и сделать общедоступным (публичным).Для загрузки репозитория из локального каталога на сервер выполняем следующиекоманды:git remote add originssh://git@github.com//.gitgit push -u origin masterДалееналокальномкомпьютереможновыполнятьстандартныепроцедурыдляработысgitпри наличии

центрального репозитория.

6)У Git две основных задачи: первая — хранить информацию о всех изменениях в коде, а вторая — обеспечение удобства командной работы над кодом.

7)Наиболее часто используемые командыgit:–создание основного дерева репозитория:git init–получение обновлений (изменений)текущего дерева из центрального репозитория:git pull–отправка всех произведённых изменений локального дерева в центральный репози-торий:git push–просмотр списка изменённых файлов втекущей директории:git status–просмотртекущих изменения:git diff–сохранениетекущих изменений:–добавить все изменённые и/или созданные файлы и/или каталоги:git add .–добавить конкретные изменённые и/или созданные файлы и/или каталоги:git add имена_файлов–удалитьфайли/иликаталогизиндексарепозитория(приэтомфайли/или каталог остаётся в локальной директории):git rm имена_файлов–сохранение добавленных изменений:–сохранить все добавленные изменения и все изменённые файлы:git commit -am’Описание коммита’–сохранитьдобавленные изменения с внесением комментария через встроенныйредактор:git commit–создание новой ветки,базирующейся натекущей:git checkout -b имя_ветки–переключение на некоторую ветку:git checkout имя_ветки(при переключении на ветку,которой ещё нетв локальном репозитории,она будетсоздана и связана с удалённой)–отправка изменений конкретной ветки в центральный репозиторий:git push origin имя_ветки–слияние ветки стекущим деревом:git merge –no-ff имя_ветки–удаление ветки:–удаление локальной уже слитой с основным деревом ветки:git branch -d имя_ветки–принудительное удаление локальной ветки:git branch -D имя_ветки–удаление ветки с центрального репозитория:git push origin :имя_ветки

8) Гид для создания текстового файла echo’hello world’> hello.txt git add hello.txt

9)Для фиксации истории проекта в рамках этого процесса вместо одной ветки master используются две ветки.В ветке master хранится официальная история релиза,а ветка develop предназначена для объединения всех функций. Ветви

решают следующие проблемы нужно постоянно создавать архивы с рабочим кодом сложно “переключаться” между архивами сложно перетаскивать изменения между архивами легко запутаться в файлах

10) Во время работы над проектом так или иначе могутсоздаваться файлы,которые нетребуется добавлять в последствии в репозиторий. Например, временные файлы, со-здаваемые редакторами,или объектные файлы,создаваемые компиляторами.Можно прописать шаблоны игнорируемых при добавлении в репозиторий типов файлов в файл.gitignore с помощью сервисов. Для этого сначаланужно получить список имеющихся шаблонов:curl -L -s https://www.gitignore.io/api/listЗатем скачать шаблон,например,для C и C++curl -L -s https://www.gitignore.io/api/c » .gitignorecurl -L -s https://www.gitignore.io/api/c++ » .gitignore.

6 Список литературы

Лабораторная работа №1

Лабораторная работа № 2. Управление версиями [Электронный ресурс]. 2019. URL:https://esystem.rudn.ru/mod/resource/view.php?id=1154997