

Отчёт по лабораторной работе №2

Управление версиями

Хусаинова Динара Айратовна

Содержание

1	Цель работы	5
2	Ход работы	6
3	Контрольные вопросы	13
4	Вывод	16

Список иллюстраций

2.1	Учетная запись	6
2.2	gh	7
2.3	Базовая настройка	7
2.4	utf-8	7
2.5	Генерация ключа	8
2.6	Выбираем опции	9
2.7	Список ключей	9
2.8	Приватный ключ	9
2.9	Сайт	9
2.10	Открываем	10
2.11	Вставляем полученный ключ	10
2.12	Коммиты git	10
2.13	Отвечаем на необходимые вопросы	11
2.14	Пишем код из терминала	11
2.15	Продельываем выше описанные действия	11
2.16	Отправляем файлы	12

List of Tables

1 Цель работы

Изучить идеологию и применение средств контроля версий и освоить умения по работе с git.

2 Ход работы

1.Создаем учетную запись на GitHub. Устанавливаем git-flow в Fedora Linux (рис. 2.1).

```
[dakhusainova@fedora tmp]$ wget --no-check-certificate -q https://raw.githubusercontent.com/petervanderdoes/gitflow/develop/contrib/gitflow-installer.sh
[dakhusainova@fedora tmp]$ low-installer.sh
[dakhusainova@fedora tmp]$ wget --no-check-certificate -q https://raw.githubusercontent.com/petervanderdoes/gitflow/develop/contrib/gitflow-installer.sh
[dakhusainova@fedora tmp]$ chmod +x gitflow-installer.sh
chmod: невозможно получить доступ к 'gitflow-installer.sh': Нет такого файла или каталога
[dakhusainova@fedora tmp]$ chmod +x gitflow-installer.sh
chmod: невозможно получить доступ к 'gitflow-installer.sh': Нет такого файла или каталога
[dakhusainova@fedora tmp]$ chmod +x gitflow-installer.sh
chmod: невозможно получить доступ к 'gitflow-installer.sh': Нет такого файла или каталога
[dakhusainova@fedora tmp]$ wget --no-check-certificate -q https://raw.githubusercontent.com/petervanderdoes/gitflow/develop/contrib/gitflow-installer.sh
[dakhusainova@fedora tmp]$ chmod +x gitflow-installer.sh
[dakhusainova@fedora tmp]$ sudo ./gitflow-installer.sh install stable

Мы полагаем, что ваш системный администратор изложил вам основы безопасности. Как правило, всё сводится к трём следующим правилам:

    №1) Уважайте частную жизнь других.
    №2) Думайте, прежде что-то вводить.
    №3) С большой властью приходит большая ответственность.

[sudo] пароль для dakhusainova: █
```

Рис. 2.1: Учетная запись

Устанавливаем gh в Fedora Linux с помощью команды sudo(рис. 2.2).

```
[dakhusainova@fedora tmp]$ sudo dnf install gh
Последняя проверка окончания срока действия метаданных: 0:02:42 назад, Чт 21 апр 2022 21:45:37.
Зависимости разрешены.
=====
Пакет      Архитектура  Версия      Репозиторий  Размер
=====
Установка:
gh         x86_64       2.7.0-1.fc35 updates      6.8 М
=====
Результат транзакции
=====
Установка 1 Пакет

Объем загрузки: 6.8 М
Объем изменений: 32 М
Продолжить? [д/н]:
```

Рис. 2.2: gh

Далее следует базовая настройка git. Используем имя, которое мы вводили при регистрации на GitHub и ту же электронную почту(рис. 2.3).

```
Общий размер                    573 kB/s | 6.8 MB    00:12
Проверка транзакции
Проверка транзакции успешно завершена.
Идет проверка транзакции
Тест транзакции проведен успешно.
Выполнение транзакции
Подготовка      : 1/1
Установка       : gh-2.7.0-1.fc35.x86_64 1/1
Запуск скрипта  : gh-2.7.0-1.fc35.x86_64 1/1
Проверка        : gh-2.7.0-1.fc35.x86_64 1/1

Установлен:
gh-2.7.0-1.fc35.x86_64

Выполнено!
[dakhusainova@fedora tmp]$ git config --global user.email "volkov_lenya2000@mail.ru"
[dakhusainova@fedora tmp]$ git config --global core.quotepath false
```

Рис. 2.3: Базовая настройка

Настраиваем utf-8 в выводе сообщений git (рис. 2.4).

```
[dakhusainova@fedora tmp]$ git config --global init.defaultBranch master
[dakhusainova@fedora tmp]$ git config --global core.autocrlf input
[dakhusainova@fedora tmp]$ git config --global core.safecrlf warn
[dakhusainova@fedora tmp]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/dakhusainova/.ssh/id_rsa): rsa
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
```

Рис. 2.4: utf-8

При генерации ключа выбираем необходимые опции (рис. 2.5,2.6).

- тип RSA and RSA;
- размер 4096;

- выберите срок действия; значение по умолчанию— 0 (срок действия не истекает никогда).
- GPG запросит личную информацию, которая сохранится в ключе:
- Имя (не менее 5 символов).
- Адрес электронной почты.
- При вводе email убедитесь, что он соответствует адресу, используемому на GitHub.
- Комментарий

```
gpg: создан каталог '/home/dakhusainova/.gnupg'
gpg: создан щит с ключами '/home/dakhusainova/.gnupg/pubring.kbx'
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)
```

Рис. 2.5: Генерация ключа

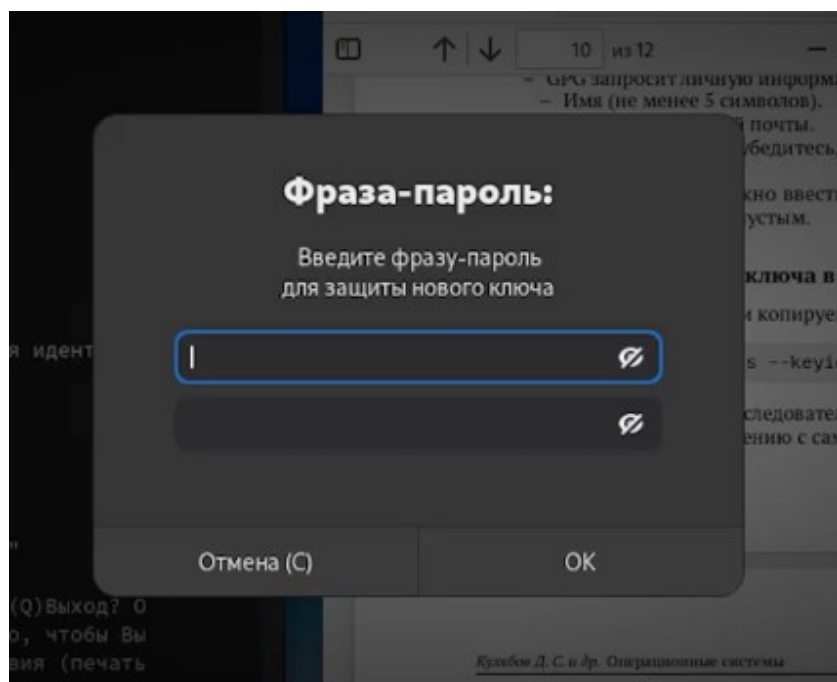


Рис. 2.6: Выбираем опции

Выводим список ключей и копируем отпечаток приватного ключа, устанавливаем необходимый пакет для дальнейших действий с ключом (рис. 2.7 -2.8).

```
[dakhusainova@fedora tmp]$ gpg --list-secret-keys --keyid-format LONG
```

Рис. 2.7: Список ключей

```
[dakhusainova@fedora tmp]$ gpg --armor --export 2B1064E240FD3A05 | xclip -sel clip
bash: xclip: command not found...
Install package 'xclip' to provide command 'xclip'? [N/y]
```

Рис. 2.8: Приватный ключ

Заходим на сам сайт GitHub, копируем сгенерированный PGP ключ в буфер обмена и вставляем полученный ключ в поле ввода (рис.2.9,2.10,2.11).



Рис. 2.9: Сайт

GPG keys / Add new

Key

Begins with '-----BEGIN PGP PUBLIC KEY BLOCK-----'

Рис. 2.10: Открываем

GPG keys / Add new

Key

```
X2eRqTu1G7OCrkfsJ
DfrpSaRdkM2yPhRk7FOHD97jd0EXDIR5aAoGGa+AH42fOj
vbYn9E/O3LiTg24ozN
yPwlKZAPk9jNA4qnbMbTdisdwbcORFnwXWMxFDovE/gfv9
dky+UNwGSSwo5sQQLV
ym8Zi3q2gBdEFEP3ClxX
=iwVU
-----END PGP PUBLIC KEY BLOCK-----
```

Add GPG key

Рис. 2.11: Вставляем полученный ключ

Настраиваем автоматические подписи коммитов git, отвечаем на вопросы, выбирая GitHub,SSH,yes,вводим пароль, «залогиниться через браузер»(рис.2.12-2.13).

```
[dakhusainova@fedora tmp]$ gpg --armor --export 2B1064E240FD3A05 | xclip -sel clip
[dakhusainova@fedora tmp]$ gpg --armor --export 2B1064E240FD3A05 | xclip -sel clip
[dakhusainova@fedora tmp]$ git config --global user.signingkey volkov_lenya2000@mail.ru
[dakhusainova@fedora tmp]$ git config --global commit.gpgsign true
[dakhusainova@fedora tmp]$ git config --global gpg.program $(which gpg2)
[dakhusainova@fedora tmp]$ gpg --auth login
```

Рис. 2.12: Коммиты git

```
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations? SSH
? Generate a new SSH key to add to your GitHub account? Yes
? Enter a passphrase for your new SSH key (Optional) *****
? How would you like to authenticate GitHub CLI? Login with a web browser

! First copy your one-time code: 154B-F60C
Press Enter to open github.com in your browser...
```

Рис. 2.13: Отвечаем на необходимые вопросы

Получаем код в терминале и вводим его на сайте(рис.2.14).

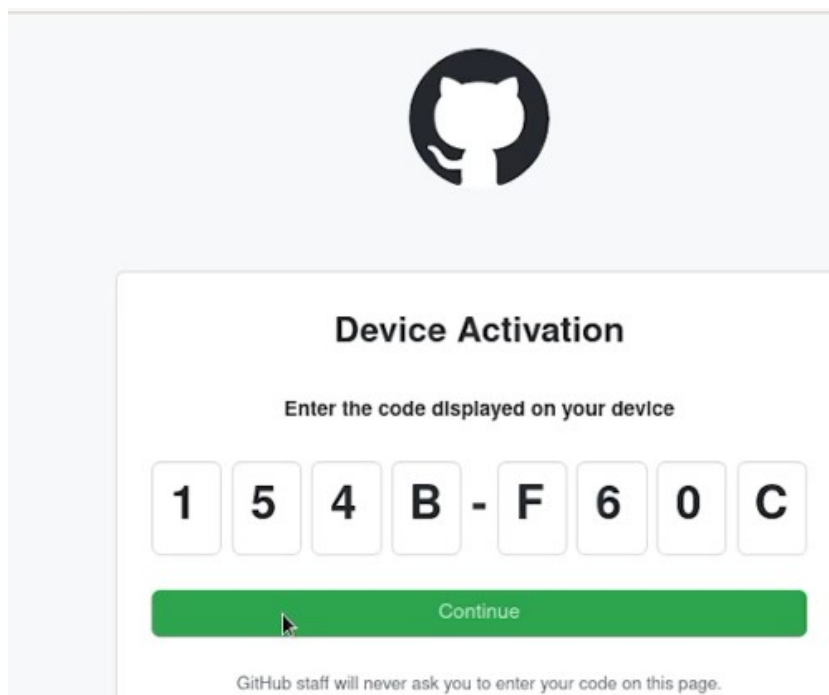


Рис. 2.14: Пишем код из терминала

Переходим в каталог курса, удаляем ненужные файлы, создаем, создаем каталог os-intro и отправляем файлы на сервер(рис. 2.15,2.16).

```
[dakhusainova@fedora Операционные системы]$ cd ~/work/study/2021-2022/"Операционные с
истемы"/os-intro
[dakhusainova@fedora os-intro]$ rm package.json
[dakhusainova@fedora os-intro]$ make COURSE=os-intro
[dakhusainova@fedora os-intro]$ git commit -am 'feat(main): make course structure'
```

Рис. 2.15: Прodelываем выше описанные действия

```
[dakhusainova@fedora os-intro]$ git push
Перечисление объектов: 20, готово.
Подсчет объектов: 100% (20/20), готово.
Сжатие объектов: 100% (16/16), готово.
Запись объектов: 100% (19/19), 266.56 КиБ | 2.34 МБ/с, готово.
Всего 19 (изменений 2), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To github.com:dakhusainova/study_2021-2022_os-intro.git
   c005604..35cacle  master -> master
[dakhusainova@fedora os-intro]$
```

Рис. 2.16: Отправляем файлы

3 Контрольные вопросы

1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначены?

Система контроля версий (Version Control System, VCS) — программное обеспечение для облегчения работы с изменяющейся информацией. и VCS позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое.

2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.

- Commit (Коммит) – сохранение изменений в репозиторий.
- Хранилище (repository), или репозиторий, — место хранения всех версий и служебной информации.
- История — список всех изменений проекта с возможностью отката в любую точку истории.
- Рабочая копия (working copy) — текущее состояние файлов проекта (любой версии), полученных из хранилища и, возможно, измененных.

3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида.

Централизованные системы — это системы, которые используют архитектуру клиент / сервер, где один или несколько клиентских узлов напрямую подключены к центральному серверу. Это наиболее часто используемый тип систем во многих

организациях, где клиент отправляет запрос на сервер компании и получает ответ. - Wikipedia. Рассмотрим огромный сервер, на который мы отправляем наши запросы, и сервер отвечает запрашиваемой статьей. Предположим, мы ввели поисковый запрос «нездоровая пища» в строке поиска Википедии. Этот поисковый запрос отправляется как запрос на серверы Википедии (в основном, расположенные в штате Вирджиния, США), которые затем возвращают статьи, основанные на релевантности. В этой ситуации мы являемся клиентским узлом, серверы Википедии являются центральным сервером.

В децентрализованных системах каждый узел принимает свое собственное решение. Конечное поведение системы является совокупностью решений отдельных узлов. Обратите внимание, что нет единого объекта, который получает и отвечает на запрос - Bitcoin. Давайте возьмем биткойны, например, потому что это самый популярный пример использования децентрализованных систем. Ни одна организация / организация не владеет сетью биткойнов. Сеть представляет собой сумму всех узлов, которые общаются друг с другом для поддержания количества биткойнов, которое есть у каждого владельца счета.

4. Опишите действия с VCS при единоличной работе с хранилищем.

Инициализация системы управления версиями git через git init. Работа над проектом используя git-flow для отдельных частей проекта. Git commit для фиксации изменений. При необходимости использование удаленного сервера для хранения с помощью remote и git push. Удаленный сервер также позволяет работать с нескольких устройств с использованием git pull.

5. Опишите порядок работы с общим хранилищем VCS.

При существующей версии проекта в хранилище, скопировать его оттуда через git pull. Использовать git-flow для работы над частями проекта. После окончания работы зафиксировать изменения через git commit и загрузить в хранилище через git push.

6. Каковы основные задачи, решаемые инструментальным средством git?

Git — это система управления версиями. У Git две основных задачи: первая —

хранить информацию о всех изменениях в вашем коде, начиная с самой первой строки, а вторая — обеспечение удобства командной работы над кодом. Ведение истории изменений, фиксирование изменений, совмещение версий, веток и др., а также откат к прошлым версиям.

7. Назовите и дайте краткую характеристику командам git.

- `git init` — инициализация проекта с системой контроля версий.
- `git add` — добавление файла/директории в систему контроля версий как отслеживаемое.
- `git commit` — фиксация изменений в отслеживаемых файлах.

8. Приведите примеры использования при работе с локальным и удалённым репозиториями.

При работе с локальным репозиторием все изменения хранятся локально и не выгружаются на удаленный сервер. Не требуется использование команд `push`, `pull`, `remote` и т.д. При работе с удаленным репозиторием для отображения изменения на удаленном репозитории и его актуализации, последние изменения должны быть загружены на удаленный сервер.

9. Что такое и зачем могут быть нужны ветви (branches)?

Ветка (англ. `branch`) — это последовательность коммитов, в которой ведётся параллельная разработка какого-либо функционала. Основная ветка — `master`. Ветки в GIT. Показать все ветки, существующие в репозитории `git branch`. Создать ветку `git branch имя`.

10. Как и зачем можно игнорировать некоторые файлы при `commit`? Игнорирование файлов при `commit` происходит с помощью `.gitignore` файла.

В нем указываются пути, названия, расширения и другие идентификации нежелательных объектов которые не будут учитываться в `commit`. Это полезно для исключения как “мусорных” файлов, которые не являются значимой частью проекта, а также конфиденциальных файлов, которые содержат в себе приватную информацию, такую как пароли и токены.

4 Вывод

Мы изучили идеологию и применение средств контроля версий и освоили умения по работе с git.