Лабораторная работа №3

Markdown

Камкина Арина Леонидовна

Содержание

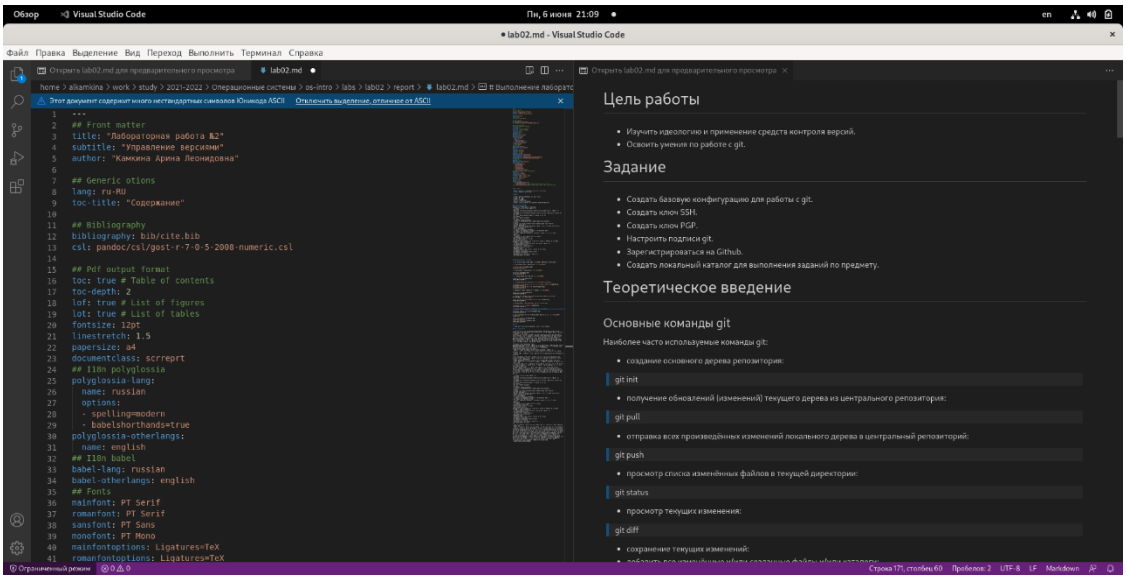
-Чтобы вложить один список в другой, добавьте отступ для элементов дочернего списка:

- List item 1
-
  - List item A
-
  - List item B
- **Упорядоченный список можно отформатировать с помощью соответствующих цифр:** > 1. First instruction > 2. Second instruction
- **Синтаксис** Markdown **для встроенной ссылки состоит из части** [link text] , **представляющей текст гиперссылки, и части** (file-name.md) – **URL-адреса или имени файла, на который дается ссылка:** > link text

Выполнение лабораторной работы

1. **Для начала создала установила** Visual Studio Code **на** Linux Fedora **и** pandoc, **который понадобиться, чтобы поменять расширение.**

Так выглядит окно VS Code **с** Markdown, **слева - окно, в которм я работаю, справа - окно, чтобы смотреть, что в итоге выходит. (рис.** [-@fig:001])



{ #fig:001 width=70% }

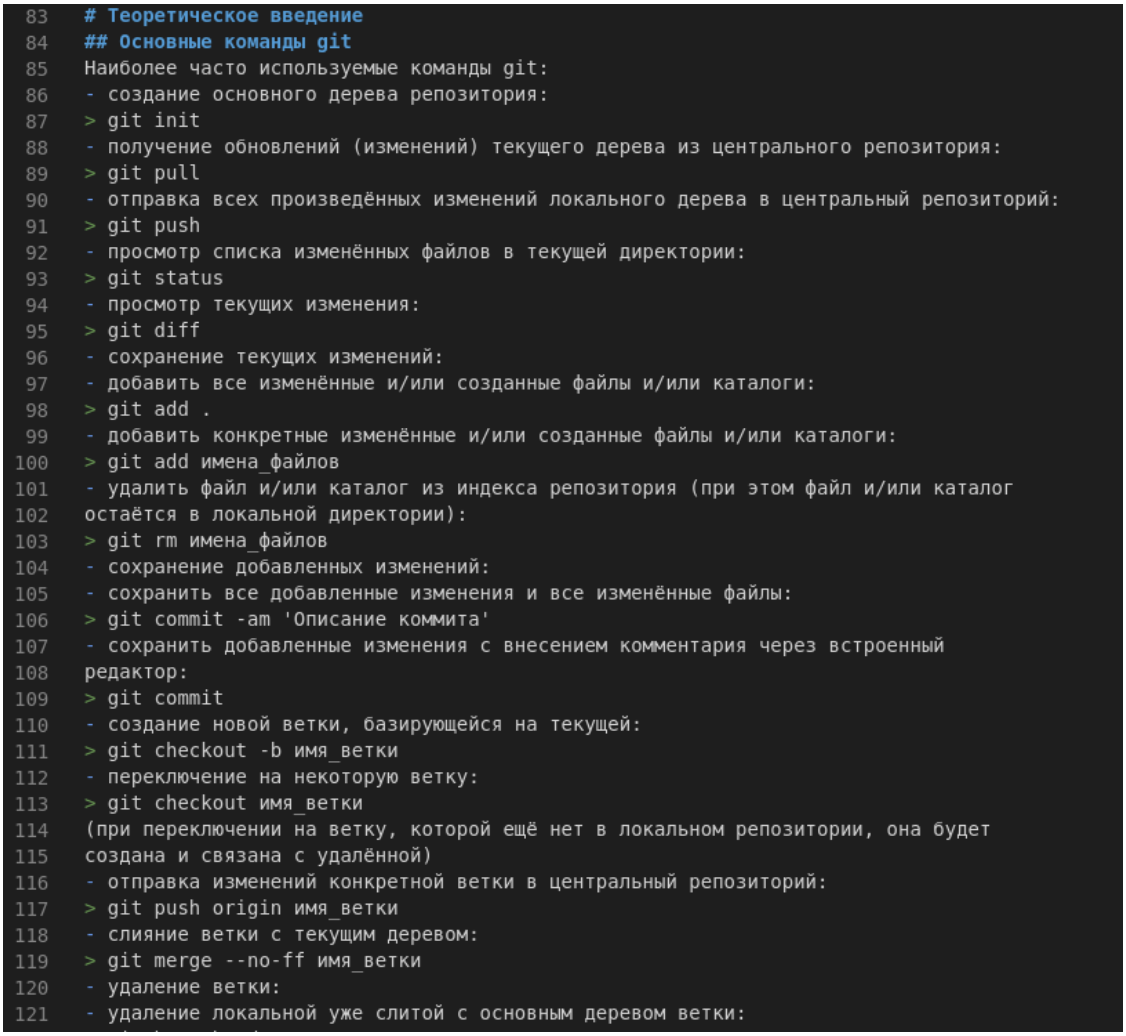
2. **Введение данных в титульный лист (рис.** [-@fig:002])

```
---
## Front matter
title: "Лабораторная работа №2"
subtitle: "Управление версиями"
author: "Камкина Арина Леонидовна"
```
{ #fig:002 width=70% }

3. **Написала цели и задания, использу я знаки** “-” **для списка (рис.** [-@fig:003])

```

# Цель работы

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

# Задание

- Создать базовую конфигурацию для работы с git.
- Создать ключ SSH.
- Создать ключ PGP.
- Настроить подписи git.
- Зарегистрироваться на Github.
- Создать локальный каталог для выполнения заданий по предмету.

```
{ #fig:003 width=70% }

4. **Написала теоретическое введение используя два вида** #Head **(рис.** [-@fig:004])



```
# Теоретическое введение
## Основные команды git
Наиболее часто используемые команды git:
- создание основного дерева репозитория:
> git init
- получение обновлений (изменений) текущего дерева из центрального репозитория:
> git pull
- отправка всех произведённых изменений локального дерева в центральный репозиторий:
> git push
- просмотр списка изменённых файлов в текущей директории:
> git status
- просмотр текущих изменения:
> git diff
- сохранение текущих 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
- удалить файл и/или каталог из индекса репозитория (при этом файл и/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'
- сохранить добавленные изменения с внесением комментария через встроенный
редактор:
> git commit
- создание новой ветки, базирующейся на текущей:
> git checkout -b имя_ветки
- переключение на некоторую ветку:
> git checkout имя_ветки
(при переключении на ветку, которой ещё нет в локальном репозитории, она будет
создана и связана с удалённой)
- отправка изменений конкретной ветки в центральный репозиторий:
> git push origin имя_ветки
- слияние ветки с текущим деревом:
> git merge --no-ff имя_ветки
- удаление ветки:
- удаление локальной уже слитой с основным деревом ветки:
```

{ #fig:004 width=70% }

5. **Описала ход работы и вставила картинки (рис.** [-@fig:005])

```
# Выполнение лабораторной работы

1. Для начала создала учётную запись на github и заполнила основные данные.

2. Установила git-flow в Fedora Linux (рис. [-@fig:001])

![Установка git-flow](image/1.png)
{ #fig:001 width=70% }

3. Установила gh в Fedora Linux (рис. [-@fig:002])

![Установка gh](image/2.png)
{ #fig:002 width=70% }

4. Произвела базовую настройку git (рис. [-@fig:003])

![Базовая настройка](image/3.png)
{ #fig:003 width=70% }

5. Произвела базовую настройку git (рис. [-@fig:004] [-@fig:005])

![создание ключа ssh по алгоритму rsa с ключём размером 4096 бит](image/4.png)
{ #fig:004 width=70% }
![создание ключа ssh по алгоритму ed25519](image/5.png)
{ #fig:004 width=70% }

6. Создала ключ pgp и добавила его в github (рис. [-@fig:006])

![Список ключей pgp](image/6.png)
{ #fig:007 width=70% }

7. Добавила его в github и настроила автоматические подписи коммитов git и
авторизовалась (рис. [-@fig:007])

![Настройка автоматических подписей коммитов git ](image/7.png)
{ #fig:007 width=70% }
```
{ #fig:005 width=70% }

6. **Написала вывод и ответила на контрольные вопросы (рис.** [-@fig:006])

```
183  # Выводы
184
185  Освоила работу с git и получила знания о некоторых его функциях.
186
187  # Контрольные вопросы
188
189  1. Системы контроля версий (Version Control System, VCS) применяются при работе
     нескольких человек над одним проектом. Системы контроля версий поддерживают возможность
     отслеживания и разрешения
190  конфликтов, которые могут возникнуть при работе нескольких человек над одним файлом.
     Можно объединить (слить) изменения, сделанные разными участниками (автоматически или
     вручную), вручную выбрать нужную версию, отменить изменения вовсе или заблокировать
     файлы для изменения. В зависимости от настроек блокировка не позволяет другим
     пользователям получить рабочую копию или препятствует изменению рабочей копии файла
     средствами файловой системы ОС, обеспечивая таким образом, привилегированный доступ
     только одному пользователю, работающему с файлом.
191  2.
192  - Хранилище - централизованная модель,
193  предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства
     функций по управлению версиями осуществляется специальным сервером. Участник проекта
     (пользователь) перед началом работы посредством определённых
194  команд получает нужную ему версию файлов
195  - Commit - команда для записи индексированных изменений в репозиторий.
196  - История - место, где сохраняются все коммиты, по которым можно посмотреть данные о
     коммитах.
197  - Рабочая копия - текущее состояние  файлов пректа основанное на версии, загруженной из
     хранилища.
198  3.
199  - Централизованные системы — это системы, в которых одно основное хранилище всего
     проекта, и каждый пользователь копирует необходимые ему файлы, изменяет и вставляет
     обратно. Пример — Subversion.
200  - Децентрализованные системы — система, в которой каждый пользователь имеет свой
     вариант репозитория и есть возможность добавлять и забирать изменения из репозиториев.
     Пример — Git.
201  4. В рабочей копии, которую исправляет человек, появляются правки, которые отправляются
     в хранилище на каждом из этапов. То есть в правки в рабочей копии появляются, только
     если человек делает их.
```
{ #fig:006 width=70% }

Выводы

Я научилась оформлять отчёты и презентации с помощью легковесного языка разметки Markdown **и менять расширение с** md **на** docx **и** pdf.