

# **Лабораторная работа №2**

**Управление версиями**

Крутова Екатерина Дмитриевна

# Содержание

<b>1</b>	<b>Цель работы:</b>	<b>5</b>
<b>2</b>	<b>Теоретическое введение</b>	<b>6</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>7</b>
<b>4</b>	<b>Выводы</b>	<b>11</b>
<b>5</b>	<b>Контрольные вопросы</b>	<b>12</b>
<b>6</b>	<b>Список литературы</b>	<b>15</b>

# Список иллюстраций

3.1	Установка git-flow в Fedora Linux . . . . .	7
3.2	Установка git-flow в Fedora Linux . . . . .	7
3.3	Установка git-flow в Fedora Linux . . . . .	7
3.4	Установка gh в Fedora Linuxx . . . . .	7
3.5	Установка gh в Fedora Linux . . . . .	7

## Список таблиц

# 1 Цель работы:

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

## 2 Теоретическое введение

Система контроля версий Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды `git` с различными опциями. Благодаря тому, что Git является распределённой системой контроля версий, резервную копию локального хранилища можно сделать простым копированием или архивацией.

Наиболее часто используемые команды `git`:

- `git init` - создание основного дерева репозитория
- `git pull` - получение обновлений (изменений) текущего дерева из центрального репозитория
- `git push` - отправка всех произведённых изменений локального дерева в центральный репозиторий
- `git status` - просмотр списка изменённых файлов в текущей директории
- `git diff` - просмотр текущих изменений
- `git add` - сохранение текущих изменений
- `git checkout -b имя_ветки` - создание новой ветки, базирующейся на текущей
- `git checkout имя_ветки` - переключение на некоторую ветку
- `git push origin имя_ветки` - отправка изменений конкретной ветки в центральный репозиторий
- `git merge --no-ff имя_ветки` - слияние ветки с текущим деревом

## 3 Выполнение лабораторной работы

### 1. Установка программного обеспечения

#### 1.1. Установка git-flow в Fedora Linux (рисунки 3.1-3.4)

```
[edkrutova@fedora ~]$ cd /tmp
```

Рис. 3.1: Установка git-flow в Fedora Linux

```
[edkrutova@fedora tmp]$ wget --no-check-certificate -q https://raw.githubusercontent.com/petervanderdoes/gitflow/develop/contrib/gitflow-installer.sh
```

Рис. 3.2: Установка git-flow в Fedora Linux

```
[edkrutova@fedora tmp]$ chmod +x gitflow-installer.sh
```

Рис. 3.3: Установка git-flow в Fedora Linux

```
[edkrutova@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 edkrutova:
```

Рис. 3.4: Установка gh в Fedora Linuxx

#### 1.2. Установка gh в Fedora Linux (Рисунок 3.5)

```
[edkrutova@fedora tmp]$ sudo dnf install gh
```

Рис. 3.5: Установка gh в Fedora Linux

## 2. Базовая настройка git

2.1. Задавание имени и email владельца репозитория, настройка utf-8 в выводе сообщений git, настройка верификации и подписания коммитов git, задавание имени начальной ветки (будем называть её master), параметра autocrlf, параметра safecrlf (Рисунок ??)

```
[edkrutova@fedora tmp]$ git config --global user.name "Ekaterina Krutova"
[edkrutova@fedora tmp]$ git config --global user.email "1032216536@pfur.ru"
[edkrutova@fedora tmp]$ git config --global core.quotepath false
[edkrutova@fedora tmp]$ git config --global init.defaultBranch master
[edkrutova@fedora tmp]$ git config --global core.autocrlf input
[edkrutova@fedora tmp]$ git config --global core.safecrlf warn
```

## 3. Создание ключа ssh

3.1. Создание по алгоритму rsa с ключём размером 4096 бит (Рисунок 3)

```
[edkrutova@fedora tmp]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/edkrutova/.ssh/id_rsa): home
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in home
Your public key has been saved in home.pub
The key fingerprint is:
SHA256:CR2x3MfPhUQ+BWwnMjtviC2MhYMvgyYdJLqbtbZuPMw edkrutova@fedora
The key's randomart image is:
+---[RSA 4096]-----+
|      o.  o+..|
|    . . o + .oo+o.|
|  . o  ..+.. o=+o.|
| .    . .o...oo o |
| . . o .S= o +o |
| . o + o o + o o |
| B +  o  . . |
| o E |
| +oo |
+---[SHA256]-----+
```

## 4. Создание ключа pgp

4.1. Генерируем ключ (Рисунок ??)

```
[edkrutova@fedora tmp]$ gpg --full-generate-key
```

## 5. Добавление PGP ключа в GitHub

5.1. Вывод списка ключей и копирование отпечатка приватного ключа (Рисунок ??)



```
[edkrutova@fedora tmp]$ 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/edkrutova/.gnupg/pubring.kbx
-----
sec   rsa4096/53D711350CC39D28 2022-04-22 [SC]
      E073FEFA2F1E8072F5D998E353D711350CC39D28
uid           [ абсолютно ] edkrutova <1032216536@pfur.ru>
ssb   rsa4096/7C32ACF4DFFBDBC3 2022-04-22 [E]
```

5.2. Копирование сгенерированного PGP ключа в буфер обмена (чтобы вставить его в настройки GitHub (<https://github.com/settings/keys>)) (Рисунок ??)

```
[edkrutova@fedora tmp]$ gpg --armor --export | xclip -sel clip
```

## 6. Настройка автоматических подписей коммитов git

6.1. Используя введенный email, указывание Git применять его при подписи коммитов (Рисунок 6)

```
[edkrutova@fedora tmp]$ git config --global user.signingkey 53D711350CC39D28
[edkrutova@fedora tmp]$ git config --global commit.gpgsign true
[edkrutova@fedora tmp]$ git config --global gpg.program $(which gpg2)
```

## 7. Настройка gh

7.1. Авторизация, ответы на вопросы утилиты (Рисунок ??)

```
[edkrutova@fedora tmp]$ gh auth login
? 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
```

## 8. Создание репозитория курса на основе шаблона

8.1. Создание шаблона рабочего пространства, создание требуемого вида (Рисунок ??-??)

```
✓ Logged in as edkrutova
[edkrutova@fedora tmp]$ mkdir -p ~/work/study/2021-2022/"Операционные системы"
[edkrutova@fedora Операционные системы]$ gh repo create study_2021-2022_os-intro --template=yamadharma/course-directory-student-template --public
✓ Created repository edkrutova/study_2021-2022_os-intro on GitHub
[edkrutova@fedora Операционные системы]$ git clone --recursive git@github.com:edkrutova/study_2021-2022_os-intro.git os-intro
```

## 9. Настройка каталога курса

### 9.1. Переход в каталог курса (Рисунок ??)

```
[edkrutova@fedora Операционные системы]$ cd ~/work/study/2021-2022/"Операционные системы"/os-intro
```

### 9.2. Удаление лишних файлов (Рисунок ??)

```
[edkrutova@fedora os-intro]$ rm package.json
```

### 9.3. Создание необходимых каталогов (Рисунок ??)

```
[edkrutova@fedora os-intro]$ make COURSE=os-intro
```

### 9.4. Отправка файлов на сервер (Рисунки ??-??)

```
[edkrutova@fedora os-intro]$ git add .  
[edkrutova@fedora os-intro]$ git commit -am 'feat(main): make course structure'
```

```
[edkrutova@fedora os-intro]$ git push
```

## 4 Выводы

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

## 5 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. 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.

Хранилище — сервер, на котором хранится вся история изменений проекта.

Commit — фиксация “дельта-изменений”, т.е. изменений с последнего commit’a с его последующей записью как версии в истории.

История — список всех изменений проекта с возможностью отката в любую точку истории.

Рабочая копия — все файлы проекта, с которыми происходит основная работа.

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

В централизованных VCS необходим центральный репозиторий для хранения файлов. Примером таковых могут служить CVS и Subversion. В децентрализованных VCS наличие центрального репозитория не обязательно.

Децентрализованными VCS являются Git, Bazaar и Mercurial.

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?

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
- git push — загрузка локальной версии на сервер
- git pull — выгрузка актуальной версии с сервера
- git fetch — “часть” команды git pull, которая собирает актуальную версию, но не вносит её в работу
- git merge — слияние веток

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)?

Ветви позволяют “разделять” части работы и работать отдельно над каждой имплементацией. Использование ветвей дает возможность комфортной ревизии и обработки нововведений в основную ветвь, которая чаще всего является релизной.

#### 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.

## **6 Список литературы**