Лабораторная работа №2

Управление версиями

Крутова Екатерина Дмитриевна

Содержание

# 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 изменени
* 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 1-4)

Рис. 1: Установка git-flow в Fedora Linux

Рис. 1: Установка git-flow в Fedora Linux

Рис. 2: Установка git-flow в Fedora Linux

Рис. 2: Установка git-flow в Fedora Linux

Рис. 3: Установка git-flow в Fedora Linux

Рис. 3: Установка git-flow в Fedora Linux

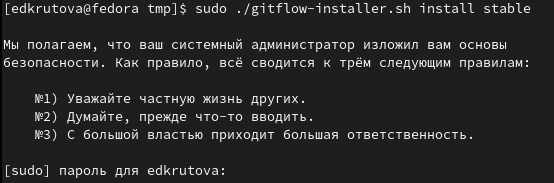


Рис. 4: Установка gh в Fedora Linuxx

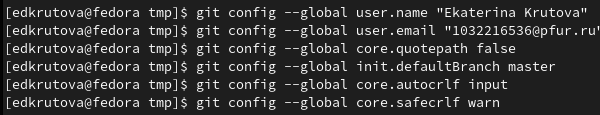
1.2. Установка gh в Fedora Linux (Рисунок 5)

Рис. 5: Установка gh в Fedora Linux

Рис. 5: Установка gh в Fedora Linux

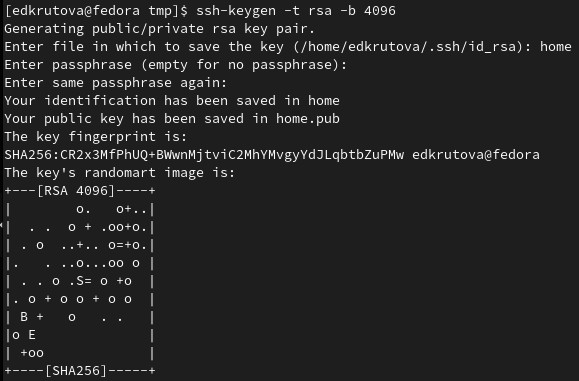
1. Базовая настройка git

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 **¿fig:006?**)



1. Создание ключа shh

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



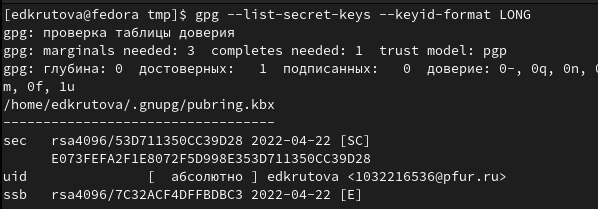
1. Создание ключа pgp

4.1. Генерируем ключ (Рисунок **¿fig:008?**)



1. Добавление PGP ключа в GitHub

5.1. Вывод списка ключей и копирование отпечатка приватного ключа (Рисунок **¿fig:009?**)

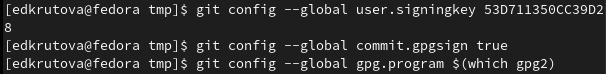


5.2. Копирование сгенерированного PGP ключа в буфер обмена (чтобы вставить его в настройки GitHub (https://github.com/settings/keys)) (Рисунок **¿fig:010?**)



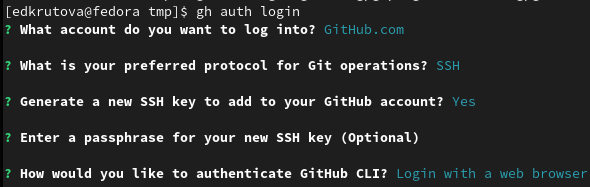
1. Настройка автоматических подписей коммитов git

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



1. Настройка gh

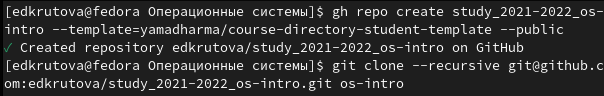
7.1. Авторизация, ответы на вопросы утилиты (Рисунок **¿fig:012?**)



1. Сознание репозитория курса на основе шаблона

8.1. Создание шаблона рабочего пространства, создание требуемого вида (Рисунки **¿fig:013?**-**¿fig:014?**)





1. Настройка каталога курса

9.1. Переход в каталог курса (Рисунок **¿fig:015?**)



9.2. Удаление лишних файлов (Рисунок **¿fig:016?**)



9.3. Создание необходимых каталогов (Рисунок **¿fig:017?**)



9.4. Отправка файлов на сервер (Рисунки **¿fig:018?**-**¿fig:019?**)





# 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 предназначаются?

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.

1. 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.

1. 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.

1. 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

1. 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.

1. Каковы основные задачи, решаемые инструментальным средством git?

Ведение истории изменений, фиксирование изменений, совмещение версий, веток и др., а также откат к прошлым версиям.

1. 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

1. 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.

1. 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.

1. 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