Лабораторная работа №2

Первоначальная настройка Git

Козлов Всеволод Павлович

Содержание

# 1 Цель работы

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

# 2 Задание

1. Создать базовую конфигурацию для работы с git.
2. Создать ключ SSH.
3. Создать ключ PGP.
4. Зарегистрироваться на Github.
5. Создать локальный каталог для выполнения заданий по предмету.

# 3 Выполнение лабораторной работы

1. Установка программного обеспечения:

Произвел установку Git через терминал (рис. [[1](#fig:001)])

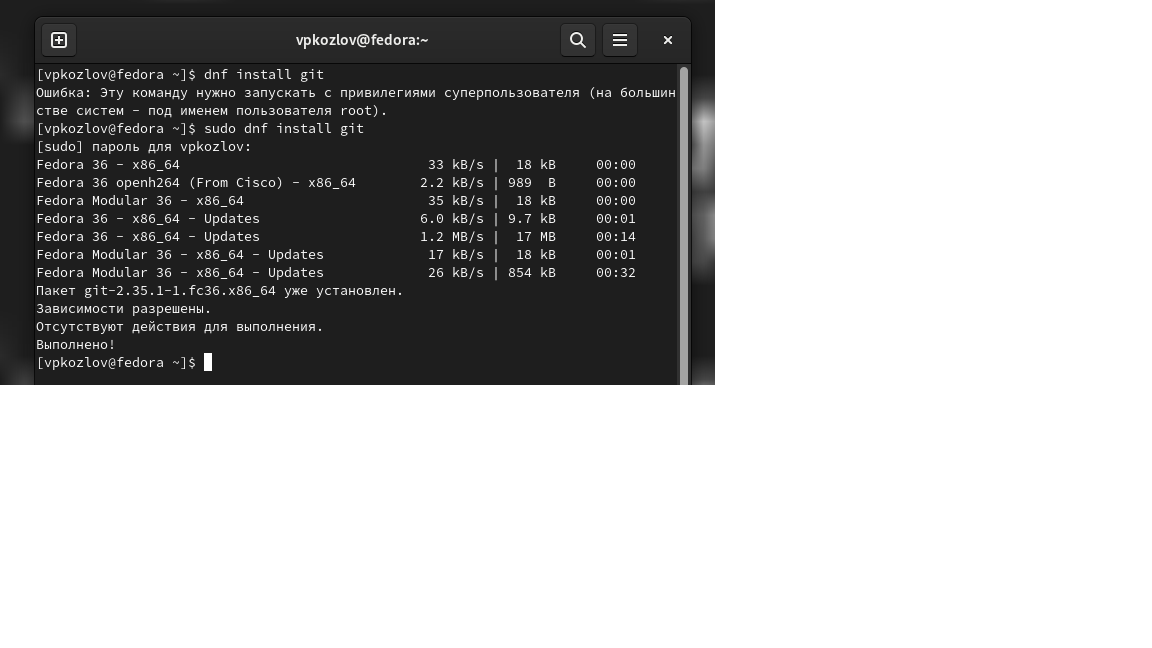


Figure 1: Установка Git

Произвел установку gh (рис. [[2](#fig:002)])



Figure 2: Установка gh

1. Базовая настройка Git:

Задал имя и почту пользователя (рис. [[3](#fig:003)])



Figure 3: Имя и почта пользователя

Настроил utf-8 в выводе сообщений git (рис. [[4](#fig:004)])



Figure 4: Настройка utf-8

Создал GPG ключ (рис. [5](#fig:005))

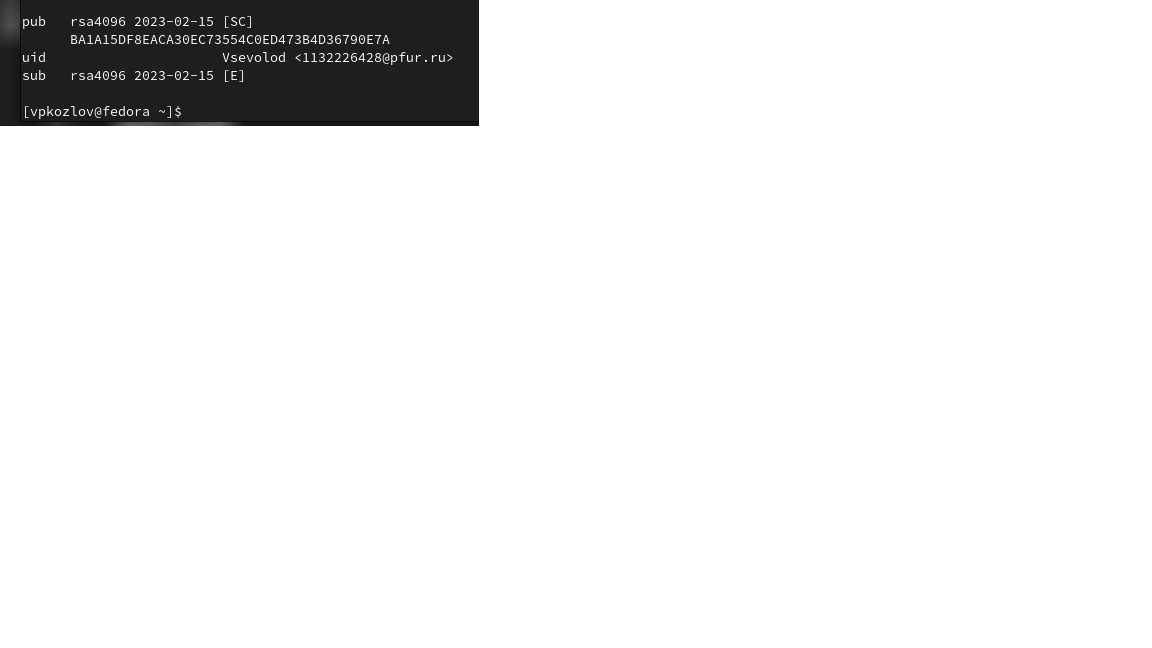


Figure 5: Создание GPG ключа

Экспортировал GPG ключ (рис. [[6](#fig:006)])

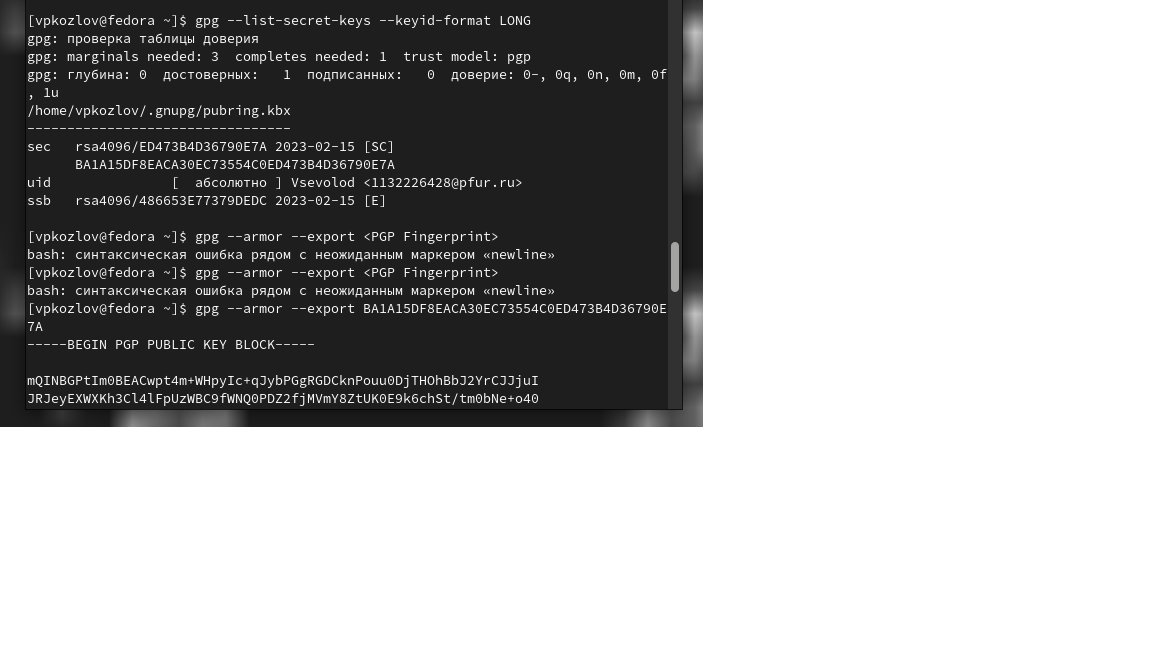


Figure 6: Экспорт ключа

Скопировал GPG ключ (рис. @[-fig:007])

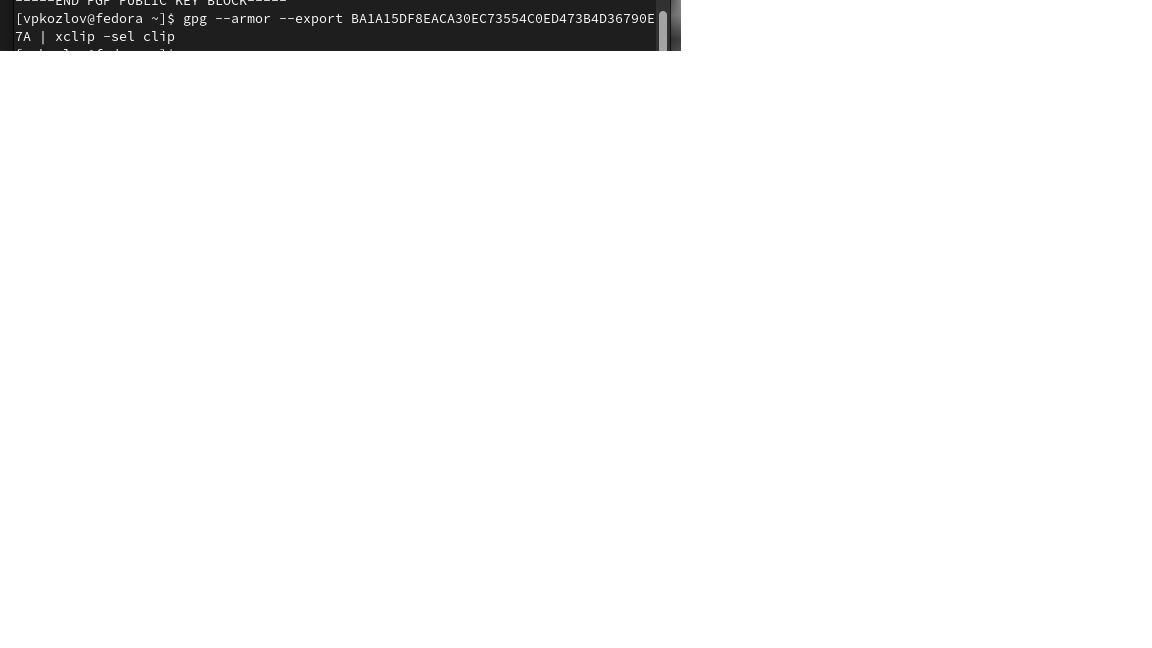


Figure 7: Копирование ключа

Передал GPG ключ на Github (рис. [[8](#fig:008)])

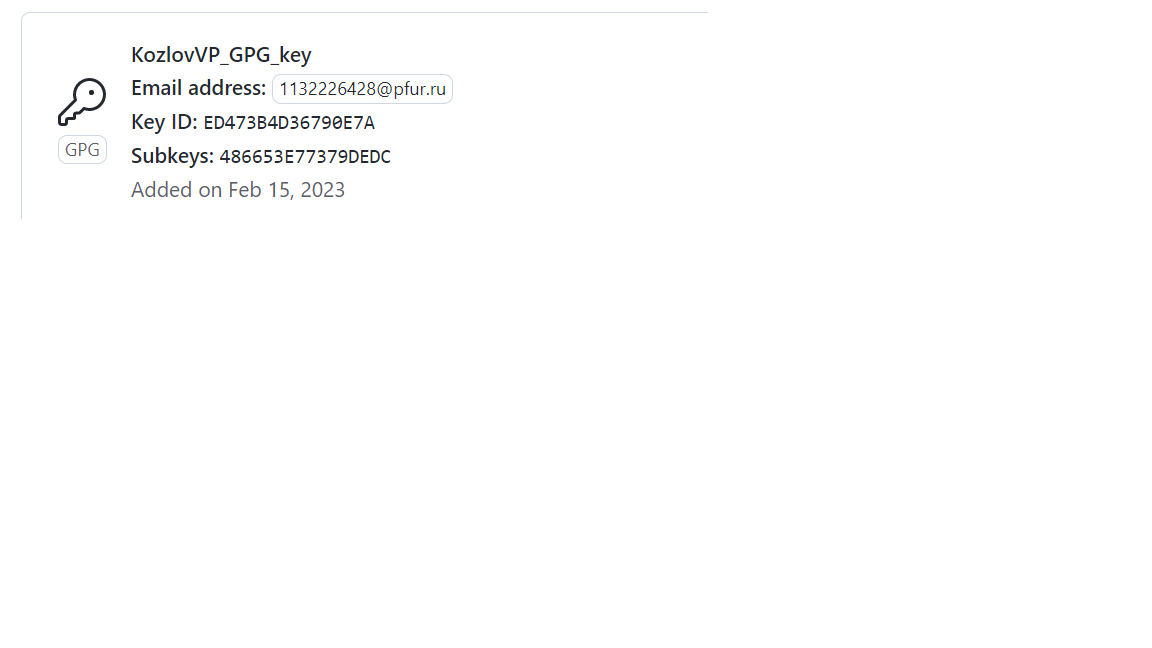


Figure 8: Передача ключа на Github

Настройка автоматических подписей коммитов git (рис. [[9](#fig:009)])



Figure 9: Настройка автоматических подписей

Задал имя начальной ветви, настроил пареметры autocrlf и safecrlf (рис. [[10](#fig:010)])



Figure 10: Задание начальной ветви, настройка autocrlf и safecrlf

1. Создание ключа SSH:

Создал SSH ключ по алгоритму rsa размером 4096 бит (рис. [[11](#fig:011)])

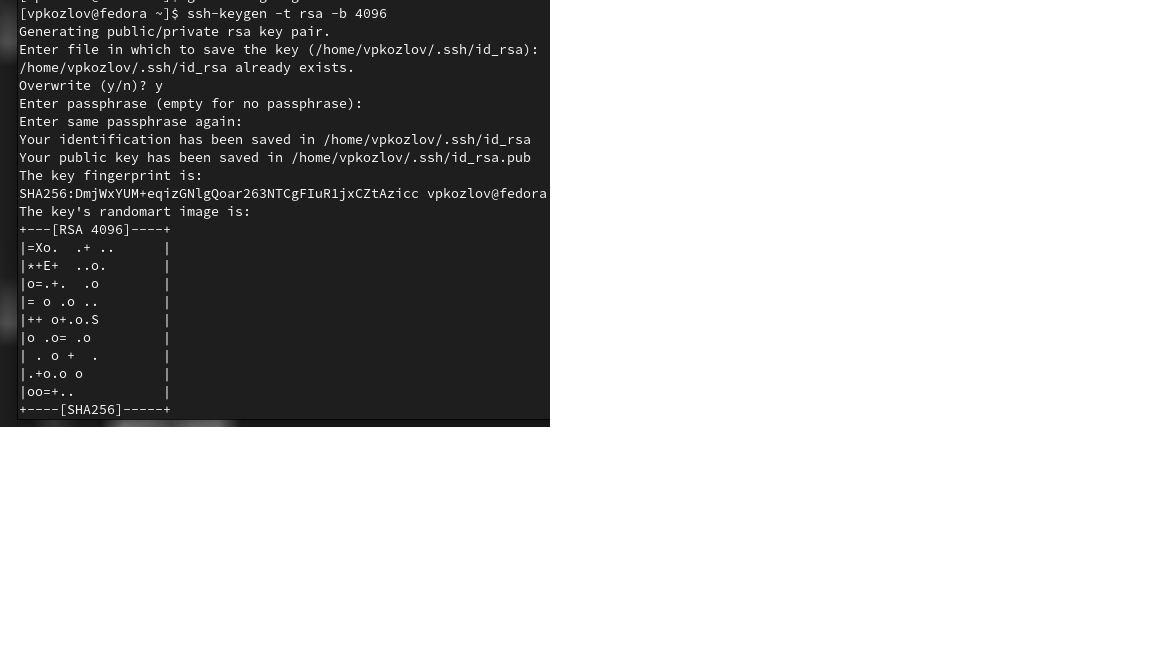


Figure 11: Создание ключа SSH через rsa

Создал SSH ключ по алгоритму ed25519 (рис. [[12](#fig:012)])



Figure 12: Создание SSH ключа через ed25519

1. Создание ключа PGP - сделано выше (рис. 5-9)
2. Регистрация на Github - произведена ранее
3. Настройка gh:

Авторизировался на gh (рис. [[13](#fig:013)])

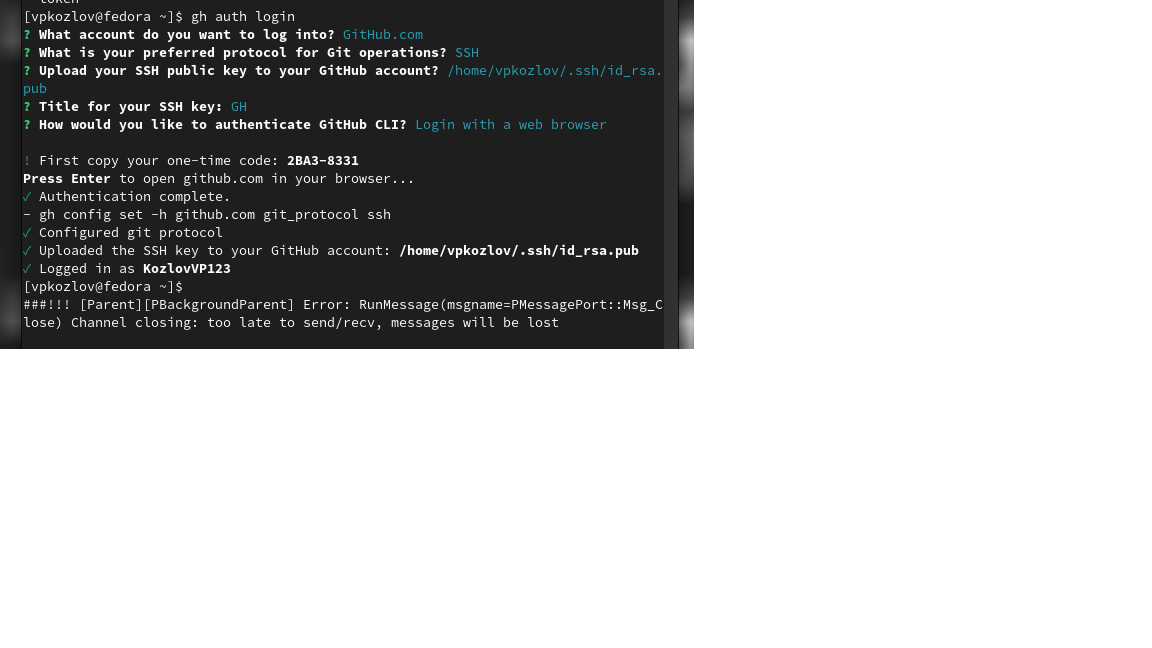


Figure 13: Авторизация на gh

1. Создание репозитория курса на основе шаблона:

Потребовалась помощь Татьяны Рефатовны, забыл сделать скриншот

1. Настройка каталога курса:

Удалил лишние файлы и создал необходимые каталоги (рис. [[14](#fig:014)])

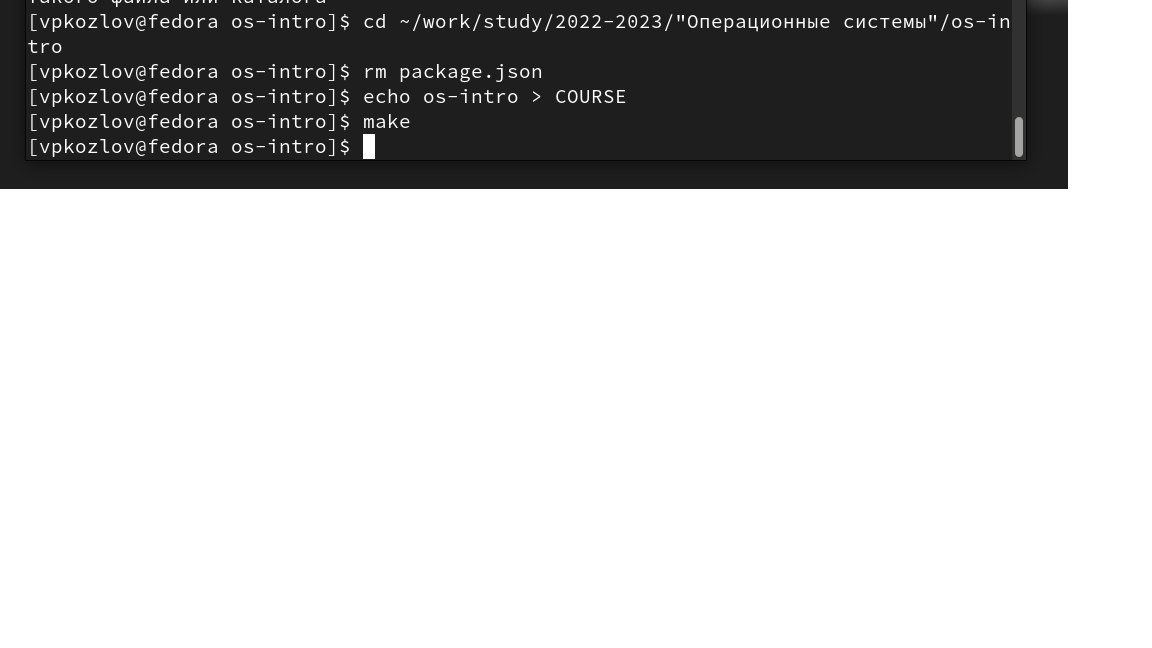


Figure 14: Удаление лишних файлов, создание каталогов

Отправил файлы на сервер (add, commit) (рис. [[15](#fig:015)])

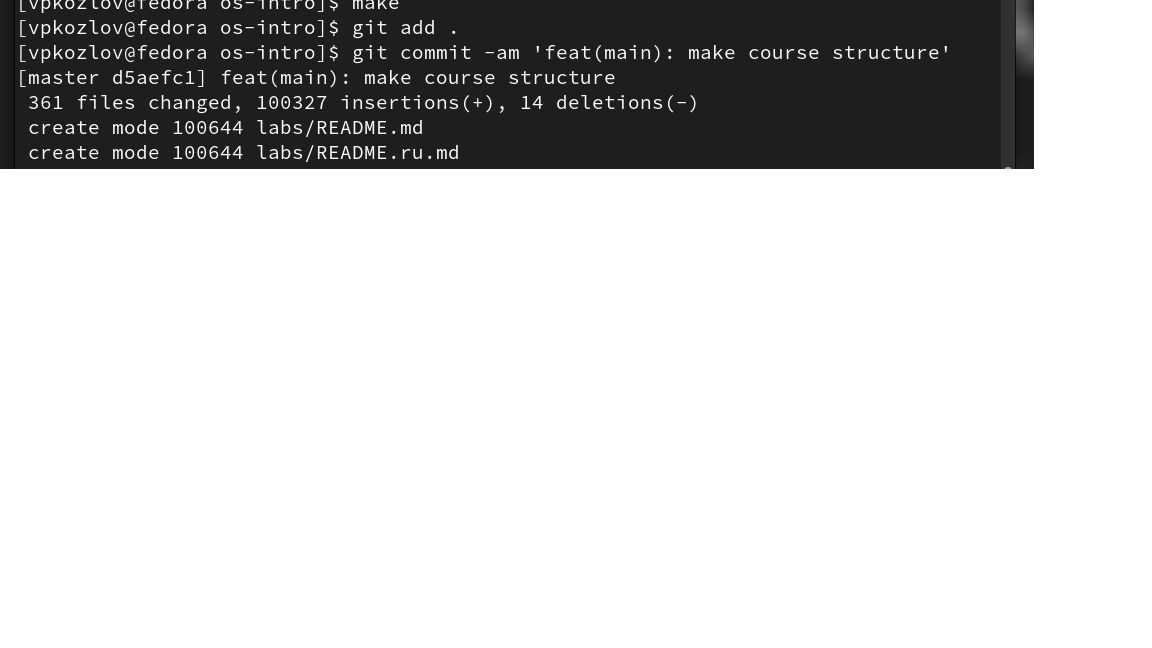


Figure 15: Отправка файлов на сервер: add, commit

Отправил файлы на сервер (push) (рис. [[16](#fig:016)])

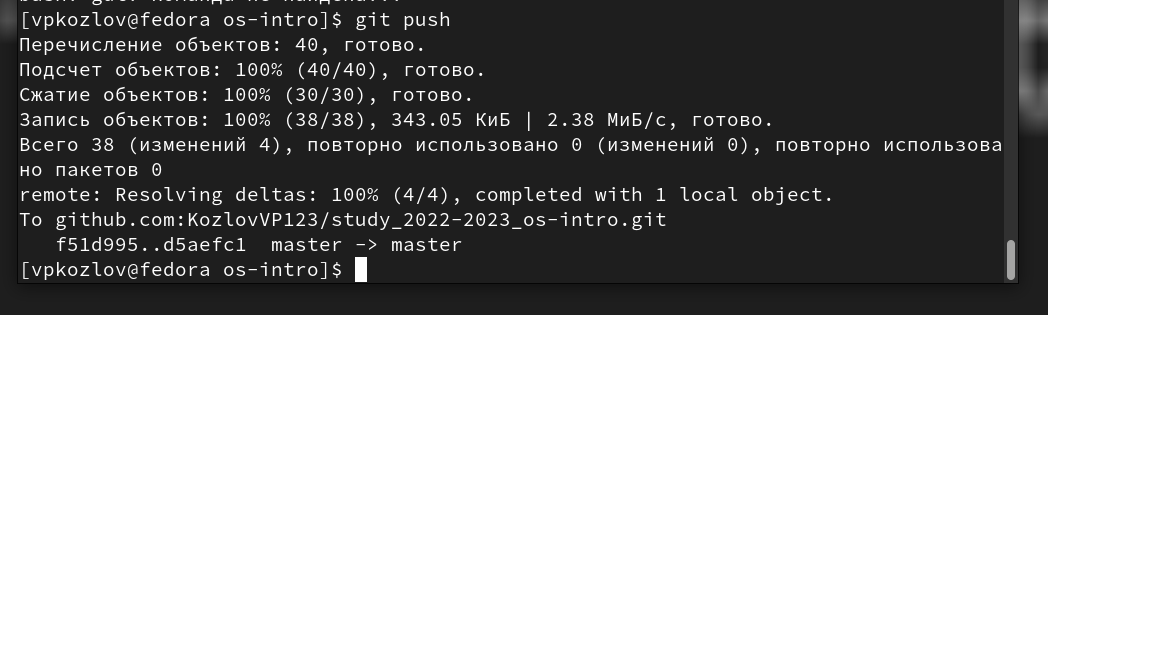


Figure 16: Отправка файлов на сервер: push

Ответы на контрольные вопросы:

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

Система контроля версий — программное обеспечение для облегчения работы с изменяющейся информацией. Система управления версиями 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. Системы контроля версий (Version Control System, VCS) применяются для: • Хранение полной истории изменений • причин всех производимых изменений • Откат изменений, если что-то пошло не так • Поиск причины и ответственного за появления ошибок в программе • Совместная работа группы над одним проектом • Возможность изменять код, не мешая работе других пользователей

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

Репозиторий - хранилище версий - в нем хранятся все документы вместе с историей их изменения и другой служебной информацией. Commit — отслеживание изменений, сохраняет разницу в изменениях Рабочая копия - копия проекта, связанная с репозиторием (текущее состояние файлов проекта, основанное на версии из хранилища (обычно на последней)) История хранит все изменения в проекте и позволяет при необходимости обратиться к нужным данным.

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

Централизованные VCS (Subversion; CVS; TFS; VAULT; AccuRev): • Одно основное хранилище всего проекта • Каждый пользователь копирует себе необходимые ему файлы из этого репозитория, изменяет и, затем, добавляет свои изменения обратно Децентрализованные VCS (Git; Mercurial; Bazaar): • У каждого пользователя свой вариант (возможно не один) репозитория • Присутствует возможность добавлять и забирать изменения из любого репозитория [2] В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером. В отличие от классических, в распределённых системах контроля версий центральный репозиторий не является обязательным.

1. Опишите действия с VCS при единоличной работе с хранилищем

Сначала создаем и подключаем удаленный репозиторий. Затем по мере изменения проекта отправлять эти изменения на сервер.

1. Опишите порядок работы с общим хранилищем VCS.

Участник проекта (пользователь) перед началом работы посредством определённых команд получает нужную ему версию файлов. После внесения изменений, пользователь размещает новую версию в хранилище. При этом предыдущие версии не удаляются из центрального хранилища и к ним можно вернуться в любой момент.

1. Каковы основные задачи, решаемые инструментальным средством git?

Первая — хранить информацию о всех изменениях в вашем коде, начиная с самой первой строчки, а вторая — обеспечение удобства командной работы над кодом.

1. Назовите и дайте краткую характеристику командам 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: git branch -d имя\_ветки – принудительное удаление локальной ветки: git branch -D имя\_ветки – удаление ветки с центрального репозитория: git push origin :имя\_ветки

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

git push –all (push origin master/любой branch)

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

Ветвление («ветка», branch) — один из параллельных участков истории в одном хранилище, исходящих из одной версии (точки ветвления). [3] • Обычно есть главная ветка (master), или ствол (trunk). • Между ветками, то есть их концами, возможно слияние. Используются для разработки новых функций.

1. Как и зачем можно игнорировать некоторые файлы при commit?

Во время работы над проектом так или иначе могут создаваться файлы, которые не требуется добавлять в последствии в репозиторий. Например, временные файлы, создаваемые редакторами, или объектные файлы, создаваемые компиляторами. Можно прописать шаблоны игнорируемых при добавлении в репозиторий типов файлов в файл .gitignore с помощью сервисов.

# 4 Выводы

Изучил идеологию и применение средств контроля версий, освоил умения по работе с git.

# Список литературы