Отчёт по лабораторной работе № 2

НММбд-02-22

Никулина Ксения Ильинична

Содержание

# 1 Цель работы

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

# 2 Задание

* Установить и настроить ПО для работы с git.

# 3 Выполнение лабораторной работы

## 3.1 Установка программного обеспечения

Установили git:(рис. [[1](#fig:001)])

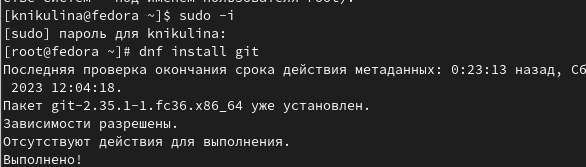


Figure 1: .

Установили gh:(рис. [[2](#fig:002)])

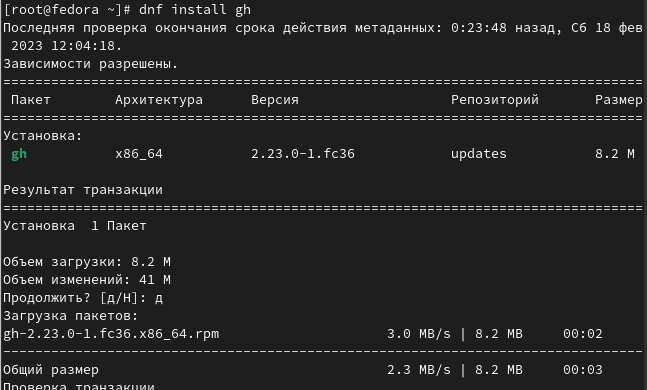


Figure 2: .

## 3.2 Базовая настройка git

Задали имя и email владельца репозитория: (рис. [[3](#fig:003)])

Figure 3: .

Figure 3: .

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

Figure 4: .

Figure 4: .

Настроили верификацию и подписание коммитов git. Задали имя начальной ветки (будем называть её master).(рис. [[5](#fig:005)])

Figure 5: .

Figure 5: .

Параметр autocrlf:(рис. [[6](#fig:006)])

Figure 6: .

Figure 6: .

Параметр safecrlf: (рис. [[7](#fig:007)])

Figure 7: .

Figure 7: .

## 3.3 Создали ключи ssh

по алгоритму rsa с ключём размером 4096 бит: (рис. [[8](#fig:008)])

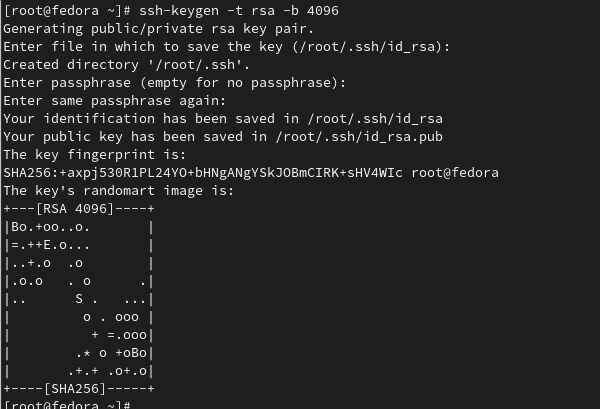


Figure 8: .

по алгоритму ed25519: (рис. [[9](#fig:009)])

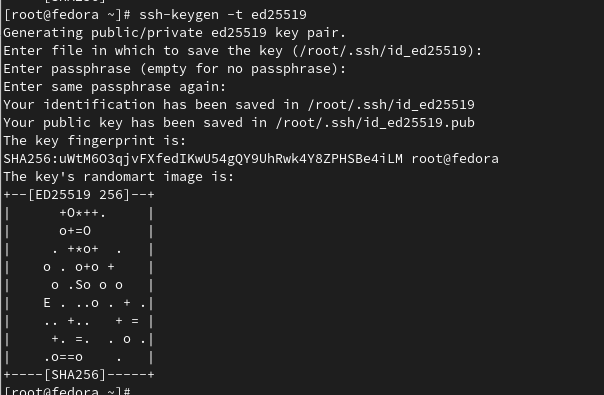


Figure 9: .

## 3.4 Создали ключи pgp

Сгенерировали ключ (рис. [[10](#fig:010)])

Из предложенных опций выбирали: тип RSA and RSA; размер 4096; выберали срок действия; значение по умолчанию — 0 (срок действия не истекает никогда). GPG запросил личную информацию, которая сохранится в ключе: Имя. Адрес электронной почты. При вводе email убедились, что он соответствует адресу, используемому на GitHub. (рис. [[11](#fig:011)])

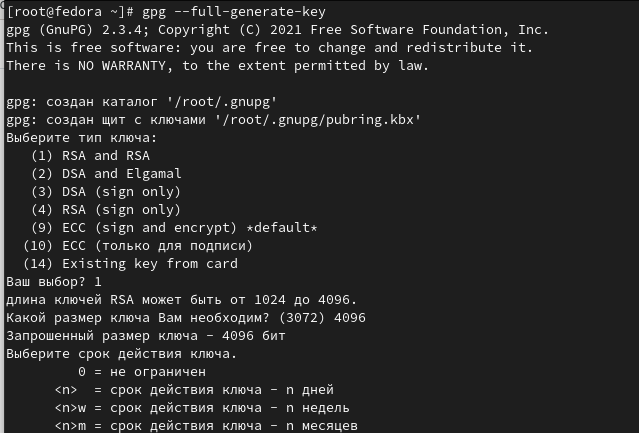


Figure 10: .

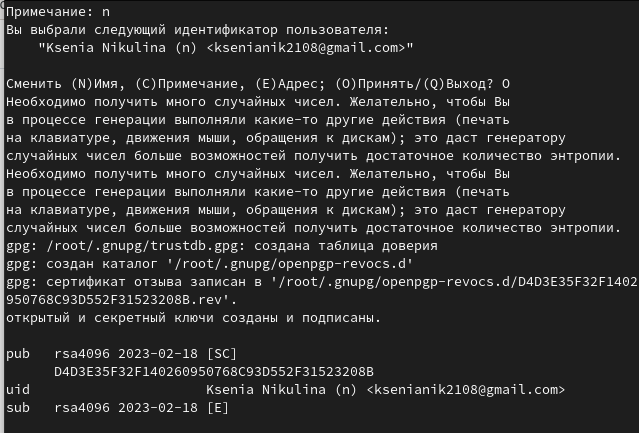


Figure 11: .

## 3.5 Настройка github

Создайте учётную запись на github.com. (рис. [[12](#fig:012)])

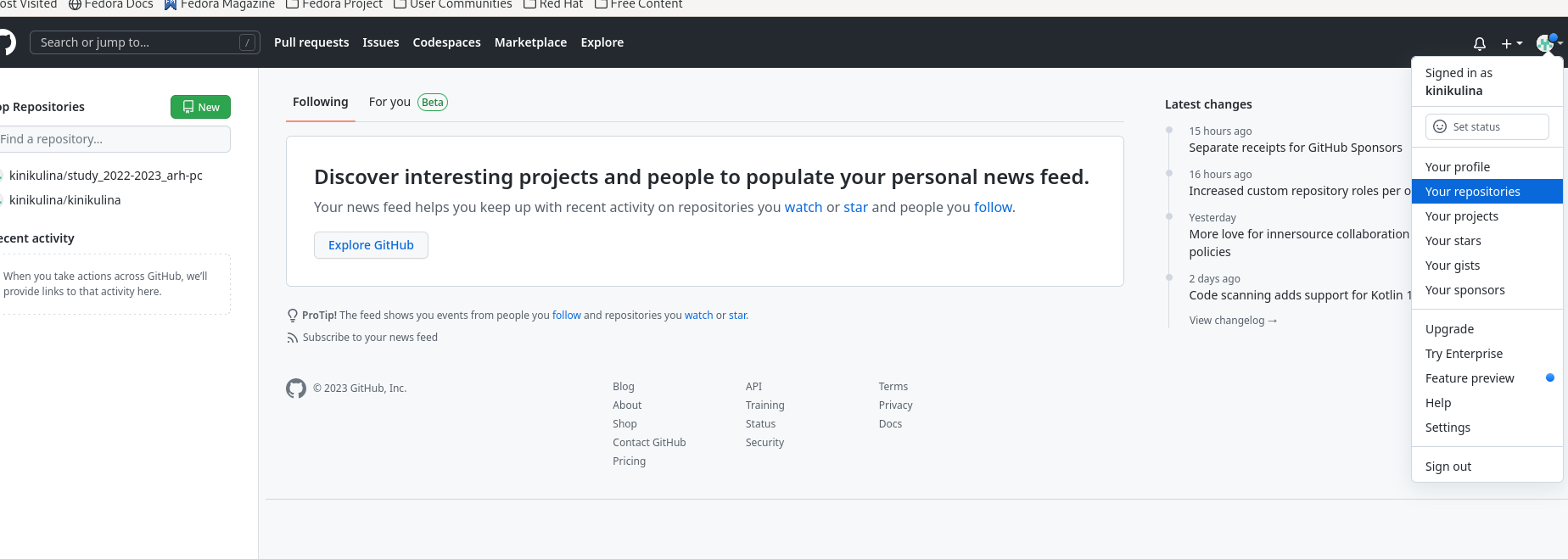


Figure 12: .

Заполните основные данные на github.com. (рис. [[13](#fig:013)])

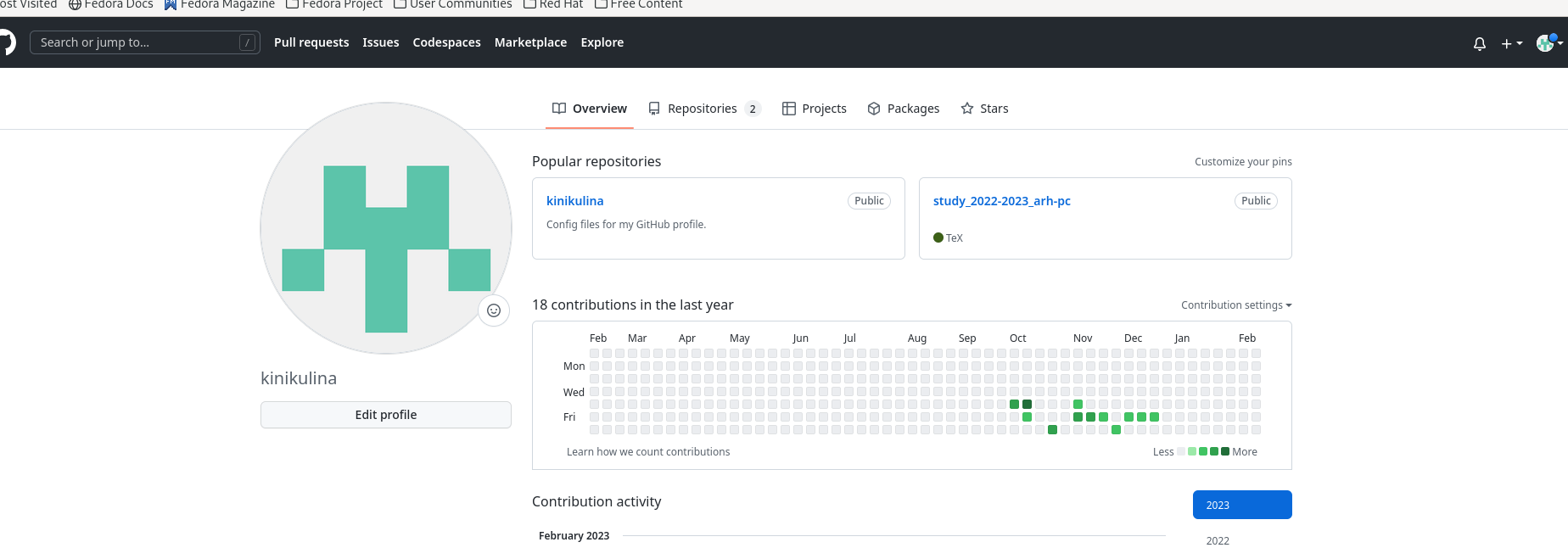


Figure 13: .

## 3.6 Добавление PGP ключа в GitHub

Вывели список ключей и копировали отпечаток приватного ключа: (рис. [[14](#fig:014)]) Отпечаток ключа — это последовательность байтов, используемая для идентификации более длинного, по сравнению с самим отпечатком ключа.

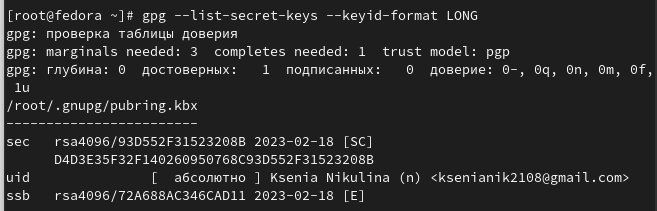


Figure 14: .

Cкопировали сгенерированный PGP ключ в буфер обмена: (рис. [[15](#fig:015)])

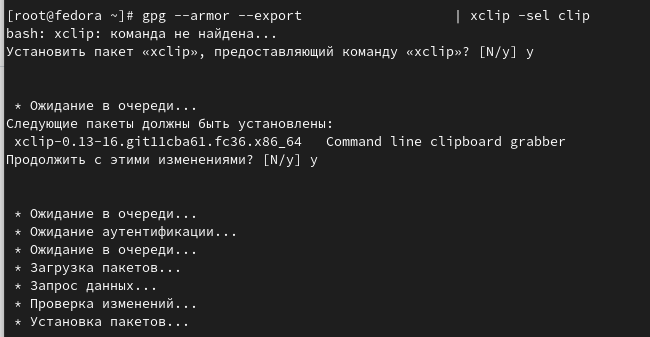


Figure 15: .

Перешли в настройки GitHub, нажали на кнопку New GPG key и вставили полученный ключ в поле ввода. (рис. [[16](#fig:016)], [[17](#fig:017)])

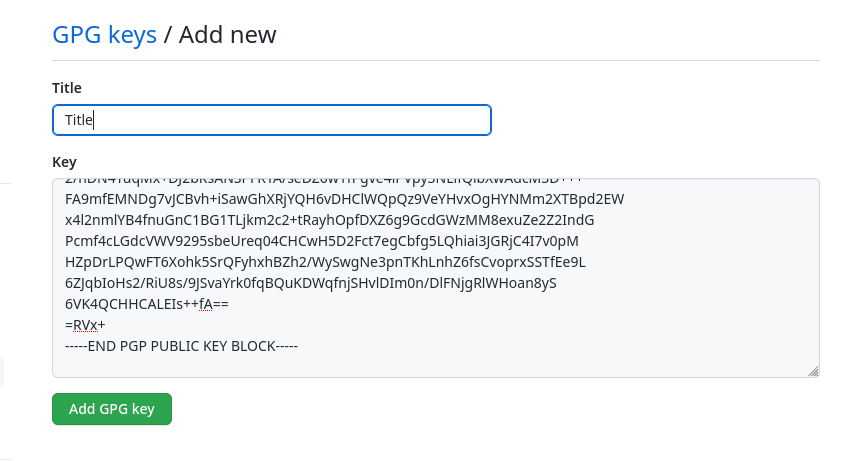


Figure 16: .

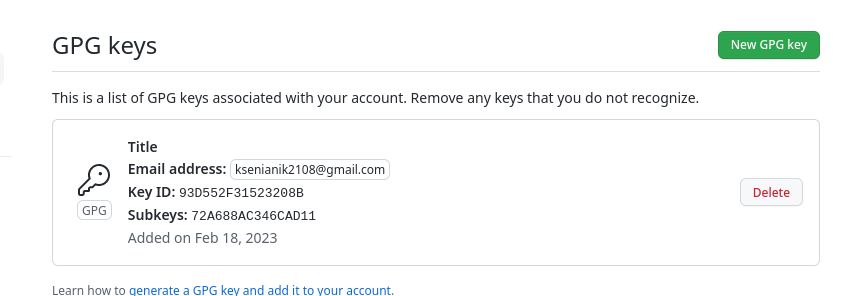


Figure 17: .

## 3.7 Настройка автоматических подписей коммитов git

Используя введёный email, указали Git применять его при подписи коммитов: (рис. [[18](#fig:018)])

Figure 18: .

Figure 18: .

## 3.8 Настройка gh

Авторизовались в gh. (рис. [[19](#fig:019)]) Утилита задали несколько наводящих вопросов.

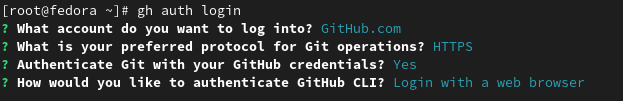


Figure 19: .

## 3.9 Сознание репозитория курса на основе шаблона

Создали шаблон рабочего пространства. (рис. [[20](#fig:020)], [[21](#fig:021)], [[22](#fig:022)])

Figure 20: .

Figure 20: .

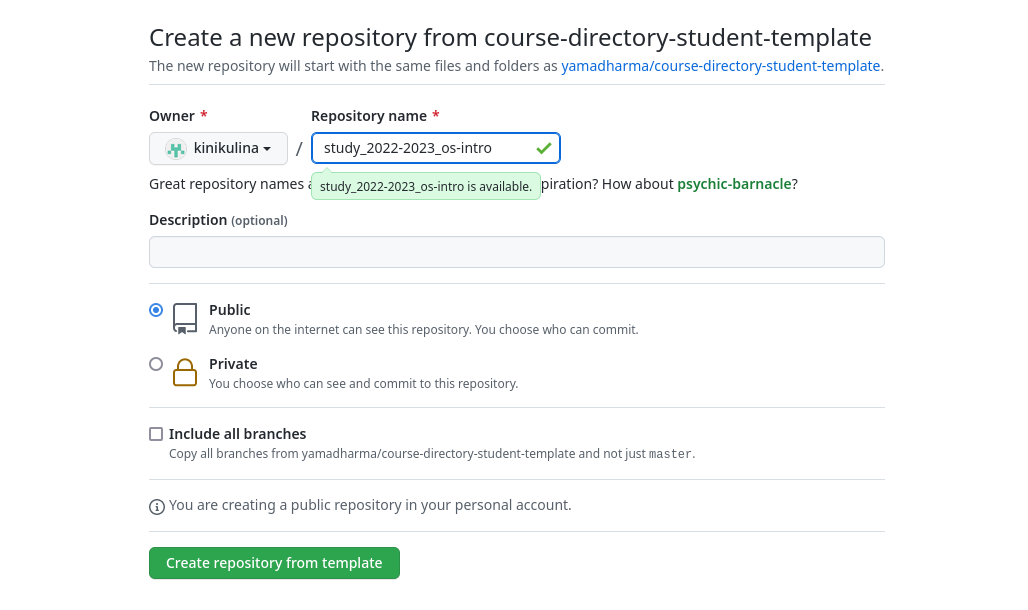


Figure 21: .

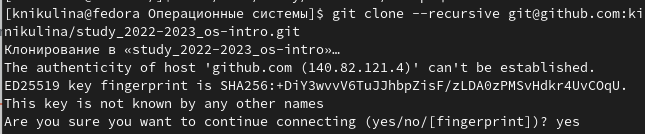


Figure 22: .

## 3.10 Настройка каталога курса

Перешли в каталог курса: (рис. [[23](#fig:023)])

Figure 23: .

Figure 23: .

Удалили лишние файлы: (рис. [[24](#fig:024)])

Figure 24: .

Figure 24: .

Создали необходимые каталоги: (рис. [[25](#fig:025)])

Figure 25: .

Figure 25: .

Отправили файлы на сервер: (рис. [[26](#fig:026)], [[27](#fig:027)])

Figure 26: .

Figure 26: .

Figure 27: .

Figure 27: .

# 4 Выводы

В ходе выполнения данной лабораторной работы была изучена идеология и применение средств контроля версий и освоены умения по работе с git.

# 5 Ответы на контрольные вопросы

1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются? Система управления версиями (также используется определение «система контроля версий», от англ. Version Control System, VCS или Revision Control System) — 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.
2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия. Хранилище (repository), или репозитарий, — место хранения файлов и их версий, служебной информации. Версия (revision), или ревизия, — состояние всего хранилища или отдельных файлов в момент времени («пункт истории»). Commit («трудовой вклад», не переводится) — процесс создания новой версии; иногда синоним версии. 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. Децентрализованные VCS: У каждого пользователя свой вариант (возможно не один) репозитория Присутствует возможность добавлять и забирать изменения из любого репозитория ( Git, Mercurial,Bazaar)

Централизованные VCS : Одно основное хранилище всего проекта Каждый пользователь копирует себе необходимые ему файлы из этого репозитория, изменяет и, затем, добавляет свои изменения обратно (Subversion, CVS, TFS, VAULT, AccuRev)

1. Опишите действия с VCS при единоличной работе с хранилищем. (рис. [[28](#fig:028)])

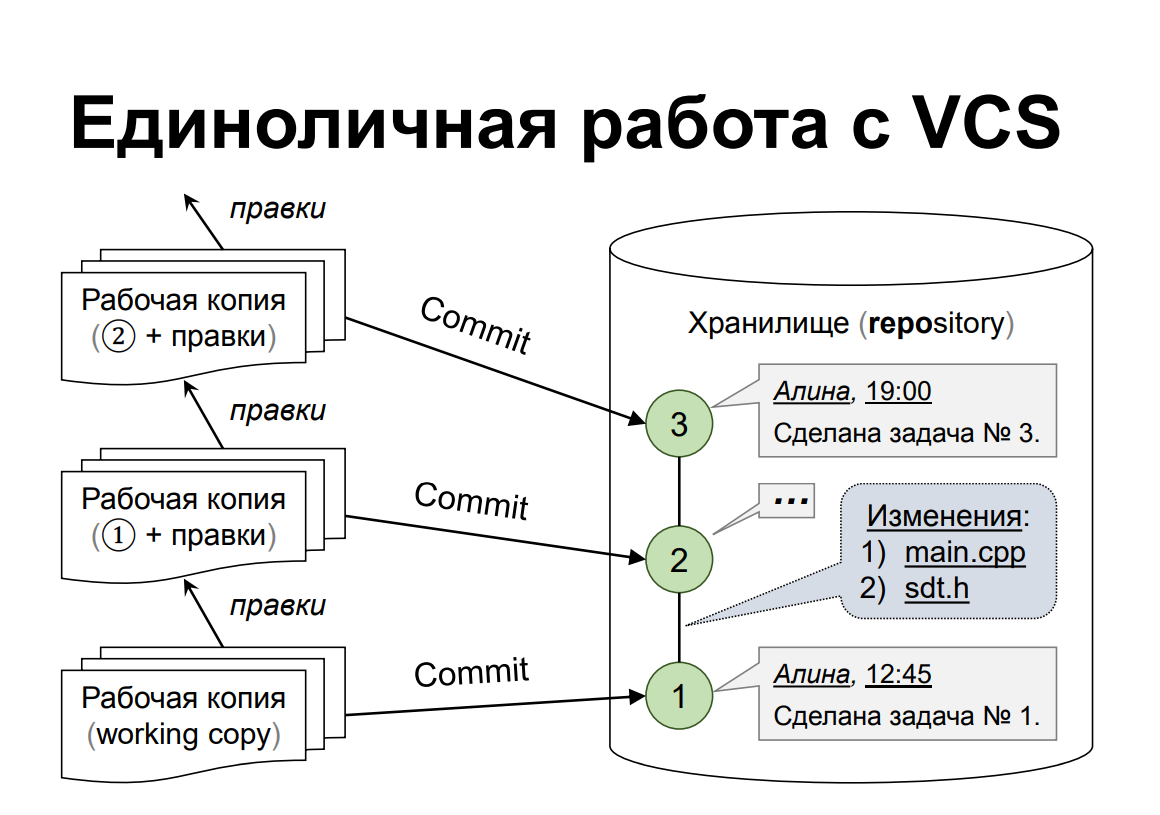


Figure 28: .

1. Опишите порядок работы с общим хранилищем VCS. (рис. [[29](#fig:029)])

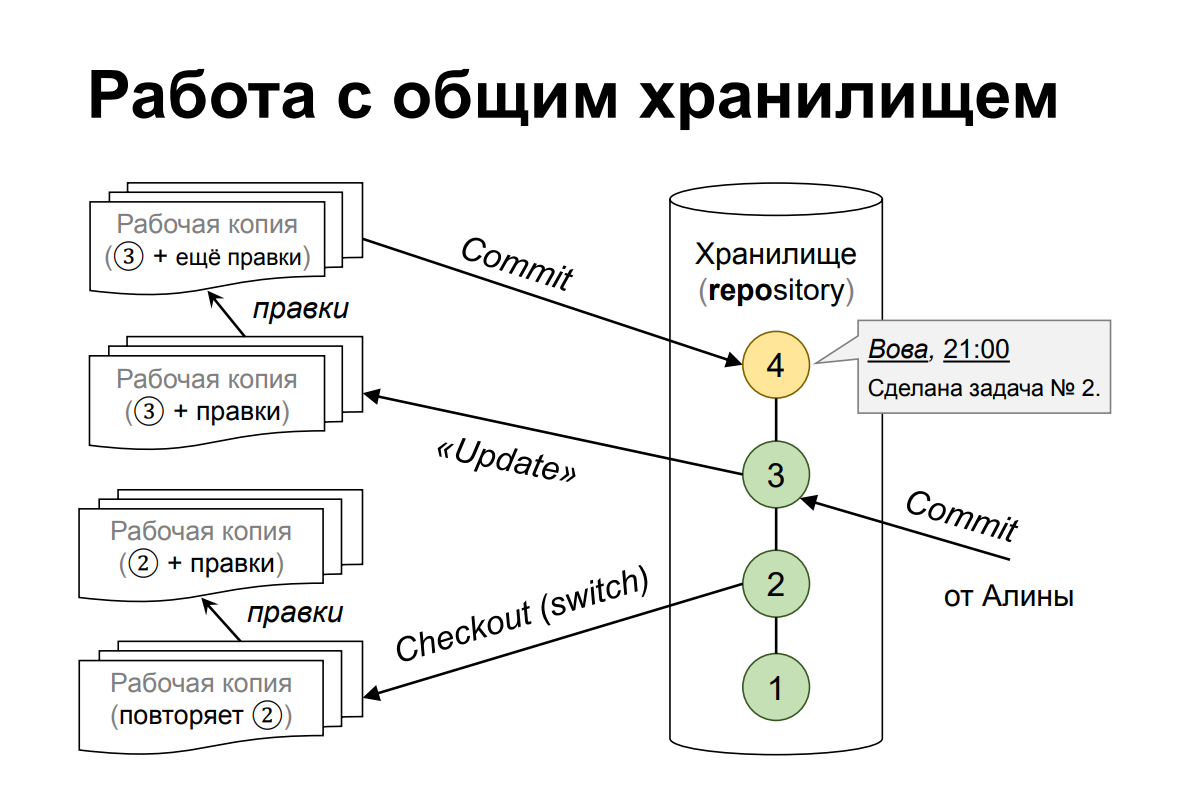


Figure 29: .

1. Каковы основные задачи, решаемые инструментальным средством git? У Git две основных задачи: первая — хранить информацию о всех изменениях в вашем коде, начиная с самой первой строчки, а вторая — обеспечение удобства командной работы над кодом.
2. Назовите и дайте краткую характеристику командам git. git init - создание репозитория git add (имена файлов) - Добавляет файлы в индекс git commit – выполняет коммит проиндексированных файлов в репозиторий git status – показывает какие файлы изменились между текущей стадией и HEAD. Файлы разделяются на 3 категории: новые файлы, измененные файлы, добавленные новые файлы git checkout (sha1 или метка) - получение указанной версии файла git push – отправка изменений в удаленный репозиторий git fetch – получение изменений из удаленного репозитория git clone (remote url) - клонирование удаленного репозитория себе
3. Приведите примеры использования при работе с локальным и удалённым репозиториями. (рис. [[30](#fig:030)])

Figure 30: .

Figure 30: .

1. 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 репозитарии git branch. Создать ветку git branch имя.

Ветки нужны, чтобы несколько программистов могли вести работу над одним и тем же проектом или даже файлом одновременно, при этом не мешая друг другу. Кроме того, ветки используются для тестирования экспериментальных функций: чтобы не повредить основному проекту, создается новая ветка специально для экспериментов.

1. Как и зачем можно игнорировать некоторые файлы при commit? Игнорируемые файлы — это, как правило, артефакты сборки и файлы, генерируемые машиной из исходных файлов в вашем репозитории, либо файлы, которые по какой-либо иной причине не должны попадать в коммиты. Вот некоторые распространенные примеры таких файлов:

кэши зависимостей, например содержимое node\_modules или packages; скомпилированный код, например файлы .o, .pyc и .class ; каталоги для выходных данных сборки, например bin, out или target; файлы, сгенерированные во время выполнения, например .log, .lock или .tmp; скрытые системные файлы, например .DS\_Store или Thumbs.db; личные файлы конфигурации IDE, например .idea.workspace.xml.