Отчёт по лабораторной работе №2

Первоначальная настройка Git

Киньябаева Аиша Иделевна

Содержание

# 1 Цель работы

Целью данной работы является изучение контроля версий и создание репозитория на основе шаблона курса

# 2 Задание

Создать репозиторий на Git

# 3 Выполнение лабораторной работы

Регистрация на Github и заполнение основных данных репозитория (рис. [1](#fig:001)), (рис. [2](#fig:002))

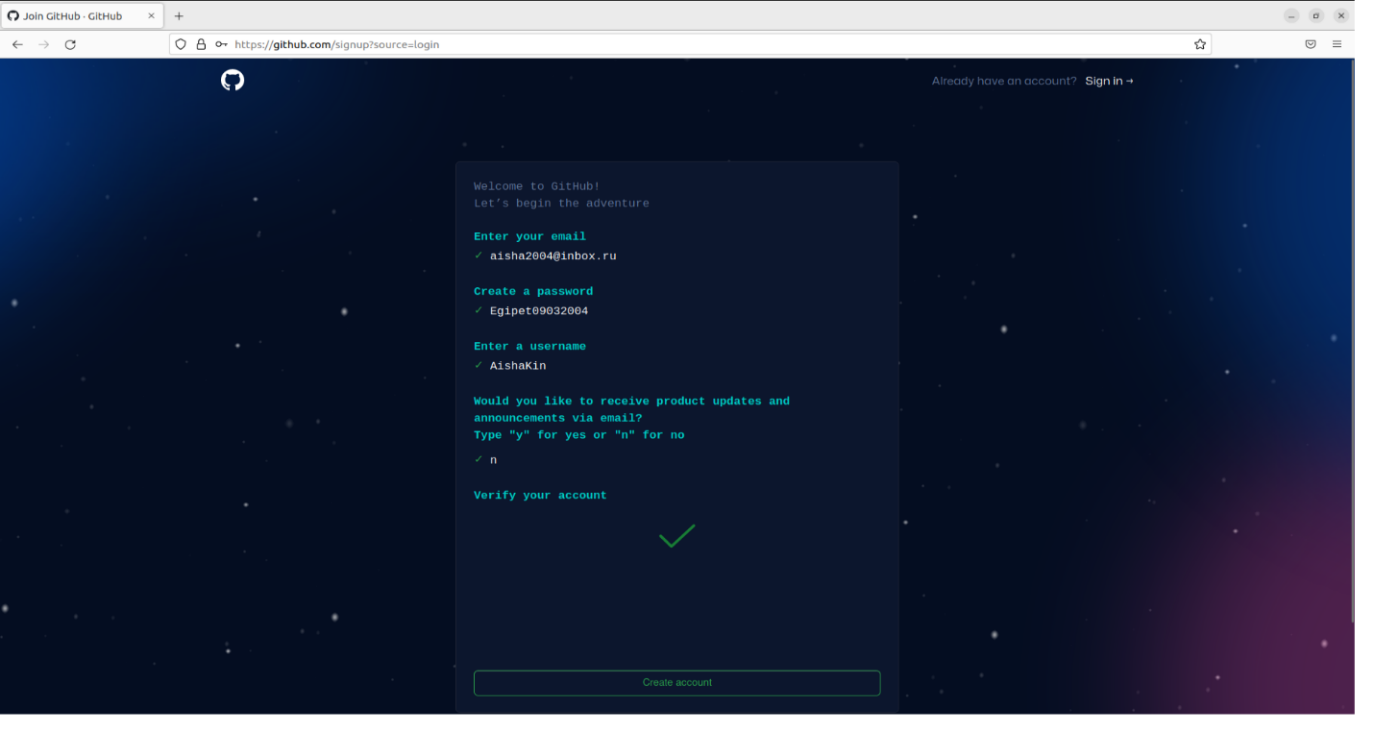


Figure 1: регистрация

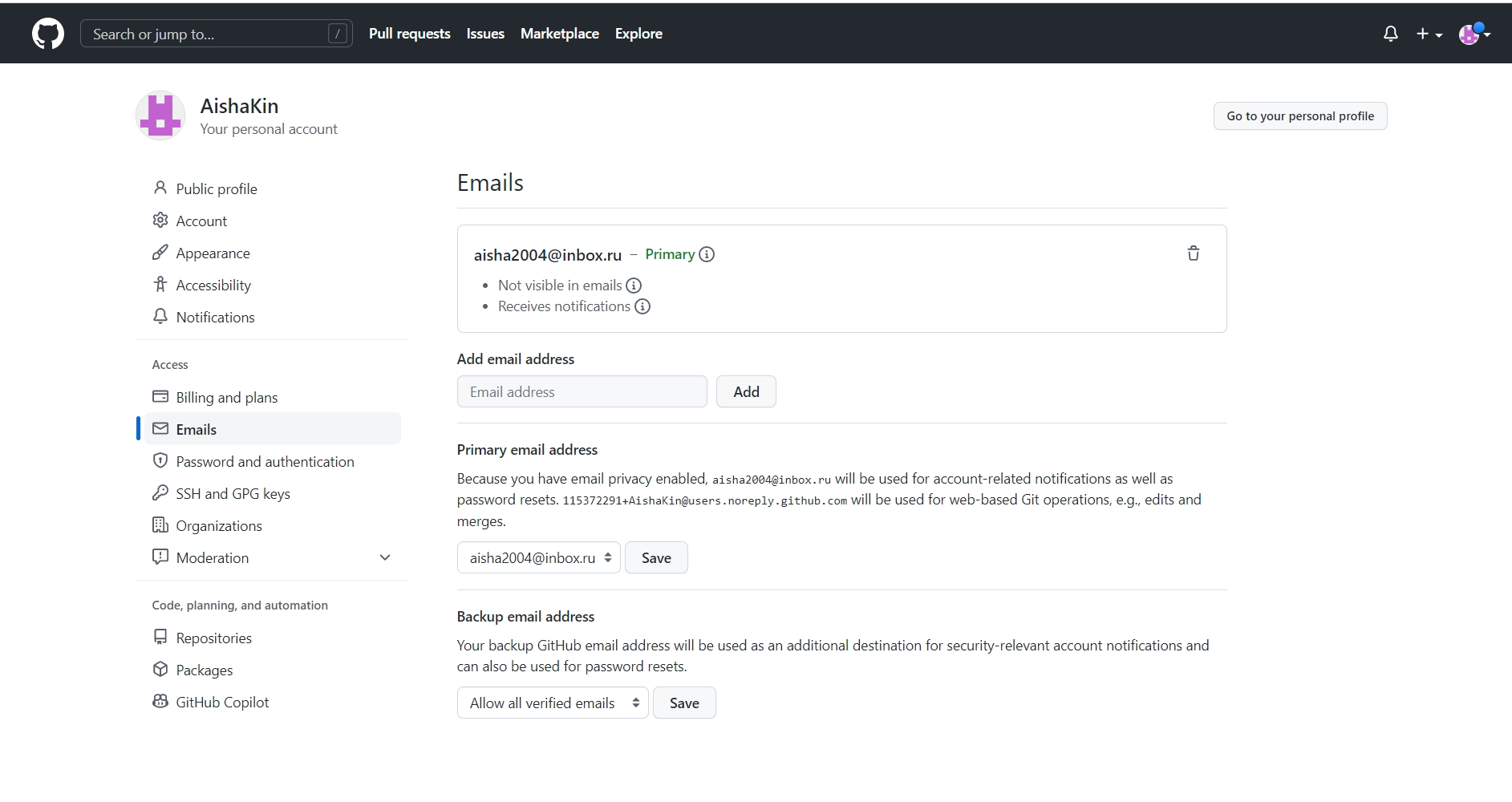


Figure 2: Заполнение данных

Базовая настройка Git с использованием имени пользователя и электронной почты (рис. [3](#fig:003) )

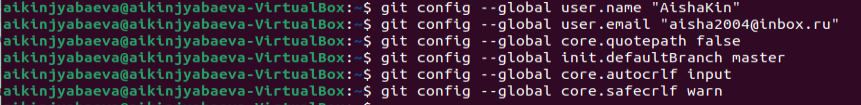


Figure 3: Базовая настройка

Создание SSH ключа и копирование его в репозиторий (рис. [4](#fig:004) ), (рис. [5](#fig:005) )

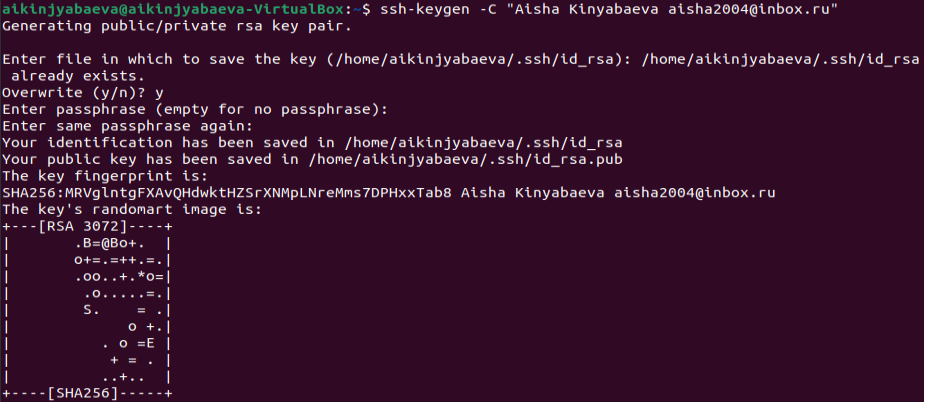


Figure 4: Генерация ключа

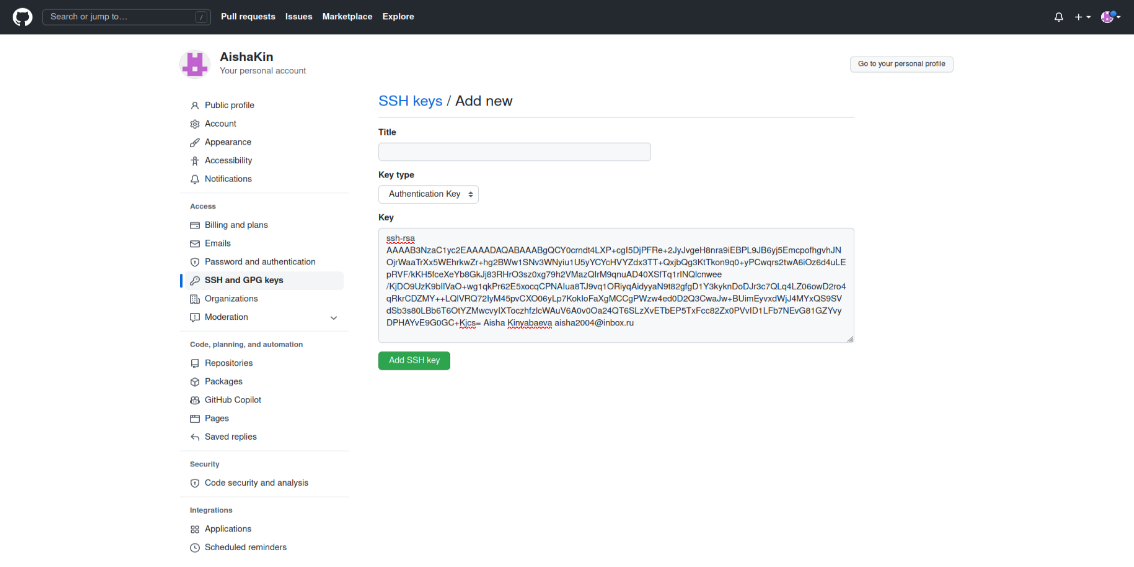


Figure 5: Копирование ключа

Далее настраиваю параметры autocrlf и safecrlf. А также генерирую ключ gpg и копирую его на git (рис. [6](#fig:006) )

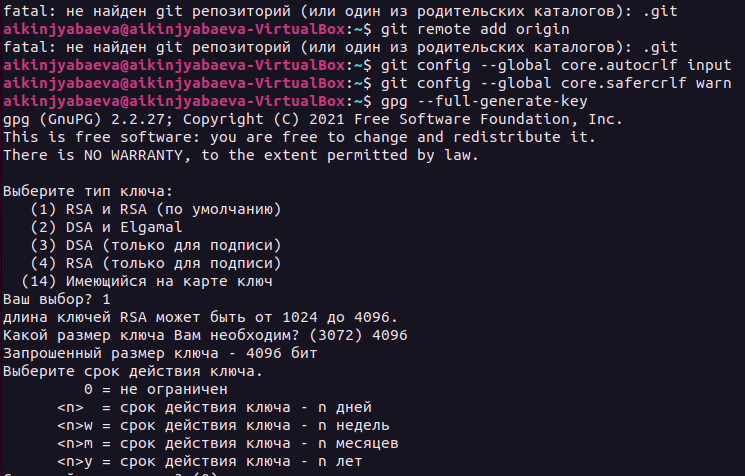


Figure 6: Генерация ключа

Настройка автоматических подписей коммитов git (рис. [7](#fig:007) )

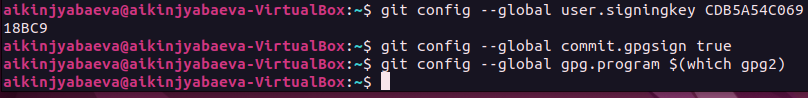


Figure 7: Настройка подписей

Копирование шаблона курса (рис. [8](#fig:008) )

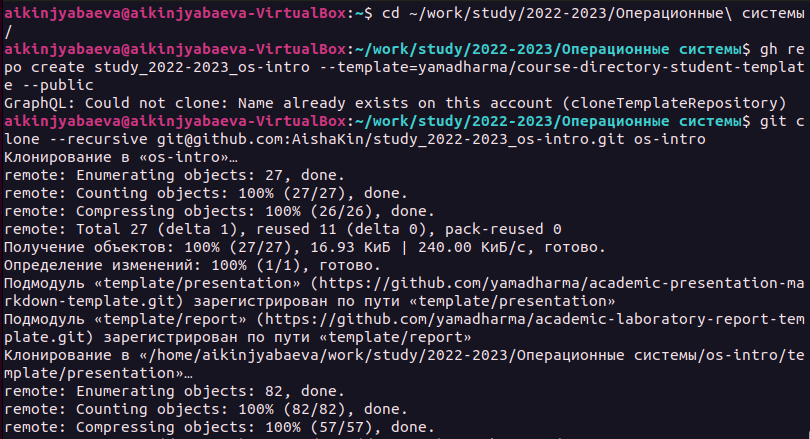


Figure 8: Копирование шаблона

Настройка каталога курса и отправка файлов на репозиторий (рис. [9](#fig:009) ), (рис. [10](#fig:010) )

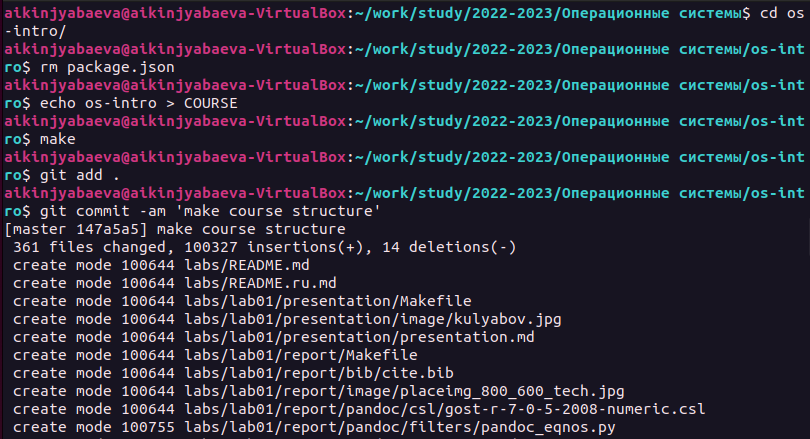


Figure 9: Настройка каталога

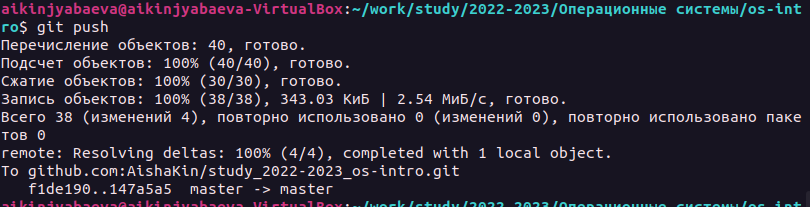


Figure 10: Отправка файлов

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются? Системы контроля версий необходимы при работе нескольких человек над одним проектом. Прдназначены для решения таких задач как:
   * внесение изменений в проект разными участниками
   * совмещение изменений
   * возврат к более ранним версиям проекта
   * сохранение изменений
2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.
   * Репозиторий - место хранения файловой ситсемы каждого отдельного проекта. Существуют локальные и удаленные репозитории, где локальные для работы над проектом на компьютере, а удаленные в качестве хранилища
   * Commit - Операция, позволяющая сохранить текущее состояние проекта и добавить его в репозиторий
   * История - способность системы контроля версий сохранять и работать с несколькими версиями одного файла и его ветвлений
   * Рабочая копия - является снимком одной версии проекта. Эти файлы извлекаются из сжатой базы данных в каталоге Git и помещаются на диск, для того чтобы их можно было использовать или редактировать.
3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида.
   * Централизованные системы контроля версий предполагают сохранение версий проектов на общий сервер, с которого потом получают нужные версии клиенты. Примеры: CVS, Subversion, Perforce
   * В децентрализованных системах контроля версий при каждом копировании удалённого репозитория (расположенного на сервере) происходит полное копирование данных в локальный репозиторий (установленный на рабочем компьютере). Каждая копия содержит все данные, хранящиеся в удалённом репозитории. В случае, возникновения технической неисправности на стороне сервера, удаленный репозиторий можно перезаписать с любой сохраненной копии. Пример: Mercurial, Bazaar
4. Опишите действия с VCS при единоличной работе с хранилищем.
   * Создадим локальный репозиторий.
   * Сделаем предварительную конфигурацию, указав имя и email владельца репозитория
   * Инициализация локального репозитория, расположенного, например, в каталоге ~/tutorial
   * После это в каталоге tutorial появится каталог .git, в котором будет храниться история изменений.
   * Воспользуемся командой status для просмотра изменений в рабочем каталоге, сделанных с момента последней ревизии:git status
   * 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. Для этого сначала нужно получить список имеющихся шаблонов
5. Каковы основные задачи, решаемые инструментальным средством git? Git не только позволяет сохранять контрольные точки проекта, но и помогает устранять конфликты. Часто бывает так, что программисты одновременно работают над одной функцией и заливают изменения в репозиторий. В этом случае система обнаруживает конфликт и пытается исправить его автоматически.
6. Назовите и дайте краткую характеристику командам 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
   * Добавить все изменённые и/или созданные файлы и/или каталоги: git add .
   * Удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории): git rm имена\_файлов
   * Сохранить все добавленные изменения и все изменённые файлы: git commit -am ‘Описание коммита’
   * Сохранить добавленные изменения с внесением комментария через встроенный редактор: git commit
   * Отправка изменений конкретной ветки в центральный репозиторий: git push origin имя\_ветки
7. Что такое и зачем могут быть нужны ветви (branches)? Ветки нужны для того, чтобы программисты могли вести совместную работу над проектом и не мешать друг другу при этом. При создании проекта, Git создает базовую ветку. Она называется master веткой. Она считается центральной веткой, т.е. в ней содержится основной код приложения.
8. 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 ЫИгнорируемые файлы отслеживаются в специальном файле .gitignore, который регистрируется в корневом каталоге репозитория. В Git нет специальной команды для указания игнорируемых файлов: вместо этого необходимо вручную отредактировать файл .gitignore, чтобы указать в нем новые файлы, которые должны быть проигнорированы. Файлы .gitignore содержат шаблоны, которые сопоставляются с именами файлов в репозитории для определения необходимости игнорировать эти файлы

# 4 Выводы

В ходе данной лабораторной работы были изучены основные моменты создания репозитория на git, основные команды гит и непосредственно копирование файлов на репозиторий.