Отчет по лабораторной работе №2

Первоначальная настройка git

Ангелина Павловна Ким

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. Настроить подписи git.
5. Зарегистрироваться на Github.
6. Создать локальный каталог для выполнения заданий по предмету.

# 3 Выполнение лабораторной работы

Для начала установим git (рис. [1](#fig:001)).

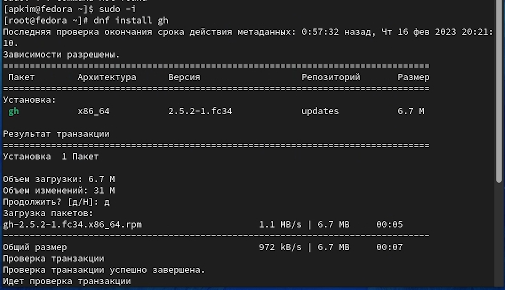


Figure 1: Установка git

Далее зададим имя и почту владельца репозитория, а затем настроим utf-8 в выводе сообщений git (рис. [2](#fig:002)).

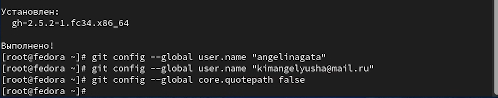


Figure 2: Имя пользователя и почта

Следующим шагом настроим верификацию и подписание коммитов git. Сначала зададим имя начальной ветки, потом параметр autocrlf, а затем параметр safecrlf (рис. [3](#fig:003)).

Figure 3: Настройка верификации и подписание коммитов git

Figure 3: Настройка верификации и подписание коммитов git

Далее нам нужно создать SSH ключи. По алгоритму rsa с ключем размером 4096 бит, по алгоритму ed25519 (рис. [4](#fig:004)).

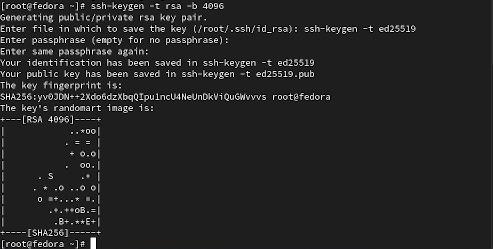


Figure 4: Создание SSH ключа

Теперь создаем ключи pgp. Сначала генерируем ключ. Затем из предложенных опций выбираем то, что нужно. (рис. [5](#fig:005)).

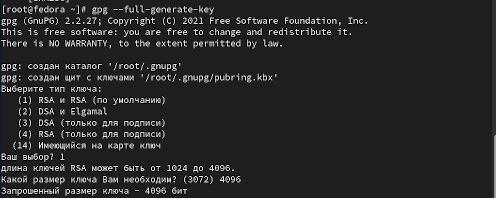


Figure 5: Генерируем ключ pgp

Продолжение опций (рис. [6](#fig:006)).

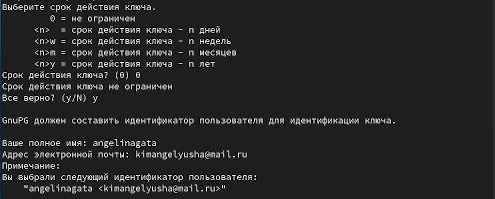


Figure 6: Опции

Далее выводим список ключей и копируем отпечаток приватного ключа. Затем копируем сгенерированный pgp ключ и вставляем его на github. (рис. [7](#fig:007)).

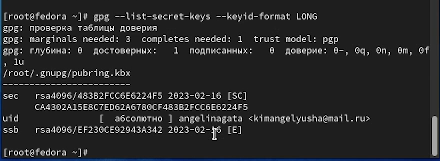


Figure 7: Копирование приватного ключа

Ключ присоединился (рис. [8](#fig:008)).

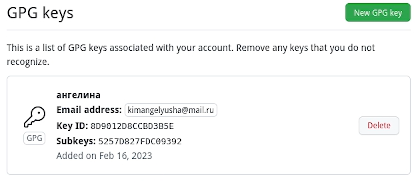


Figure 8: Готовый ключ

Следующим шагом нужно, используя введенную почту, указать Git и применять его при подписи коммитов (рис. [9](#fig:009)).



Figure 9: Подписи коммитов

Далее необходимо авторизоваться, отвечая на вопросы (рис. [10](#fig:0010)).

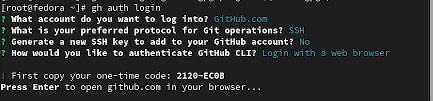


Figure 10: Авторизация

Успешное подсоединение (рис. [11](#fig:0011)).

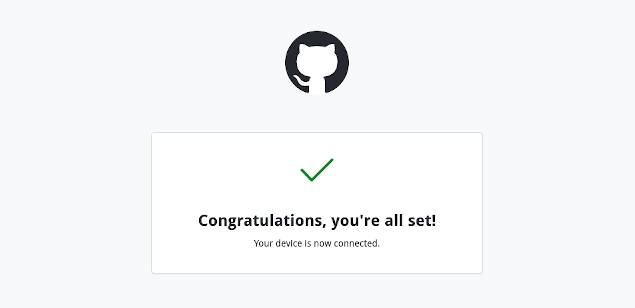


Figure 11: Подсоединение

Теперь нам нужно создать репозиторий на основе шаблона (рис. [12](#fig:0012)).

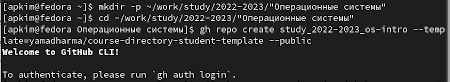


Figure 12: Создание репозитория (1)

Продолжение (рис. [13](#fig:0013)).

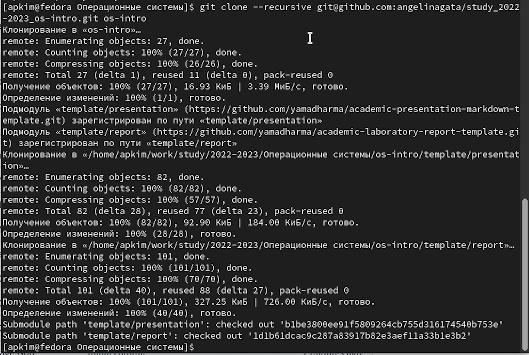


Figure 13: Создание репозитория (2)

Теперь нам нужно настроить каталог курса. Сначала переходим в каталог курса, затем удаляем лишние файлы и создаем необходимые каталоги (рис. [14](#fig:0014)).

Figure 14: Настройка каталога курса

Figure 14: Настройка каталога курса

Отправляем файлы на сервер (рис. [15](#fig:0015)).

Figure 15: Отправка файлов (1)

Figure 15: Отправка файлов (1)

Продолжение (рис. [16](#fig:0016)).

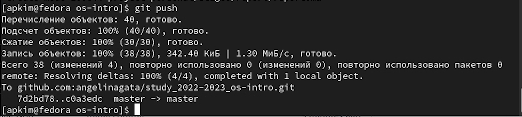


Figure 16: Отправка файлов (2)

Ответы на контрольные вопросы: 1. Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом. Обычно основное дерево проекта хранится в локальном или удалённом репозитории, к которому настроен доступ для участников проекта. При внесении изменений в содержание проекта система контроля версий позволяет их фиксировать, совмещать изменения, произведённые разными участниками проекта, производить откат к любой более ранней версии проекта, если это требуется. Системы контроля версий поддерживают возможность отслеживания и разрешения конфликтов, которые могут возникнуть при работе нескольких человек над одним файлом. Системы контроля версий также могут обеспечивать дополнительные, более гибкие функциональные возможности. Например, они могут поддерживать работу с несколькими версиями одного файла, сохраняя общую историю изменений до точки ветвления версий и собственные истории изменений каждой ветви. 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. 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. Сервер может сохранять не полную версию изменённых файлов, а производить так называемую дельта-компрессию — сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных. Сохранить добавленные изменения с внесением комментария через встроенный редактор: git commit 3. Централизованные системы контроля версий представляют собой приложения типа клиент-сервер, когда репозиторий проекта существует в единственном экземпляре и хранится на сервере. Доступ к нему осуществлялся через специальное клиентское приложение. В качестве примеров таких программных продуктов можно привести CVS, Subversion. Распределенные системы контроля версий (Distributed Version Control System, DVCS) позволяют хранить репозиторий (его копию) у каждого разработчика, работающего с данной системой. При этом можно выделить центральный репозиторий (условно), в который будут отправляться изменения из локальных и, с ним же эти локальные репозитории будут синхронизироваться. При работе с такой системой, пользователи периодически синхронизируют свои локальные репозитории с центральным и работают непосредственно со своей локальной копией. После внесения достаточного количества изменений в локальную копию они (изменения) отправляются на сервер. При этом сервер, чаще всего, выбирается условно, т.к. в большинстве DVCS нет такого понятия как “выделенный сервер с центральным репозиторием”. 4. 5. 6. 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. 7. Основные команды git Перечислим 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 изменений:

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. 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.

# 4 Выводы

В ходе данной лабораторной работы мы изучили идеологию и применение средств контроля версий, а также освоили умения по работе с git.

# Список литературы