МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

## Факультет информационных технологий и робототехники

Кафедра программного обеспечения информационных систем и технологий

**Отчет по лабораторной работе № 1**

по дисциплине: ”Конструирование программного обеспечения”

на тему: ”***Git”***

Выполнили**:** студенты группы 10701321

Машкарёва М.А., Радионов Д.В.

Принял**:** Станкевич С.Н.

Минск 2023

# Лабораторная работа № 1. Git

**Цель работы: ознакомится с системой контроля версий – Git.**

История: Разработка ядра Linux велась на проприетарной системе BitKeeper, которую автор — Ларри Маквой, сам разработчик Linux — предоставил проекту по бесплатной лицензии. Разработчики, высококлассные программисты, написали несколько утилит, и для одной Эндрю Триджелл произвёл реверс-инжиниринг формата передачи данных BitKeeper. В ответ Маквой обвинил разработчиков в нарушении соглашения и отозвал лицензию, и Торвальдс взялся за новую систему: ни одна из открытых систем не позволяла тысячам программистов кооперировать свои усилия (тот же конфликт привёл к написанию Mercurial). Идеология была проста: взять подход CVS и перевернуть с ног на голову, и заодно добавить надёжности.

Начальная разработка велась меньше, чем неделю: 3 апреля 2005 года разработка началась, и уже 7 апреля код Git управлялся неготовой системой. 16 июня Linux был переведён на Git, а 25 июля Торвальдс отказался от обязанностей ведущего разработчика.

Основные данные: Git — распределённая система управления версиями. Проект был создан Линусом Торвальдсом для управления разработкой ядра Linux, первая версия выпущена 7 апреля 2005 года. На сегодняшний день его поддерживает Джунио Хамано.

Разработчик: Software Freedom Conservancy

Лицензия: GNU GPL 2

Написана на: Си, командная оболочка UNIX, Perl, Tcl, Python и C++

Операционная система: кроссплатформенность

Репозиторий: git.kernel.org/pub/scm/g…

Git применяется для управления версиями в рамках колоссального количества проектов по разработке ПО, как коммерческих, так и с открытым исходным кодом.

Основные команды Git:

Команда git add добавляет содержимое рабочей директории в индекс (staging area) для последующего коммита. По умолчанию git commit использует лишь этот индекс, так что вы можете использовать git add для сборки слепка вашего следующего коммита.

Команда git status показывает состояния файлов в рабочей директории и индексе: какие файлы изменены, но не добавлены в индекс; какие ожидают коммита в индексе. Вдобавок к этому выводятся подсказки о том, как изменить состояние файлов.

Команда git diff используется для вычисления разницы между любыми двумя Git деревьями. Это может быть разница между вашей рабочей директорией и индексом (собственно git diff), разница между индексом и последним коммитом (git diff --staged), или между любыми двумя коммитами (git diff master branchB).

Команда git difftool просто запускает внешнюю утилиту сравнения для показа различий в двух деревьях, на случай если вы хотите использовать что-либо отличное от встроенного просмотрщика git diff.

Команда git commit берёт все данные, добавленные в индекс с помощью git add, и сохраняет их слепок во внутренней базе данных, а затем сдвигает указатель текущей ветки на этот слепок.

Команда git reset, как можно догадаться из названия, используется в основном для отмены изменений. Она изменяет указатель HEAD и, опционально, состояние индекса. Также эта команда может изменить файлы в рабочей директории при использовании параметра --hard, что может привести к потере наработок при неправильном использовании, так что убедитесь в серьёзности своих намерений прежде чем использовать его.

Команда git rm используется в Git для удаления файлов из индекса и рабочей директории. Она похожа на git add с тем лишь исключением, что она удаляет, а не добавляет файлы для следующего коммита.

Команда git mv — это всего лишь удобный способ переместить файл, а затем выполнить git addдля нового файла и git rm для старого.

Команда git clean используется для удаления мусора из рабочей директории. Это могут быть результаты сборки проекта или файлы конфликтов слияний.

Команды ветвления и слияния

Команда git branch — это своего рода “менеджер веток”. Она умеет перечислять ваши ветки, создавать новые, удалять и переименовывать их.

Команда git checkout используется для переключения веток и выгрузки их содержимого в рабочую директорию.

Команда git merge используется для слияния одной или нескольких веток в текущую. Затем она устанавливает указатель текущей ветки на результирующий коммит.

Команда git mergetool просто вызывает внешнюю программу слияний, в случае если у вас возникли проблемы слияния.

Команда git log используется для просмотра истории коммитов, начиная с самого свежего и уходя к истокам проекта. По умолчанию, она показывает лишь историю текущей ветки, но может быть настроена на вывод истории других, даже нескольких сразу, веток. Также её можно использовать для просмотра различий между ветками на уровне коммитов.

Команда git stash используется для временного сохранения всех незакоммиченных изменений для очистки рабочей директории без необходимости коммитить незавершённую работу в новую ветку.

Команда git tag используется для задания постоянной метки на какой-либо момент в истории проекта. Обычно она используется для релизов.

Команды совместной работы и обновлению проектов

Не так уж много команд в Git требуют сетевого подключения для своей работы, практически все команды оперируют с локальной копией проекта. Когда вы готовы поделиться своими наработками, всего несколько команд помогут вам работать с удалёнными репозиториями.

Команда git fetch связывается с удалённым репозиторием и забирает из него все изменения, которых у вас пока нет и сохраняет их локально.

Команда git pull работает как комбинация команд git fetch и git merge, т.е. Git вначале забирает изменения из указанного удалённого репозитория, а затем пытается слить их с текущей веткой.

Команда git push используется для установления связи с удалённым репозиторием, вычисления локальных изменений отсутствующих в нём, и собственно их передачи в вышеупомянутый репозиторий. Этой команде нужно право на запись в репозиторий, поэтому она использует аутентификацию.

Команда git remote служит для управления списком удалённых репозиториев. Она позволяет сохранять длинные URL репозиториев в виде понятных коротких строк, например "origin", так что вам не придётся забивать голову всякой ерундой и набирать её каждый раз для связи с сервером. Вы можете использовать несколько удалённых репозиториев для работы и git remote поможет добавлять, изменять и удалять их.

Команда git archive используется для упаковки в архив указанных коммитов или всего репозитория.

Команда git submodule используется для управления вложенными репозиториями. Например, это могут быть библиотеки или другие, используемые не только в этом проекте ресурсы. У команды submodule есть несколько под-команд — add, update, sync и др. — для управления такими репозиториями.

Команды по осмотру и сравнению

Команда git show отображает объект в простом и человекопонятном виде. Обычно она используется для просмотра информации о метке или коммите.

Команда git shortlog служит для подведения итогов команды git log. Она принимает практически те же параметры, что и git log, но вместо простого листинга всех коммитов, они будут сгруппированы по автору.

Команда git describe принимает на вход что угодно, что можно трактовать как коммит (ветку, тег) и выводит более-менее человекочитаемую строку, которая не изменится в будущем для данного коммита. Это может быть использовано как более удобная, но по-прежнему уникальная, замена SHA-1.

Команды по отладке

В Git есть несколько команд, используемых для нахождения проблем в коде. Это команды для поиска места в истории, где проблема впервые проявилась и собственно виновника этой проблемы.

Команда git bisect — это чрезвычайно полезная утилита для поиска коммита в котором впервые проявился баг или проблема с помощью автоматического бинарного поиска.

Команда git blame выводит перед каждой строкой файла SHA-1 коммита, последний раз менявшего эту строку и автора этого коммита. Это помогает в поисках человека, которому нужно задавать вопросы о проблемном куске кода.

Команда git grep используется для поиска любой строки или регулярного выражения в любом из файлов вашего проекта, даже в более ранних его версиях.

GitHub — веб-сервис, который основан на системе Git. Это такая социальная сеть для разработчиков, которая помогает удобно вести коллективную разработку IT-проектов. Здесь можно публиковать и редактировать свой код, комментировать чужие наработки, следить за новостями других пользователей.

Настройка Git

1. Установите git и сгенерируйте пару SSH-ключей. Авторизуйте публичный ключ на github.com

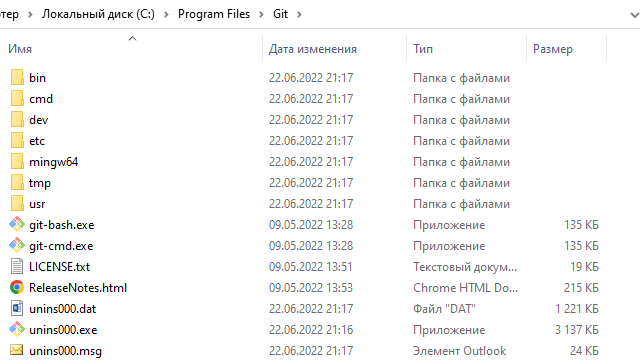


Рисунок 1 – Git установлен

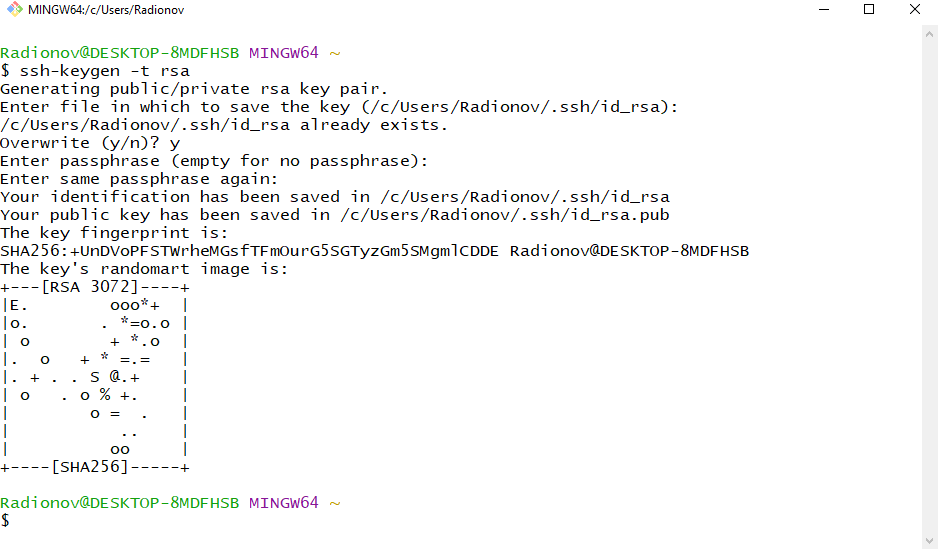


Рисунок 2 – Генерация SSH-ключей

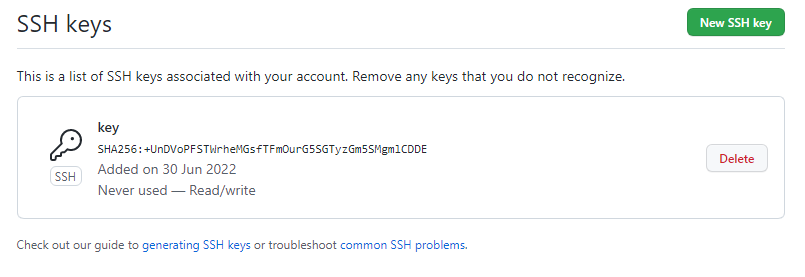


Рисунок 3 – Авторизация публичного ключа

2. Укажите свой user.name и user.email в git

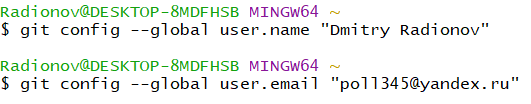


Рисунок 4 – Указание user.name и user.email

3. Создайте новый репозиторий на github.com и склонируйте его локально на свой компьютер

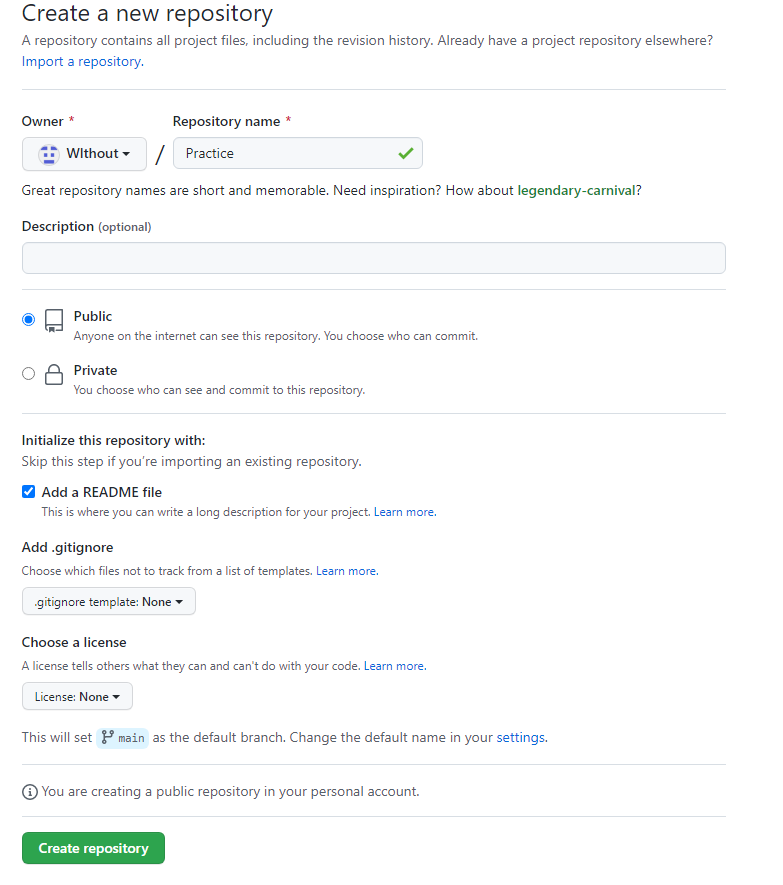


Рисунок 5 – Создание нового репозитория на github.com

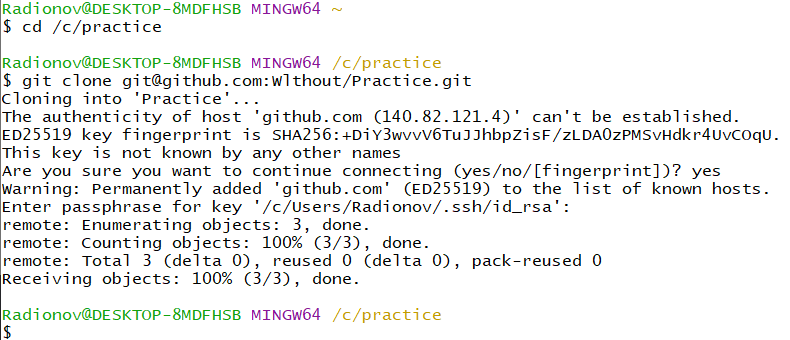


Рисунок 6 – Клонирование локального репозитория на свой компьютер

Ссылка на GitHub - https://github.com/Wlthout

Вывод*:* в ходе выполнения лабораторной работы ознакомились с системой контроля версий – Git.