Отчет по лабораторной работе №1: Система контроля версий Git

*дисциплина: Математическое моделирование*

Сасин Ярослав Игоревич, НФИбд-03-18

Содержание

# Введение

## Цель работы

Основной целью лабораторной работы можно считать ознакомление с системой контроля версий Git.

## Задачи работы

Можно выделить три основные задачи данной лабораторной работы: 1. ознакомление с системой контроля версий Git; 2. создание каталогов для хранения данных последующих лабораторнвх работ с их выгрузкой на Github; 3. ознакомление с набором расширений git-flow.

## Объект и предмет исследования

Объектом исследований для данной лабораторной работы является система котроля версий git, предметом же исследования можно считать непосредственное взаимодействие с git, его преимущества и особенности, а также использование платформы GitHub для размещение git-репозиториев.

# Терминология. Условные обозначения

**git** - это бесплатная распределенная система контроля версий с открытым исходным кодом, предназначенная для быстрого и эффективного управления всеми проектами любого размера.

[**GitHub**](https://github.com) - крупнейший веб-сервис для хостинга IT-проектов и их совместной разработки. Веб-сервис основан на системе контроля версий Git и разработан на Ruby on Rails и Erlang компанией GitHub, Inc.

**git-flow** — это набор расширений git предоставляющий высокоуровневые операции над репозиторием для поддержки модели ветвления Vincent Driessen.

[**Homebrew**](https://brew.sh) — утилита командной строки в macOS и Linux, которая позволяет устанавливать пакеты и приложения (менеджер пакетов). Распространяется как свободное программное обеспечение с открытым кодом.

# Выполнение лабораторной работы

Для того, чтобы использовать git на своем устройстве, для начала необходимо его установить. Так как в моем случае git уже был установлен, данный пункт работы был пропущен. Но, в случае, если бы на моем устройстве не был бы установлен данный пакет, я бы воспользовался менеджером пакетов Homebrew для установки:

brew install git

## Установка имени и электронной почты

Для того, чтобы git узнал имя и адрес электронный почты пользователя, необходимо выполнить следующие команды(рис. @fig:001):

git config --global user.name ""  
 git config --global user.email "1032182505@pfru.ru"

## Параметры установки окончаний строк

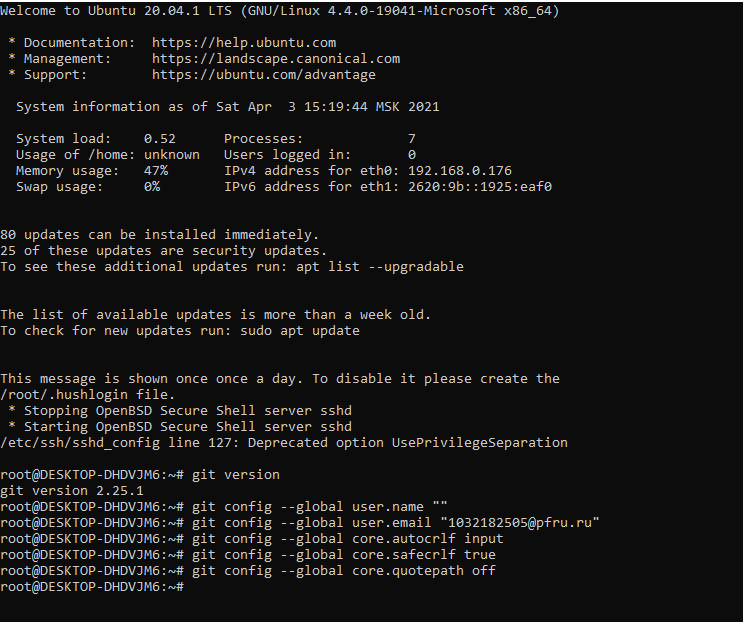
Настроил core.autocrlf с параметром input для того, чтобы все переводы строк текстовых файлов в главном репозитории одинаковы были одинаковы. Конвертация CRLF в LF будет производиться только при коммитах. При настроенном core.safecrlf в true, git будет проверять, если преобразование является обратимым для текущей настройки core.autocrlf, то есть core.safecrlf true - отвержение необратимого преобразования lf<->crlf(рис. @fig:001):

git config --global core.autocrlf input   
 git config --global core.safecrlf true

## Установка отображения unicode

Что бы избежать нечитаемых строк, установил соответствующий флаг (рис. @fig:001):

git config --global core.quotepath off



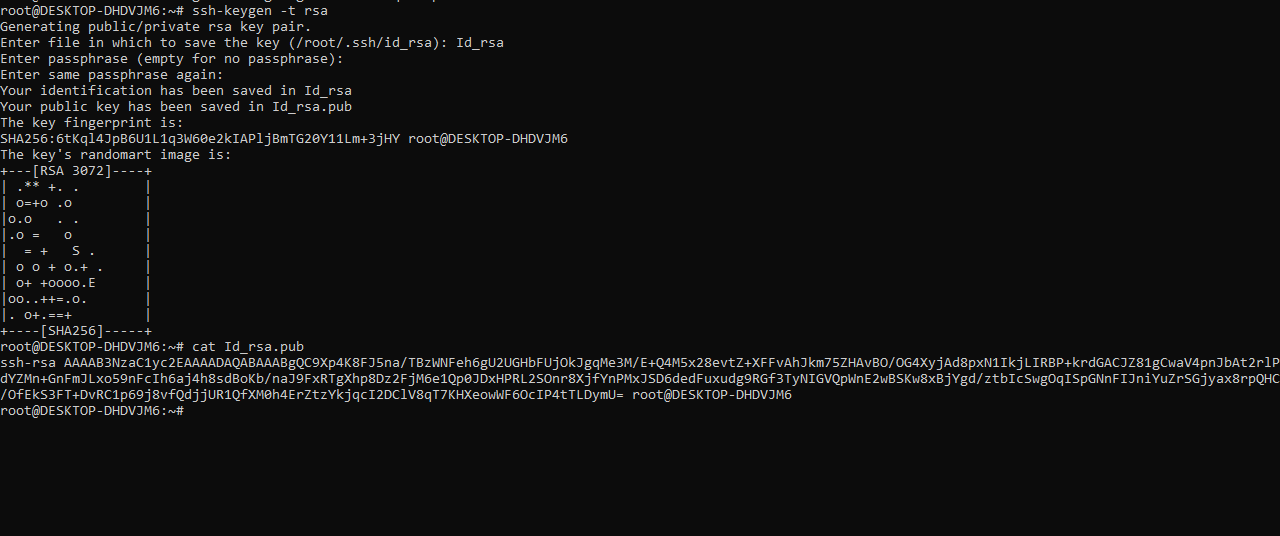
Установка имени и электронной почты и параметры установки окончаний строк

## Создание SSH-ключа с последующей выгрузкой на GitHub

Так как у меня уже имеется учетная запись на GitHub, то мне просто нужно сгенерировать SSH-ключ и выгрузить его на GitHub.

Генерация ключа и его последующее “изымание” из файла, в котором он записан, происходит при помощи следующих команд (рис. @fig:002):

ssh-keygen -t rsa  
 cat /Users//.ssh/Id-rsa.pub



Генерация и чтение SSH-ключа

Далее в настройках профиля на GitHub, во вкладке “SSH and GPG keys” необходимо добавить новый SSH-ключ. Это необходимо для быстрого доступа к репозиториям, которые созданы в моей учетной записи (рис. @fig:003).

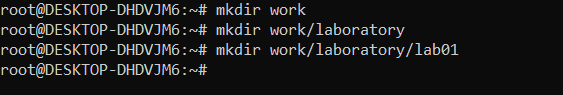


Добавление SSH-ключа на GitHub

## Создание каталога для последующей работы и его выгрузка на GitHub

Для дальнейшей работы с лабораторными работами и проектами необходимо создать директорию с определенной иерархией, которая была предложена ранее. Также была добавлена папка для файлов первой (данной) лабораторной работы (рис. @fig:004):

cd Documents   
 mkdir work  
 mkdir work/laboratory  
 mkdir work/laboratory/lab01

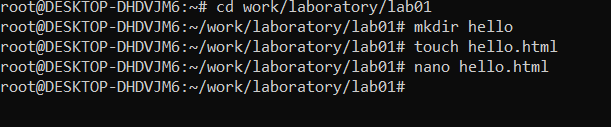


Создание директории

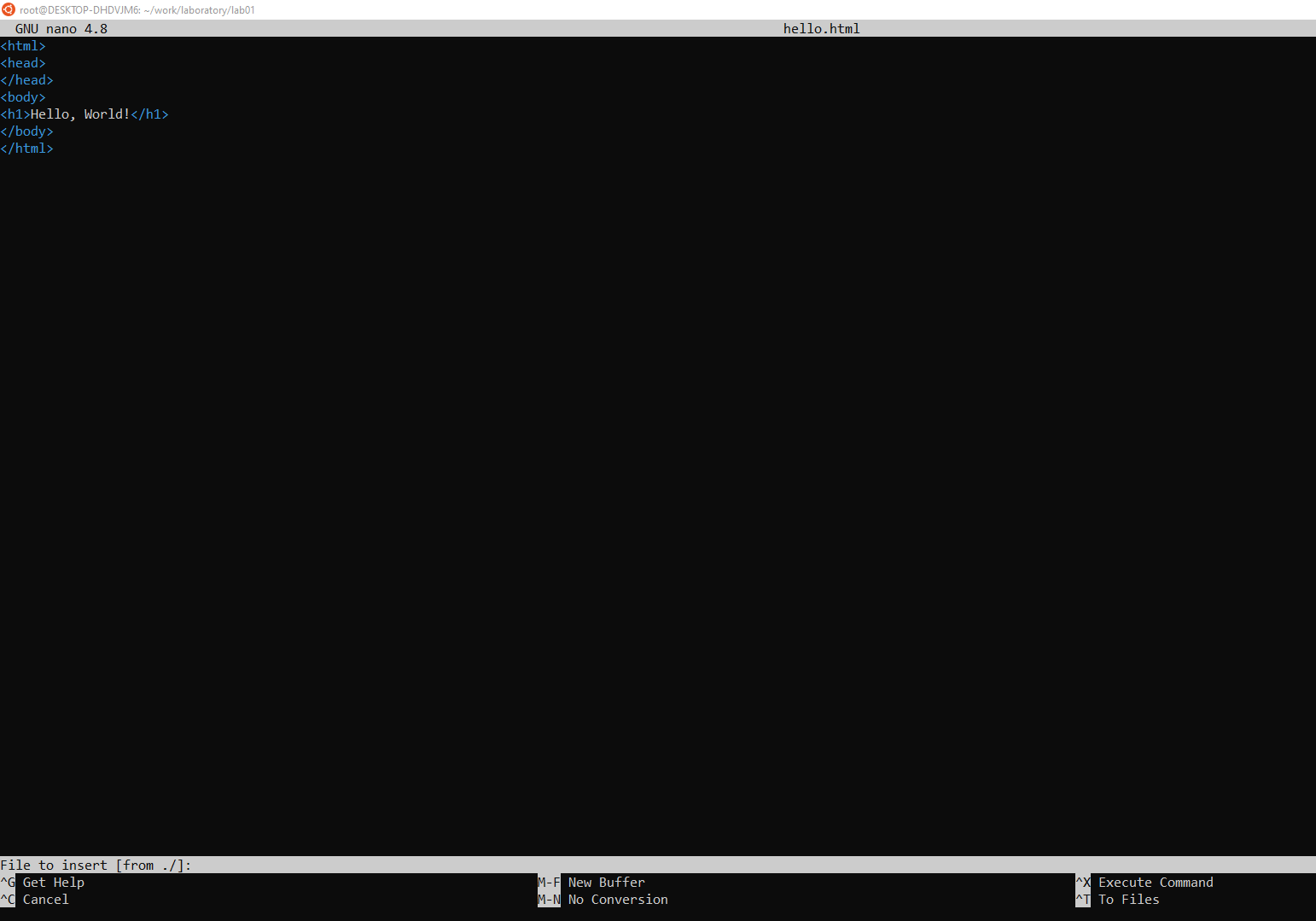
Далее необходимо создать новый репозиторий в GitHub.

Для того, чтобы не инициализировать пустой репозиторий, добавим следуюищий файл с текстом “Hello, World!” (рис. @fig:005, рис. @fig:006):

cd work/laboratory/lab01  
 mkdir hello  
 touch hello.html  
 nano hello.html



Добавление папки hello и файла hello.html



Изменение файла hello.html

Далее необходимо было выполнить. 1. создание репозитория git (‘git init’);

1. добавление всех созданных файлов в репозиторий(‘git add .’);
2. коммит этого этапа (‘git commit -m “Initial Commit”’);
3. проверка текущего состояния репозитория (‘git status’);
4. “привязка” репозитория на GitHub к текущему репозиторию;
5. отправка изменений ветки “master” (‘git push -u origin master’).

cd ~/Documents/work  
 git init  
 git add .   
 git commit -m "Initial Commit"  
 git status  
 git remote add origin git@github.com:/work.git   
 git push -u origin master

Можно увидеть, что в репозитории на GitHub появились данные, которые были добавлены при помощи git (рис. @fig:007):



Репозиторий на GitHub

## Работа с git-flow

В данной лабораторной работе взаимодействие с git-flow осуществляется посредством работы над отчетом и презентацией. Следовательно, добавить в отчет можно лишь часть процесса для более эффективной работы.

Первым делом необходимо скачать git-flow:

brew install git-flow

Далее необходимо инициализировать git-flow в директории work:

git flow init

После этого мы окажемся на ветке “develop”. Для того, чтобы процесс создания отчета был максимально независимым от работы над другими частями лабораторной работы, необходимо переключиться из ветки “develop” на векту фич:

git flow feature start MYFEATURE

При необходимости можно сохранять и публиковатть промежуточные версии при помощи команд:

git flow feature publish MYFEATURE

После окончания работы над отчетом производится слияние ветки фич с веткой разработки и удаление ветки фич:

git flow feature finish MYFEATURE

После окончания работы над фичами необходимо сделать релиз работы. Для этого необходимо создать ветку релиза, ответляя от ветки develop:

git flow release start RELEASE

Публикация релиза выполняется с помощью команды:

git flow release publish RELEASE

Завершить релиз (ветка релиз сливается с веткой master и в ветку develop и удаляется)можно с помощью команды

git flow release finish RELEASE

# Выводы

В ходе выполнения лабораторной работы были изучены основные особенности системы контроля версий git, а также специфика выполнения и оформления лабораторных работ для данной дисциплины.