
Front matter
title: "Лабораторная работа №3"
subtitle: "Архитектура компьютеров и операционные системы. Раздел "Операционные системы""
author: "Дьяконва Софья Александровна НКАбд-01-22"

Generic otions
lang: ru-RU
toc-title: "Содержание"

Цель работы

Цель данной работы --- научиться оформлять отчеты в Markdown, а также познакомиться
с основными возможностями разметки Markdown.

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.

Теоретическое введение

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.

Система контроля версий Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды git с различными опциями.
Благодаря тому, что Git является распределённой системой контроля версий, резервную копию локального хранилища можно сделать простым копированием или архивацией.

еречислим 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 .
```

охранить 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
```

удаление локальной уже слитой с основным деревом ветки:

```
git branch -d имя_ветки
```

принудительное удаление локальной ветки:

```
git branch -D имя_ветки
```

Работа пользователя со своей веткой начинается с проверки и получения изменений из центрального репозитория (при этом в локальное дерево до начала этой процедуры не должно было вноситься изменений):

```
git checkout master
git pull
git checkout -b имя_ветки
```

Если какие-либо файлы не должны попасть в коммит, то помечаем только те файлы, изменения которых нужно сохранить. Для этого используем команды добавления и/или удаления с нужными опциями:

```
git add …
git rm …
```

Для последующей идентификации пользователя на сервере репозиториев необходимо сгенерировать пару ключей (приватный и открытый):

```
ssh-keygen -C "Имя Фамилия <work@mail>"
```

Базовая настройка git

Первичная настройка параметров git

Зададим имя и email владельца репозитория:

```
git config --global user.name "Name Surname"
git config --global user.email "work@mail"
```

Настроим utf-8 в выводе сообщений git:

```
git config --global core.quotepath false
```

Настройте верификацию и подписание коммитов git.

Зададим имя начальной ветки (будем называть её master):

```
git config --global init.defaultBranch master
```

люч ssh создаётся командой:

```
ssh-keygen -t <алгоритм>
```

Создание ключа:

по алгоритму rsa с ключём размером 4096 бит:

```
ssh-keygen -t rsa -b 4096
```

по алгоритму ed25519:

```
ssh-keygen -t ed25519
```

Верификация коммитов с помощью PGP

Как настроить PGP-подпись коммитов с помощью gpg.

Общая информация

Коммиты имеют следующие свойства:

- author (автор) — контрибьютор, выполнивший работу (указывается для справки);
- committer (коммитер) — пользователь, который закоммитил изменения.

Эти свойства можно переопределить при совершении коммита.
Авторство коммита можно подделать.
В git есть функция подписи коммитов.
Для подписывания коммитов используется технология PGP (см. Работа с PGP).
Подпись коммита позволяет удостовериться в том, кто является коммитером. Авторство не проверяется.

Создание ключа

Генерируем ключ

```
gpg --full-generate-key
```

Из предложенных опций выбираем:
тип RSA and RSA;
размер 4096;
выберите срок действия; значение по умолчанию — 0 (срок действия не истекает никогда).
GPG запросит личную информацию, которая сохранится в ключе:
Имя (не менее 5 символов).
Адрес электронной почты.
При вводе email убедитесь, что он соответствует адресу, используемому на GitHub.
Комментарий. Можно ввести что угодно или нажать клавишу ввода, чтобы оставить это поле пустым.

Выполнение лабораторной работы
1. Устанавливаю git

2. Устанавливаю gh

3. Произвожу базовую настройку git: задаю имя владельца и email репозитория, настраиваю utf-8 в выводе сообщений git, Настраиваю верификацию и подписание коммитов git, зададаю имя начальной ветки, параметр autocrlf и safecrlf.

4. Создаю ssh ключ

5. Создаю pgp ключ

6. Настраиваю github

![рис.6. настройка github](/home/sadjyakonova/work/study/2022-2023/Операционные системы/labs/lab03/images/настройка github.png){ #fig:001 width=70% }

7. Добавляю PGP ключ в GitHub

![рис.7. Добавление PGP ключа в GitHub](/home/sadjyakonova/work/study/2022-2023/Операционные системы/labs/lab03/images/Добавление PGP ключа в GitHub.png){ #fig:001 width=70% }

![рис.8. Добавление PGP ключа в GitHub](/home/sadjyakonova/work/study/2022-2023/Операционные системы/labs/lab03/images/Добавление PGP ключа в GitHub.png){ #fig:001 width=70% }

8. Настраиваю автоматические подписи коммитов git

![рис.9. Настройка автоматических подписей коммитов git](/home/sadjyakonova/work/study/2022-2023/Операционные системы/labs/lab03/images/Настройка автоматических подписей коммитов git.png){ #fig:001 width=70% }

9. Настраиваю gh

![рис.10.Настройка gh](/home/sadjyakonova/work/study/2022-2023/Операционные системы/labs/lab03/images/Настройка gh.png){ #fig:001 width=70% }

10. Создаю репозитория курса на основе шаблона

![рис.11 Создание репозитория курса на основе шаблона](/home/sadjyakonova/work/study/2022-2023/Операционные системы/labs/lab03/images/Создание репозитория курса на основе шаблона.png){ #fig:001 width=70% }

11. Настраиваю каталога курса

![рис.12 Настройка каталога курса](/home/sadjyakonova/work/study/2022-2023/Операционные системы/labs/lab03/images/Настройка каталога курса1.png){ #fig:001 width=70% }

![рис.13 Настройка каталога курса](/home/sadjyakonova/work/study/2022-2023/Операционные системы/labs/lab03/images/Настройка каталога курса2.png){ #fig:001 width=70% }

![рис.14 Настройка каталога курса](/home/sadjyakonova/work/study/2022-2023/Операционные системы/labs/lab03/images/Настройка каталога курса3.png){ #fig:001 width=70% }

Контрольные вопросы

1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются? Системы контроля версий (VCS) применяются при работе нескольких человек над одним проектом. Системы контроля версий поддерживают возможность отслеживания и разрешения конфликтов, которые могут возникнуть при работе нескольких человек над одним файлом. Можно объединить (слить) изменения, сделанные разными участниками (автоматически или вручную), вручную выбрать нужную версию, отменить изменения вовсе или заблокировать файлы для изменения.
2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия. Хранилище - репозиторий, где хранится история изменений, а также док-ты, связанные с работой рабочего пространства. Commit - изменения, внесенные пользователями, работающими в одном пространстве. История - данные обо всех изменениях, внесенных в проект. Рабочая копия - последняя версия проекта. 3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида. 13 Централизованные VCS - одно

основное хранилище всего проекта. Каждый пользователь копирует себе необходимые ему файлы из этого репозитория, изменяет, затем добавляет изменения обратно в хранилище. Децентрализованные VCS - у каждого пользователя свой вариант репозитория, есть возможность добавлять и забирать изменения из любого репозитория. В отличие от классических, в децентралиованных системах контроля версий центральный репозиторий не является обязательным.

4. Опишите действия с VCS при единоличной работе с хранилищем. Как и при совместной работе, создается репозиторий, куда по мере продвижения проекта отправляются изменения.
5. Опишите порядок работы с общим хранилищем VCS. Пользователю предоставляется доступ к одной из версий проекта, которую он может редактировать и сохранять изменения, доступные другим участникам проекта.
6. Каковы основные задачи, решаемые инструментальным средством git? Предоставление удобства работы нескольких человек над одним проектом.
7. Назовите и дайте краткую характеристику командам git. Команда Описание git init Создание основного дерева репозитория git pull Получение обновлений (изменений) текущего дерева из центрального репозитория 14 Команда Описание git push Отправка всех произведённых изменений локального дерева в центральный репозиторий git status Просмотр списка изменённых файлов в текущей директории git diff Просмотр текущих изменений git add . Добавить все изменённые и/или созданные файлы и/или каталоги git commit Сохранить добавленные изменения с внесением комментария через встроенный редактор git checkout Создание новой ветки, базирующейся на текущей
10. Что такое и зачем могут быть нужны ветви (branches)? Ветви нужны для того, чтобы программисты могли вести совместную работу над одним проектом вместе, но при этом не мешали друг другу.

Выводы

Я за время проделанной работы научилась оформлять отчеты в Markdown, а также познакомилась с основными возможностями разметки Markdown.

Список литературы{.unnumbered}

**[Лабораторная работа №
2](https://esystem.rudn.ru/mod/page/view.php?id=970819#orgf425532)**