Отчёта по лабораторной работе №2

НКНбд-02-21

Акондзо Жордани Лади Гаэл

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Цель данного работы — Изучить идеологию и применение средств контроля версий. Освоить умения по работе с git.

2 Задание

- Создать базовую конфигурацию для работы с git.
- Создать ключ SSH.
- Создать ключ PGP.
- Настроить подписи git.
- Зарегистрироваться на Github.
- Создать локальный каталог для выполнения заданий по предмету.

3 Теоретическое введение

По определению система контроля версий (VCS) это программное обеспечение для облегчения работы с изменяющейся информацией. Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом. Git является распределённой системой контроля версий. Система контроля версий Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды git с различными опциями. В табл. 3.1 приведено краткое описание наиболее часто используемых команд git.

Таблица 3.1: Описание основных команд git

Команда	**Описание действия**
`git init`	создание основного дерева репозитория
`git pull`	получение обновлений (изменений) текущего дерева из центрального репозитория
`git status`	просмотр списка изменённых файлов в текущей директории
`git push`	отправка всех произведённых изменений локального дерева в центральный репозиторий
`git diff`	просмотр текущих изменения
`git add .`	добавить все изменённые и/или созданные файлы и/или каталоги
`git add` `имена_файлов`	добавить конкретные изменённые и/или созданные файлы и/или каталоги

Команда	**Описание действия**
`git rm имена_файлов`	удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории)
`git commit -am 'Коммит'`	сохранить все добавленные изменения и все изменённые файлы
`git commit`	сохранить добавленные изменения с внесением комментария через встроенный редактор
`git checkout -b имя_ветки`	создание новой ветки, базирующейся на текущей
`git checkout имя_ветки`	переключение на некоторую ветку
`git push origin имя_ветки`	отправка изменений конкретной ветки в центральный репозиторий
`git branch -d имя_ветки`	удаление локальной уже слитой с основным деревом ветки
`git branch -D имя_ветки`	принудительное удаление локальной ветки
`git push origin :имя_ветки`	удаление ветки с центрального репозитория

4 Ход работы

1. Настройка github

Зарегистрировался на Github. (сайт https://github.com) (рис.4.1)

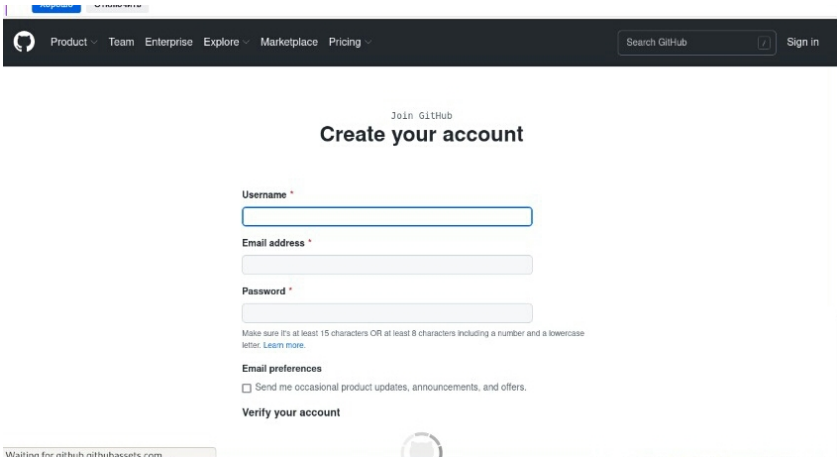

Рис. 4.1: Регистрация в github

2. Установка git-flow в Fedora Linux

Установил его вручную, введя следующие команды: (рис. 4.2)

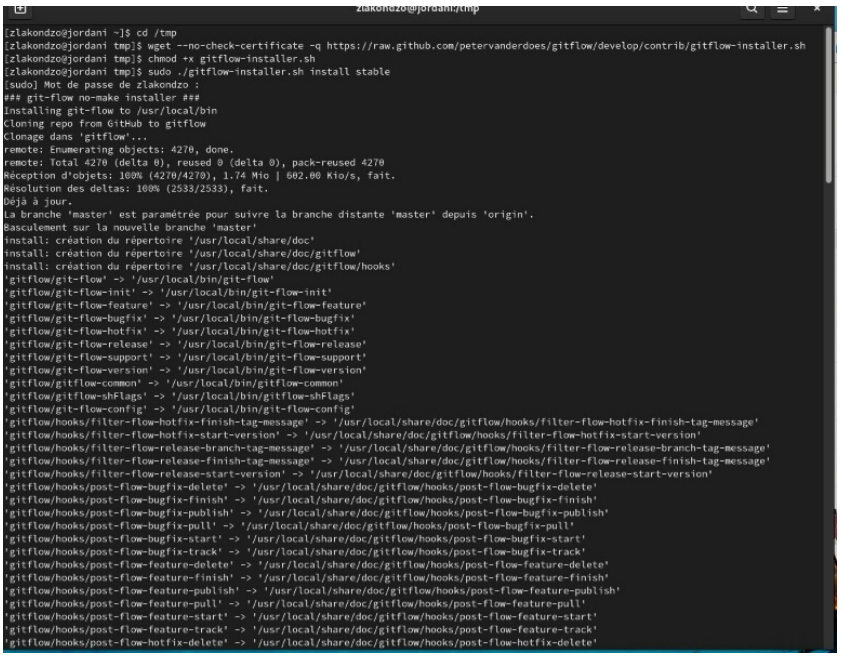

Рис. 4.2: Установка git-flow в Fedora Linux

3. Установка gh в Fedora Linux

Использовал следующую команду: (рис. 4.3)

Рис. 4.3: Установка gh в Fedora Linux

4. Базовая настройка GIT

- Задал имя и email владельца репозитория.
- Настроил utf-8 в выводе сообщений git.
- Задал имя начальной ветки, настроил параметры autocrlf и safecrlf.gpg (рис. 4.4)

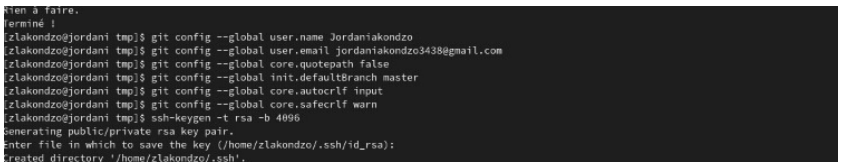

Рис. 4.4: Базовая настройка GIT

5. Создание ключа gpg

- Сгенерировал ключ ssh по алгоритму rsa и ed25519 (рис. 4.5)

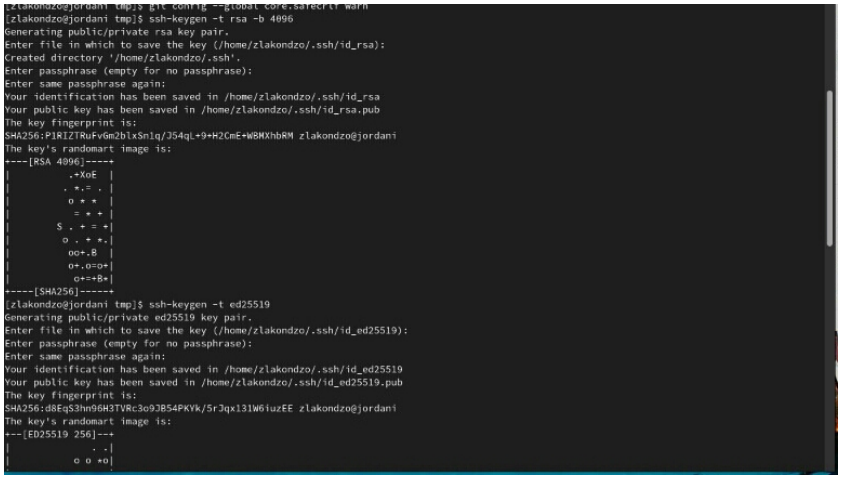

Рис. 4.5: Генерирование ключа ssh

- Сгенерировал ключ pgp. (рис. 4.6)

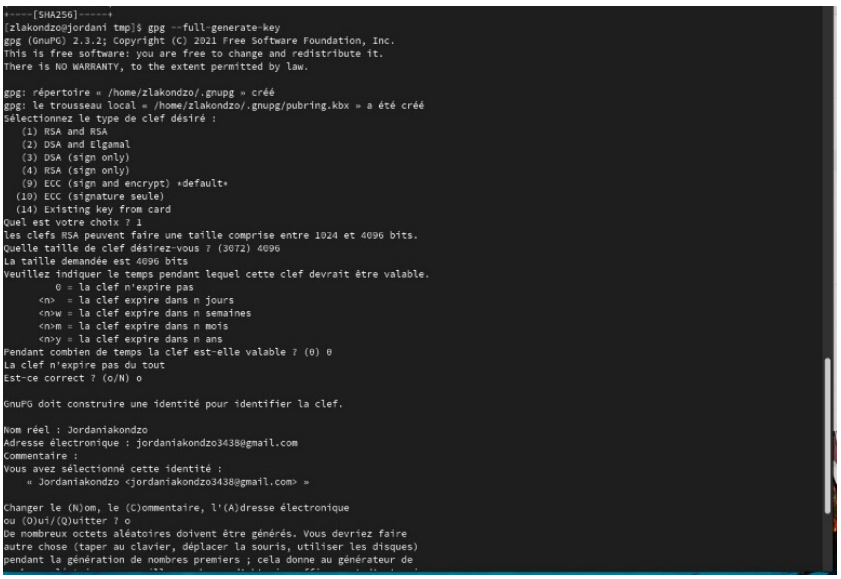

Рис. 4.6: Генерирование ключа pgp

6. Добавление pgp ключа в Github

- Выводил список ключей и копировал отпечаток приватного ключаю.
- Скопировал мой сгенерированный pgp ключ в буфер обмена. (рис. 4.7)

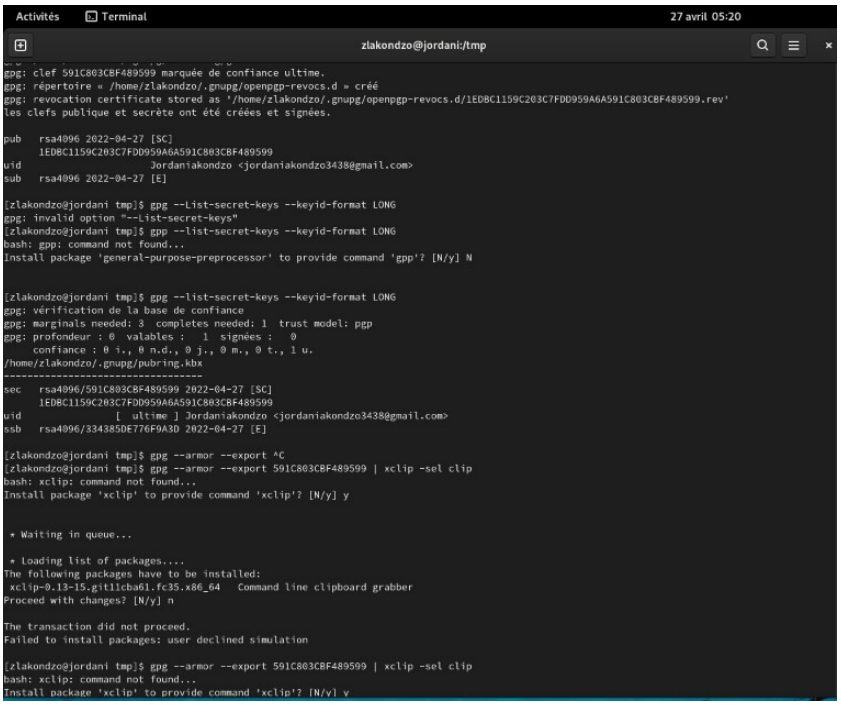

Рис. 4.7: Добавление pgp ключа в Github

7. Настройка автоматических подписей коммитов git

- Настроил автоматические подписи коммитов git: (рис. 4.8)

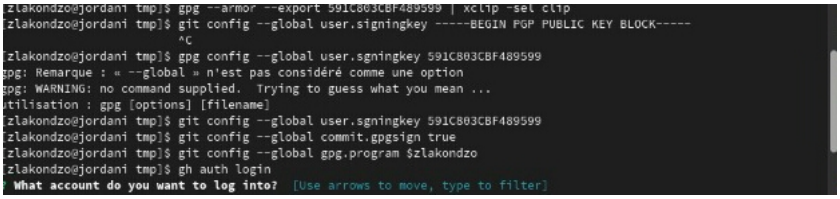

Рис. 4.8: Подписание коммитов git

8. Настройка gh

- Затем настроил gh (рис. 4.9, 4.10)

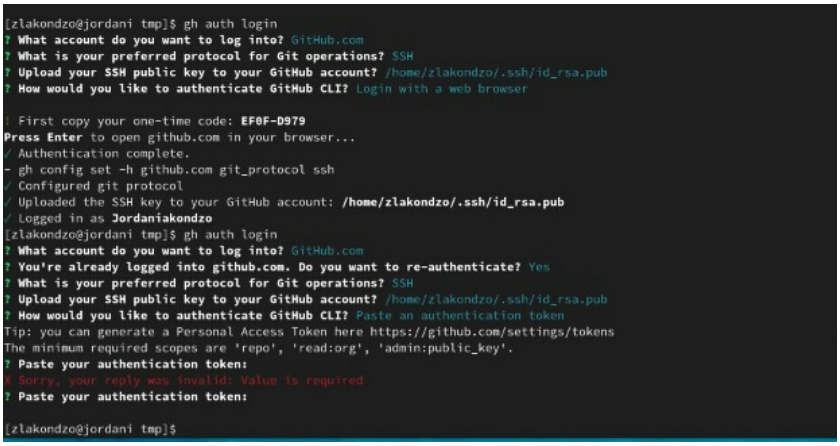

Рис. 4.9: Настройка gh

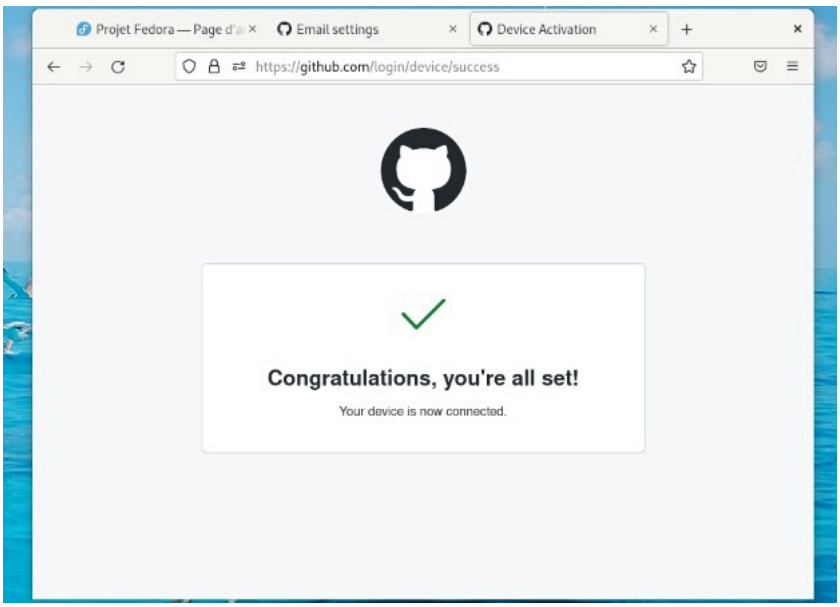

Рис. 4.10: Настройка gh

9. Создание репозитория курса на основе шаблона

- Создал шаблон рабочего пространства (рис. 4.11)

Рис. 4.11: Создание репозитория курса

10. Настройка каталога курса

- Перейдил в катог курса.
- Удалил лишние файлы.
- Создал необходимо каталоги. (рис. 4.12)

Рис. 4.12: Настройка каталога курса 1

- Отправил файлы на сервер. (рис. 4.13, 4.14)

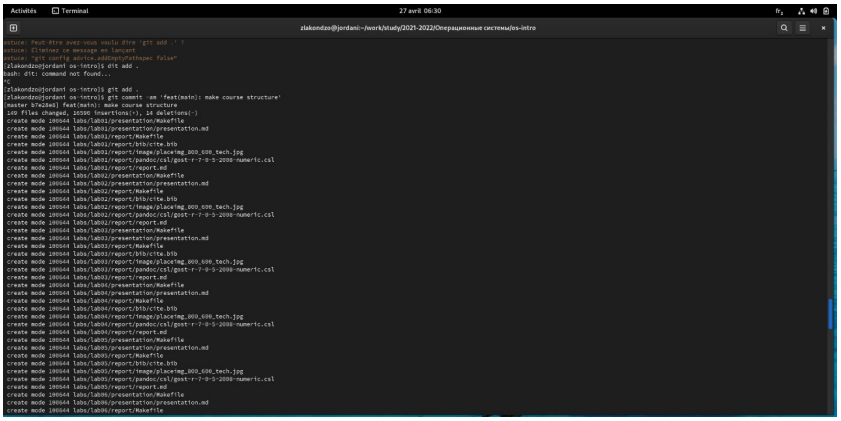

Рис. 4.13: Настройка каталога курса 2

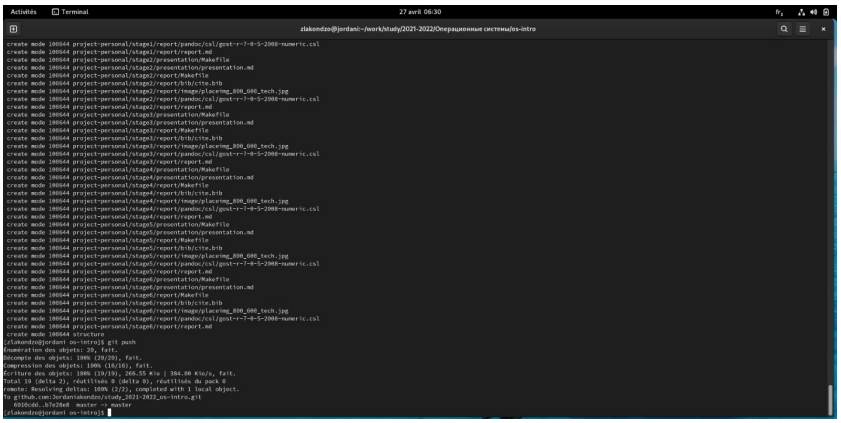

Рис. 4.14: Настройка каталога курса 2

5 Выводы

Таким образом, мы только что создали репозиторий курса “Операционные Системы”, основанный на шаблоне профессора на github. Это позволит всем студентам класса а одновременно работать над одним проектом.

6 Контрольные вопросы

1. **Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются?**

    - Система контроля версий (VCS) - программное обеспечение для облегчения работы с изменяющейся информацией. Система управления версиями позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое.
    - Такие системы наиболее широко используются при разработке программного обеспечения для хранения исходных кодов разрабатываемой программы. Однако они могут с успехом применяться и в других областях, в которых ведётся работа с большим количеством непрерывно изменяющихся электронных документов.

2. **Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.**

    - Хранилище – репозиторий - место хранения всех версий и служебной информации.
    - Commit - это команда для записи индексированных изменений в репозиторий.
    - История – место, где сохраняются все коммиты, по которым можно посмотреть данные о коммитах.
    - Рабочая копия – текущее состояние файлов проекта, основанное на

версии, загруженной из хранилища.

3. **Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида.**

   - Централизованные системы – это системы, в которых одно основное хранилище всего проекта, и каждый пользователь копирует необходимые ему файлы, изменяет и вставляет обратно.
   - Пример – Subversion.
   - Децентрализованные системы – система, в которой каждый пользователь имеет свой вариант репозитория и есть возможность добавлять и забирать изменения из репозиториев.
   - Например, Git.

4. **Опишите действия с VCS при единоличной работе с хранилищем.**

   - Сначала сделаем предварительную конфигурацию, указав имя и email владельца репозитория: 1 git config –global user.name “Имя Фамилия” 2 git config –global user.email “work@mail” и настроив utf-8 в выводе сообщений git: 1 git config –global quotepath false Для инициализации локального репозитория, расположенного, например, в каталоге ~/tutorial, необходимо ввести в командной строке: Кулябов Д. С. и др. Операционные системы 25 1 cd 2 mkdir tutorial 3 cd tutorial 4 git init После это в каталоге tutorial появится каталог .git, в котором будет храниться история изменений. Создадим тестовый текстовый файл hello.txt и добавим его в локальный репозиторий:

   1. echo ‘hello world’ > hello.txt
   2. git add hello.txt
   3. git commit -am ‘Новый файл’

   - Воспользуемся командой status для просмотра изменений в рабочем каталоге, сделанных с момента последней ревизии: 1 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:

1. curl -L -s https://www.gitignore.io/api/list Затем скачать шаблон, например, для C и C++
2. curl -L -s https://www.gitignore.io/api/c » .gitignore
3. curl -L -s https://www.gitignore.io/api/c++ » .gitignore

5. **Опишите порядок работы с общим хранилищем VCS.**

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

1. ssh-keygen -C “Имя Фамилия”

- Ключи сохраняться в каталоге ~/.ssh/. Существует несколько доступных серверов репозиториев с возможностью бесплатного размещения данных. Например, https://github.com/. Для работы с ним необходимо сначала завести на сайте https://github.com/ учётную запись. Затем необходимо загрузить сгенерённый нами ранее открытый ключ. Для этого зайти на сайт https://github.com/ под своей учётной записью и перейти в меню GitHub setting . После этого выбрать в боковом меню GitHub setting SSH-ключи и нажать кнопку Добавить ключ . Скопировав из локальной консоли ключ в буфер обмена

1. cat ~/.ssh/id_rsa.pub | xclip -sel clip

- И так вставляем ключ в появившееся на сайте поле. После этого можно создать на сайте репозиторий, выбрав в меню Репозитории Создать

репозиторий , дать ему название и сделать общедоступным (публичным). Для загрузки репозитория из локального каталога на сервер выполняем следующие команды: 1 git remote add origin

2. ssh://git@github.com//.git
3. git push -u origin master

- Далее на локальном компьютере можно выполнять стандартные процедуры для работы с git при наличии центрального репозитория.

6. **Каковы основные задачи, решаемые инструментальным средством git?**

- У Git две основных задачи:

1. Первая — хранить информацию обо всех изменениях в вашем коде, начиная с самой первой строчки,
2. А вторая — обеспечение удобства командной работы над кодом.

7. **Назовите и дайте краткую характеристику командам git.**

Команда	**Описание действия**
`git init`	создание основного дерева репозитория
`git pull`	получение обновлений (изменений) текущего дерева из центрального репозитория
`git status`	просмотр списка изменённых файлов в текущей директории
`git push`	отправка всех произведённых изменений локального дерева в центральный репозиторий
`git diff`	просмотр текущих изменения
`git add .`	добавить все изменённые и/или созданные файлы и/или каталоги

Команда	Описание действия
`git add имена_файлов`	добавить конкретные изменённые и/или созданные файлы и/или каталоги
`git rm имена_файлов`	удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории)
`git commit -am 'Коммит'`	сохранить все добавленные изменения и все изменённые файлы
`git commit`	сохранить добавленные изменения с внесением комментария через встроенный редактор
`git checkout -b имя_ветки`	создание новой ветки, базирующейся на текущей
`git checkout имя_ветки`	переключение на некоторую ветку
`git push origin имя_ветки`	отправка изменений конкретной ветки в центральный репозиторий
`git branch -d имя_ветки`	удаление локальной уже слитой с основным деревом ветки
`git branch -D имя_ветки`	принудительное удаление локальной ветки
`git push origin :имя_ветки`	удаление ветки с центрального репозитория

8. **Приведите примеры использования при работе с локальным и удалённым репозиториями**

   - Для инициализации локального репозитория, расположенного, напри-

мер, в каталоге ~/tutorial, необходимо ввести в командной строке:

1. cd
2. mkdir tutorial
3. cd tutorial
4. git init

- После это в каталоге tutorial появится каталог .git, в котором будет храниться история изменений. Создадим тестовый текстовый файл hello.txt и добавим его в локальный репозиторий:

1. echo ‘hello world’ > hello.txt
2. git add hello.txt 3 git commit -am ‘Новый файл’ Воспользуемся командой status для просмотра изменений в рабочем каталоге, сделанных с момента последней ревизии:
3. git status

9. **Что такое и зачем могут быть нужны ветви (branches)?**

- Ветка (англ. branch) — это последовательность коммитов, в которой ведётся параллельная разработка какого-либо функционала. 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.

10. **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. В Git нет специальной команды для указания игнорируемых файлов: вместо этого необходимо вручную отредактировать файл .

- Временно игнорировать изменения в файле можно с командой git update-index-assumeunchanged