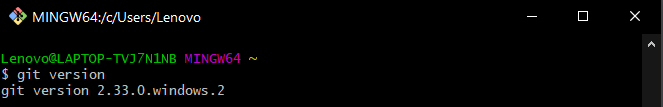
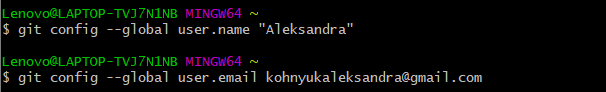
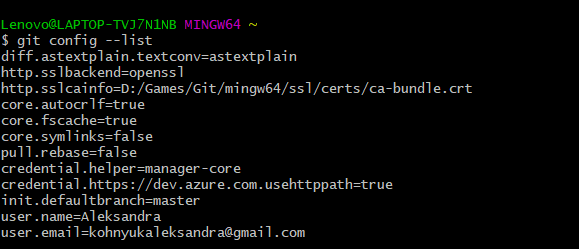
**Лабораторная работа № 1 Системы контроля версий**

**1) Версия, настройки имя пользователя и т.п**

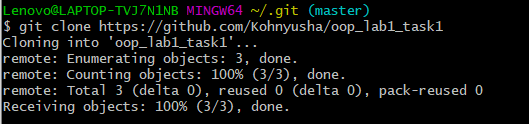


**

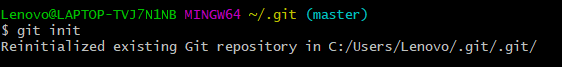
Проверить выбранные настройки позволяет команда **git config --list**, выводящая список всех обнаруженных в текущий момент параметров.



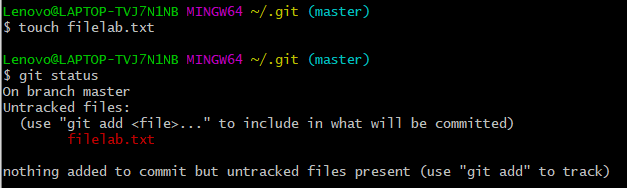
**2) Научитесь клонировать репозитории (clone). Например, созданный ранее на** [**https://github.com.**](https://github.com/)



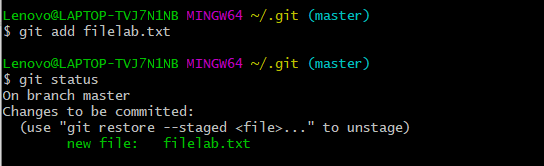
**3) Создаем локальный репозиторий.**

****

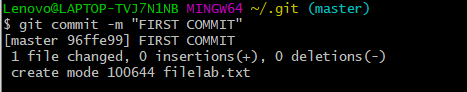
**4) Создаем файл и просматриваем его статус. Статус не индексированный**

****

**5) Индексируем новый файл.**

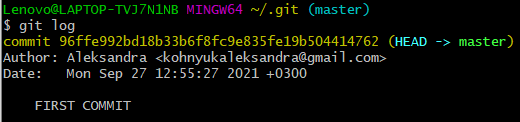
****

**6) Добавляем commit с сообщением ( состояние репозитория в определенный момент времени или фиксация изменений)**

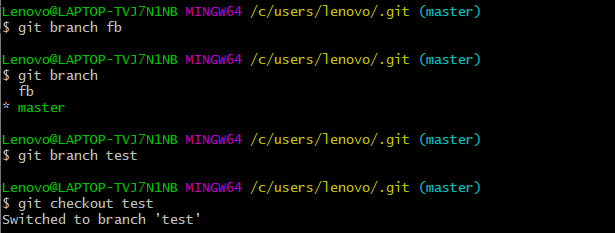


**7) Команда git log.**

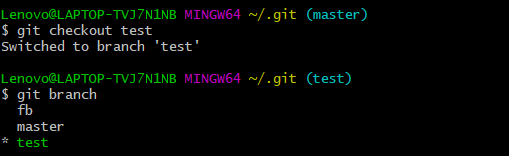
Выводит в обратном хронологическом порядке список сохраненных в данный репозиторий версий. То есть первыми показываются самые свежие коммиты. Как видите, рядом с каждым коммитом указывается его контрольная сумма SHA-1, имя и электронная почта автора, дата создания и сообщение о фиксации.



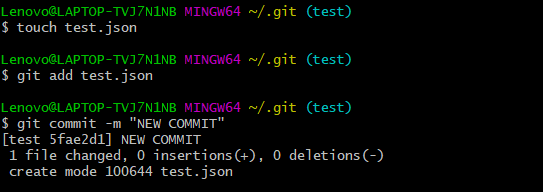
**8) Работа с ветками.**



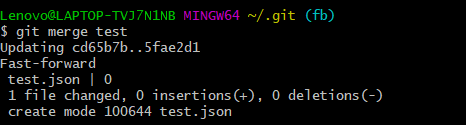
**9) Переключения с ветки на ветку. Звездочкой помечена текущая ветка.**

****

**10) Добавление нового файла и выполнение коммита.**

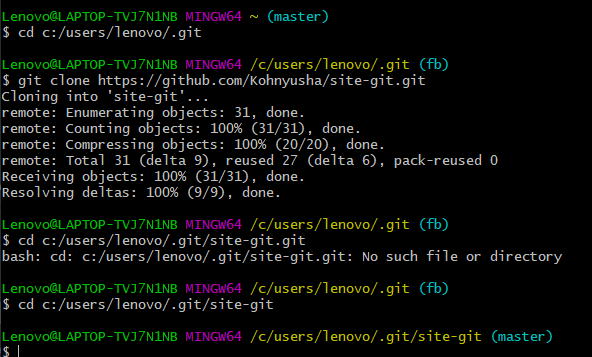
****

**12) Слияние веток.**



1. **Совместная работа**

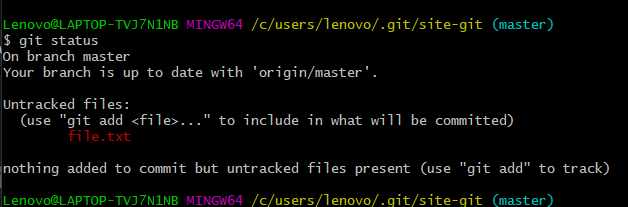
* **Клонируем себе репозиторий**



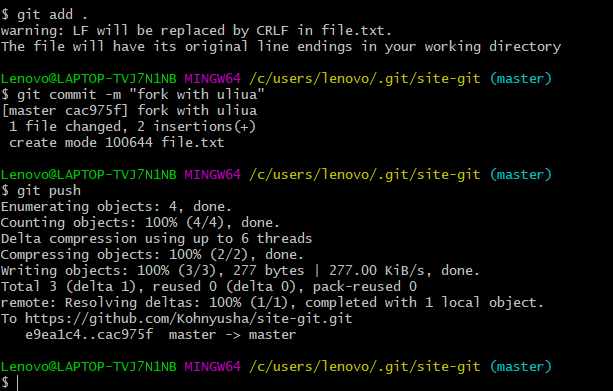
Переносим в удаленный репозиторий

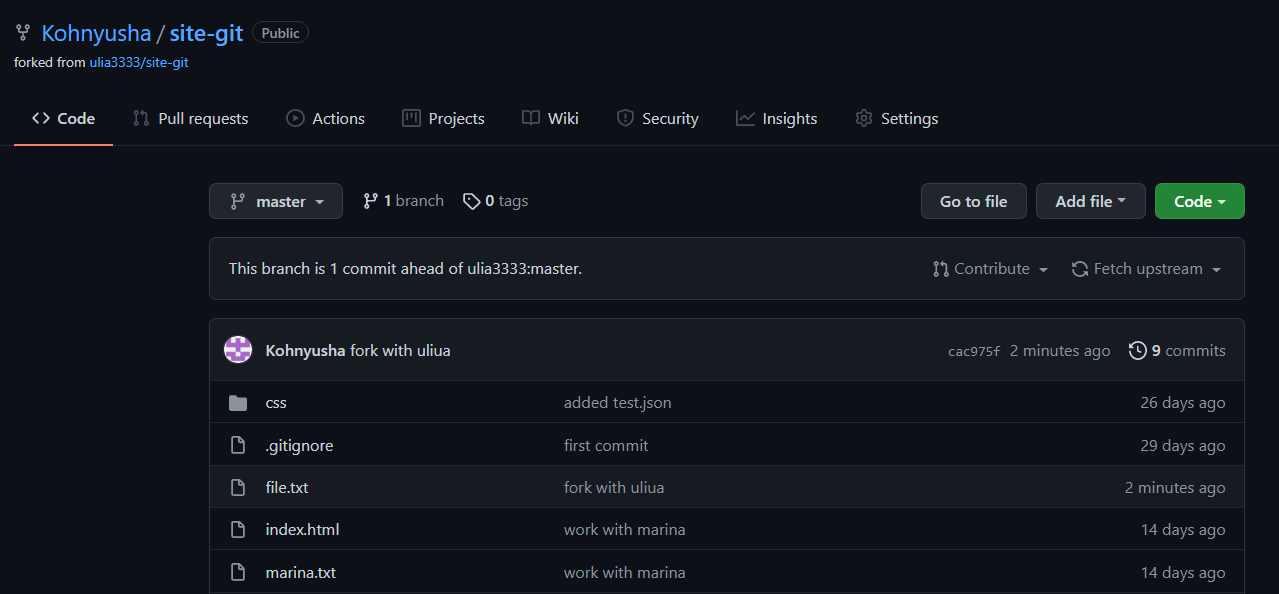


Проверяем статус

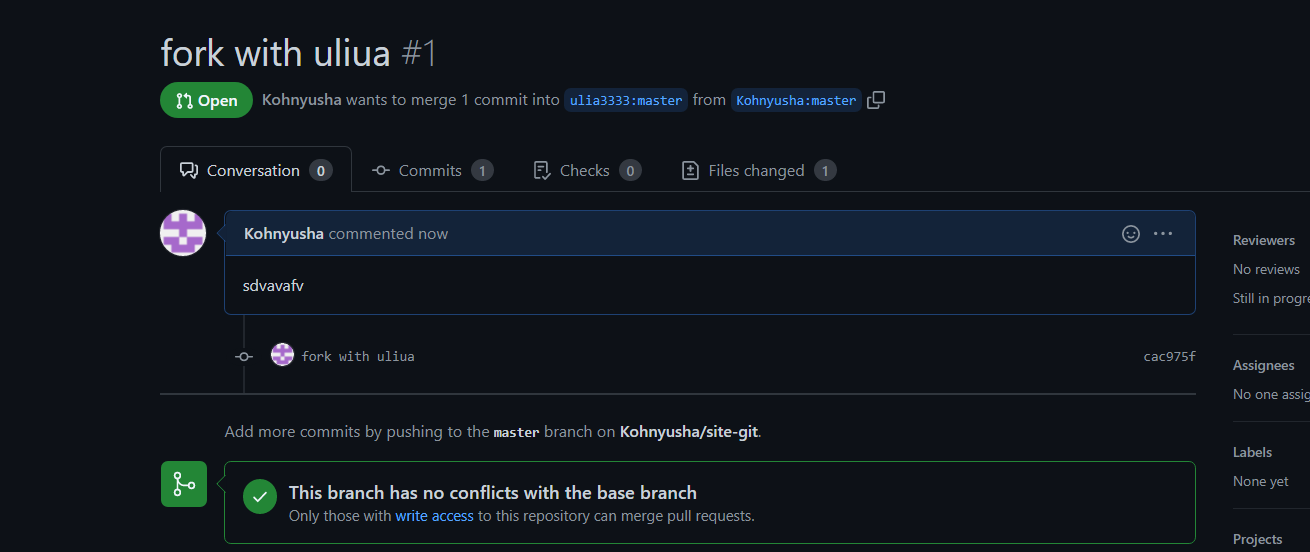


Вносим изменения и фиксируем

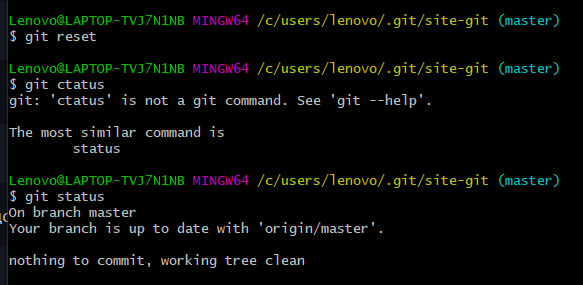


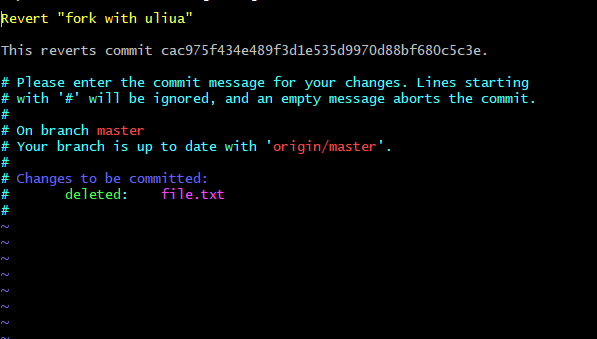


Предложила внести изменения



Git reset





Reset checkout ignore

gitignore нужен для скрытия файлов и папок от системы контроля версий

checkout – восстанавливает файлы до определенногт состояния

копирует файлы со сцены в рабочую директорию. Эту **команду** удобно **использовать**, чтобы сбросить нежелательные изменения в рабочей директории

### git reset — возврат к определенному коммиту, откат изменений, «жесткий» или «мягкий»

«Мягкий» (с ключом --soft) резет оставит нетронутыми ваши индекс и все дерево файлов и директорий проекта, вернется к работе с указанным коммитом. Иными словами, если вы обнаруживаете ошибку в только что совершенном коммите или комментарии к нему, то легко можно исправить ситуацию

«Жесткий» резет (ключ --hard) — команда, которую следует использовать с осторожностью. git reset --hard вернет дерево проекта и индекс в состояние, соответствующее указанному коммиту, удалив изменения последующих коммитов

# **Вопросы**

1. **Что такое система контроля версий, для чего ее используют?**

Git - это система контроля версий, позволяющая эффективно управлять историей исходного кода. Защищает исходный код от потери, Обеспечивает командную работу, Помогает отменить изменения.

1. **Какой принцип хранения файлов использует Git?**

Git автоматически не сохраняет состояние вашей папки. Вы можете сохранить эти изменения только вручную. процесс сохранения происходит в 3 этапа.

1 - Производятся изменения внутри папки.

2 - Сообщаем программе Git какие изменения нужно сохранить.

3 - Создается сама точка сохранения.

Эта точка сохранения в программе Git она называется commit (коммит) или слепок.

1. **В чем отличие git от других систем контроля версий? ??????**

Есть также SVN, Mercurial. Значительное увеличение быстродействия,

1. **В каких трех основных состояниях файлы могут находиться в Git ?**

изменённые, индексированные и зафиксированные.

Изменённый означает, что вы изменили файл, но ещё не зафиксировали его в своем локальном репозитории.

Индексированный - это изменённый файл, текущую версию которого вы отметили для включения в следующий коммит (для фиксации в своём локальном репозитории. Зафиксированный означает, что файл уже сохранён в вашем локальном репозитории

1. **Что такое индексация файла? Какой командой она выполняется?**

это процесс добавления текущего содержимого (изменённого) **файла** в индекс (также: index area, staging area). командой add:

1. **Что такое фиксация файла? Какой командой она выполняется?**

Это сохранение изменений в файле. Командра git commit

1. **Как инициализировать репозиторий в существующей папке?**

Инициализация начинается с выполнения команды init, далее git add , git commit – m “”

1. **Как указать файлы, за которыми должна следить система?**

Для этого потребуется несколько команд **git add**, добавляющих файлы, за которыми вы хотите следить, а затем команда **git commit**:

1. **Как выполнить фиксацию изменений?**

Командра git commit

1. **Какой командой определить состояния файлов?**

Git status

1. **Для чего создается файл .gitignore?Поясните его структуру.**

**Файл** .**gitignore** предназначен для исключения из индексации **Git** **файлов** и папок проекта. Размещаться этот **файл** может в любой папке проекта и количество этих **файлов** в разных папках не ограничено.

1. **Как используется команда git diff?**

**команда**, которую нужно обязательно запускать перед каждым коммитом. Она позволяет проанализировать добавляемые изменения и исправить возможные ошибки.

1. **Как используется команда git commit?**

это **команда** для записи индексированных изменений в репозиторий **Git**. Прежде чем создавать очередной **коммит**, необходимо проиндексировать файлы в рабочей области с помощью **команды** **git**-add.

1. **Как используется команда git log? Какие у нее есть параметры?**

**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.

Один из наиболее полезных параметров — это -p, который показывает дельту (разницу/diff), привнесенную каждым коммитом. Вы также можете использовать -2, что ограничит вывод до 2-х последних записей

Если вы хотите получить некоторую краткую статистику по каждому коммиту, вы можете использовать параметр –stat

1. **Как используется команда git commit –amend?**

 "забывает" про текущий **коммит**, берёт тот же **коммит** на котором был основан старый\*, делает поверх него\* новый **коммит**, учитывая содержимое индекса\*\* и переносит ветку на получившийся результат

1. **Как отобразить удаленные репозитории?**

git remote (–v)

1. **Как извлечь данные из удаленного репозитория?**

git remote rm:

1. **Как отправить данные в удаленный репозитроий?**

git remote add [сокращение] [url]

1. **Какая команда позволяет отобразить удаленные репозитории, связанные с текущим локальным?**

git remote (–v)

1. **Каким образом можно получить изменения из удаленного репозитория в локальный?**

Помещение push и получение pull

1. **Для чего используется команда fetch? В чем отличие команды fetch от pull?**

это основная **команда**, которая **используется** для загрузки содержимого из удаленного репозитория. Чтобы обновить локальный репозиторий до состояния удаленного репозитория, **команда** git **fetch** **используется** в сочетании с **командами** git remote, git branch, git checkout и git reset/

**Команда** git **pull** отвечает за скачивание данных с сервера. Процесс очень похож на клонирование репозитория, но здесь скачиваются не все коммиты, а только новые. По сути, git **pull** — это сочетание **команд** git **fetch** (загружает коммиты, ссылки, файлы из удаленного репозитория в локальный) и git merge (объединяет несколько коммитов в один общий).

1. **Для чего используется команда merge? В чем отличие merge от rebase?**

Cлияние. Слияние принимает содержимое ветки источника и объединяет их с целевой веткой

Слейте ветку master в ветку feature, используя команды *checkout* и *merge*.

$ git checkout feature

$ git merge master

(or)

$ git merge master feature

В отличие от слияния, перемещение перезаписывает историю, потому что она передает завершенную работу из одной ветки в другую. В процессе устраняется нежелательная история

1. **Что такое pull request?**

это запрос на вливание изменений из вашей ветки в основную ветку исходного репозитория. Таким образом они попадут к хозяевам проекта.

1. **Какая команда позволяет отобразить историю репозитория?**

**Команда** git log **отображает** отправленные снимки состояния и **позволяет** просматривать и фильтровать **историю** проекта, а также искать в ней конкретные изменения

1. **Продемонстируйте создание новых веток в Git? Что такое ветвление? Что такое указатель HEAD?**

Ветка в Git — это простой перемещаемый указатель на один из таких коммитов. По умолчанию, имя основной ветки в Git — master.

git branch testing

Как Git определяет, в какой ветке вы находитесь? Он хранит специальный указатель HEAD. Имейте ввиду, что в Git концепция HEAD значительно отличается от других систем контроля версий. В Git — это указатель на текущую локальную ветку. В нашем случае мы все еще находимся в ветке master. Команда git branch только **создаёт** новую ветку, но не переключает на неё.

1. **Как используется команда git checkout?**

копирует файлы со сцены в рабочую директорию. Эту **команду** удобно **использовать**, чтобы сбросить нежелательные изменения в рабочей директории

1. **Как выполнить включение изменений из одной ветки в другую?**

Pull push

1. **Какие проблемы могут быть при слиянии и как они разрешаются?**

Обычно конфликты возникают, когда два человека изменяют одни и те же строки в файле или один разработчик удаляет файл, который в это время изменяет другой разработчик. В таких случаях Git не может автоматически определить, какое изменение является правильным

Выполнение команды слияния прерывается в самом начале, если Git обнаруживает изменения в рабочем каталоге или разделе проиндексированных файлов текущего проекта. Git не может выполнить слияние, поскольку иначе эти ожидающие изменения будут перезаписаны новыми коммитами.

Сбой В ПРОЦЕССЕ слияния говорит о наличии конфликта между текущей локальной веткой и веткой, с которой выполняется слияние. Это свидетельствует о конфликте с кодом другого разработчика. Git сделает все возможное, чтобы объединить файлы, но оставит конфликтующие участки, чтобы вы разрешили их вручную

1. **Что такое GitLab?**

веб-приложение и система управления репозиториями программного кода для **Git**. **GitLab** предлагает решение для хранения кода и совместной разработки масштабных программных проектов