Государственный Университет Молдовы

Факультет Математики и Информатики

Департамент Информатики

**Лабораторная работа №3**

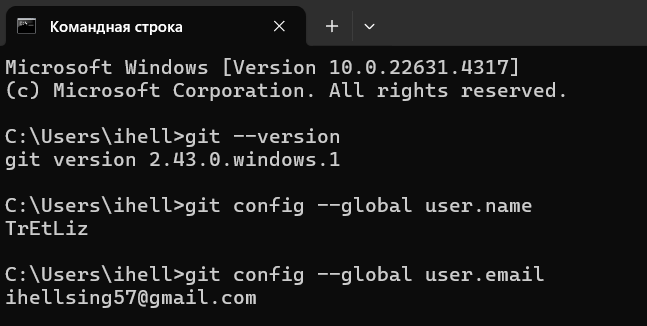
**Основы Git. Команды Git.**

Проверил: доцент, др. A. Prepelita

Выполнила: Tretiacova Elizaveta, I2202

Кишинэу 2024

Чтобы узнать, установлен ли Git, и если да, то под каким именем он настроен, нужно выполнить следующую команду:



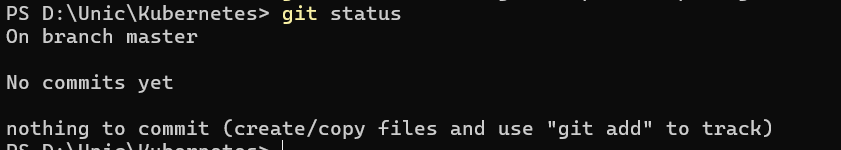
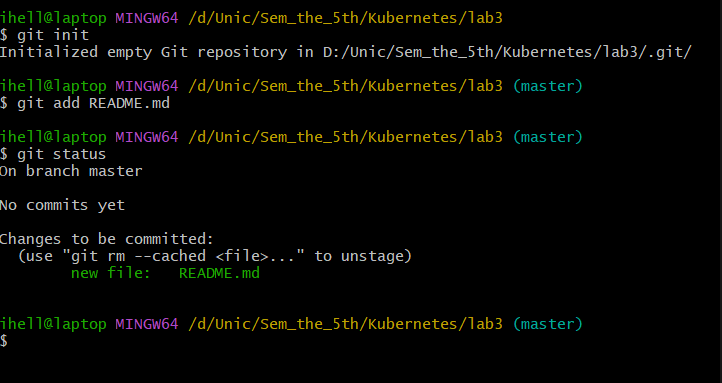
**git config --global user.name** - команда покажет имя пользователя, которое настроено для Git.

**git config --global user.email** - команда покажет email, который настроен для Git.

**git init** - эта команда используется для инициализации нового репозитория Git. Когда выполняется git init в папке, Git создает скрытую директорию .git, которая содержит все метаданные и историю версий для данного проекта. После выполнения этой команды мы можем начать отслеживать изменения в файлах и добавлять их в репозиторий.

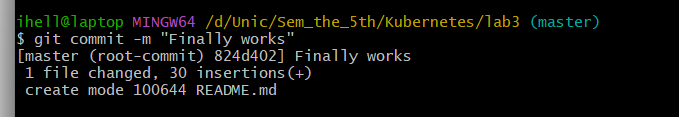
Команда git remote add origin <new-url> используется в Git для добавления удаленного репозитория, который будет использоваться для отслеживания и синхронизации локального репозитория.

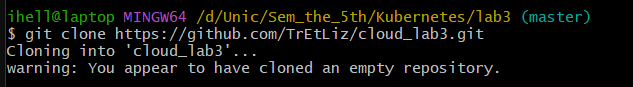
https://github.com/TrEtLiz/kubernetesss.git

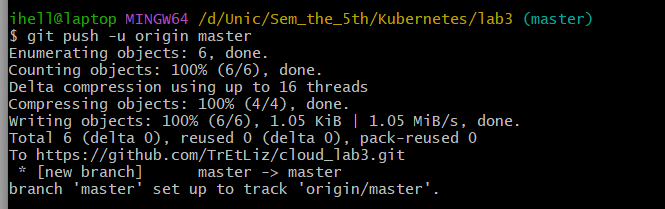
Команда **git status** используется в Git для отображения состояния текущего локального репозитория. Она предоставляет информацию о том, какие изменения были внесены в файлы, какие из них отслеживаются, а какие нет, а также какие изменения готовы к коммиту.

Команда **git add** используется в Git для добавления изменений в индекс (или «стадию»), чтобы подготовить их к следующему коммиту. Это важный шаг в процессе управления версиями, так как позволяет вам выбирать, какие изменения мы хотим включить в коммит.



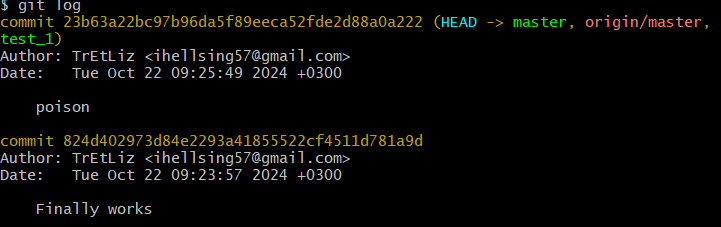
Команда **git commit -m** используется в Git для создания нового коммита с сообщением. Она сохраняет все изменения, которые были добавлены в индекс (с помощью git add), в истории репозитория. Параметр **-m** позволяет вам указать сообщение коммита непосредственно в командной строке. **Не отслеживаемые (новые) файлы** не будут добавлены в коммит.  


Команда **git clone <url>** копирует удалённый репозиторий и автоматически добавляет его как удалённый с именем по умолчанию — origin.  
  

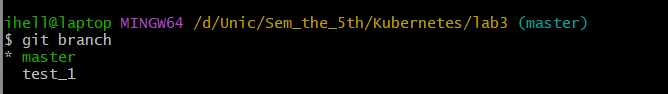

**git remote add origin url** используется для добавления локальной ветки в удаленный репозиторий  
**git push -u master** – загрузка файлов на origin github.  


Команда **git branch test\_1** используется в Git для создания новой ветки с именем test\_1

git log [--oneline --decorate] - Команда для краткого отображения истории коммитов с дополнительной информацией.

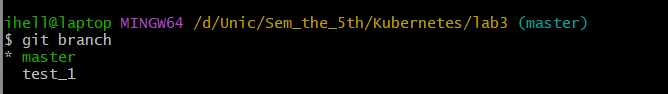


**git log** -Показывает историю коммитов для текущей ветки.

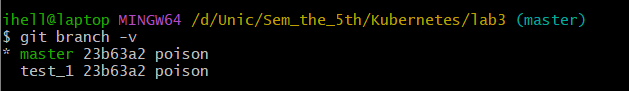


**git branch**

* Показывает список всех локальных веток в текущем репозитории.



**git branch -v**

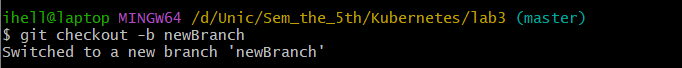
* Показывает список всех локальных веток вместе с их последними коммитами.  
  

**git branch --all**

* Показывает список всех локальных и удалённых веток.  
  

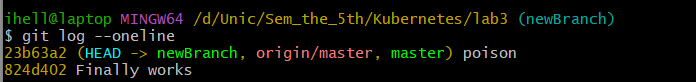
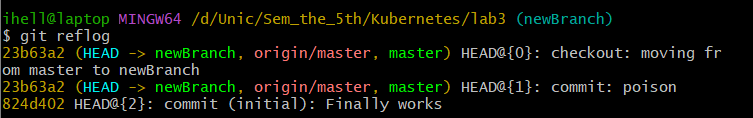
**git branch -d testing**

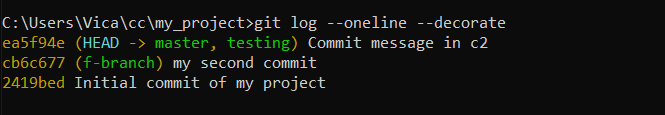
* Удаляет локальную ветку с именем testing.

**git checkout -b new-branch-name**

Эта команда одновременно создаёт новую ветку и переключается на неё.

  
  
  
**git log**

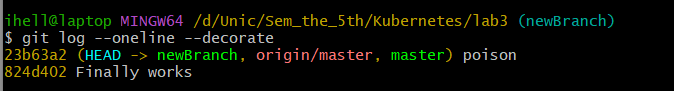
показывает список коммитов с их хэшами, автором, датой и сообщением.  
  


**git log [--oneline --decorate]**

Команда для краткого отображения истории коммитов с дополнительной информацией.

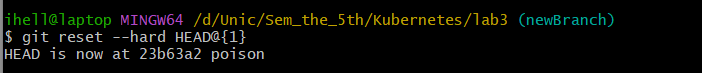
--oneline: Выводит коммиты в одной строке (короткий хэш и сообщение).

--decorate: Показывает текущие ветки и теги, привязанные к коммитам.

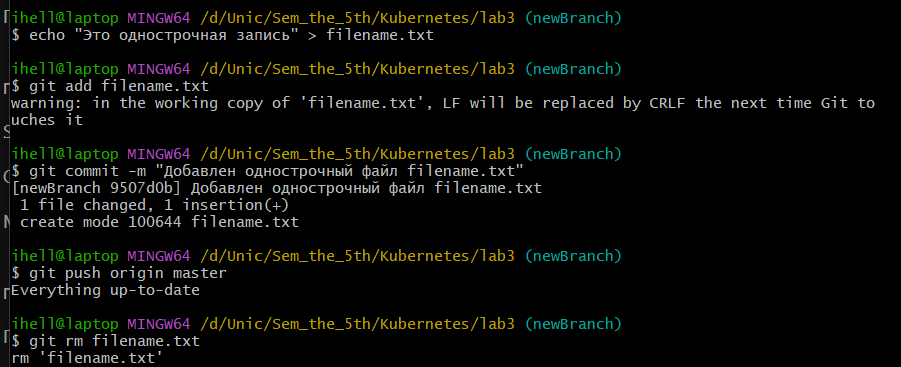
  

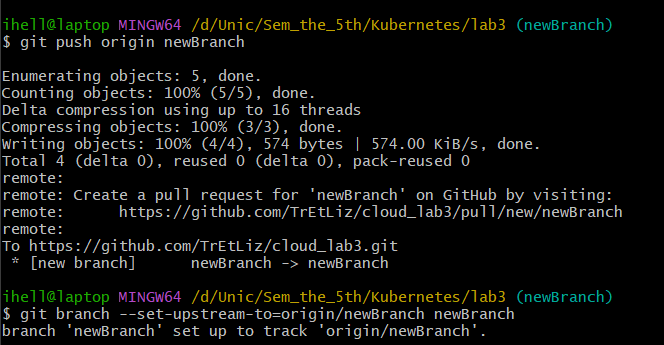

**git reset**

Команда для сброса состояния текущей ветки на указанный коммит.  
git reset --hard HEAD@{n} - Сброс на предыдущее состояние вашего репозитория

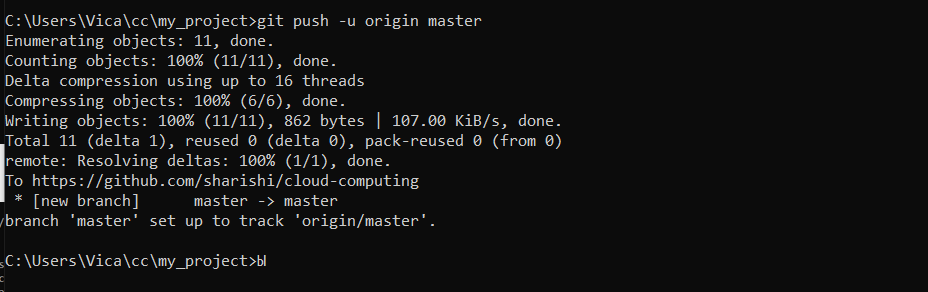


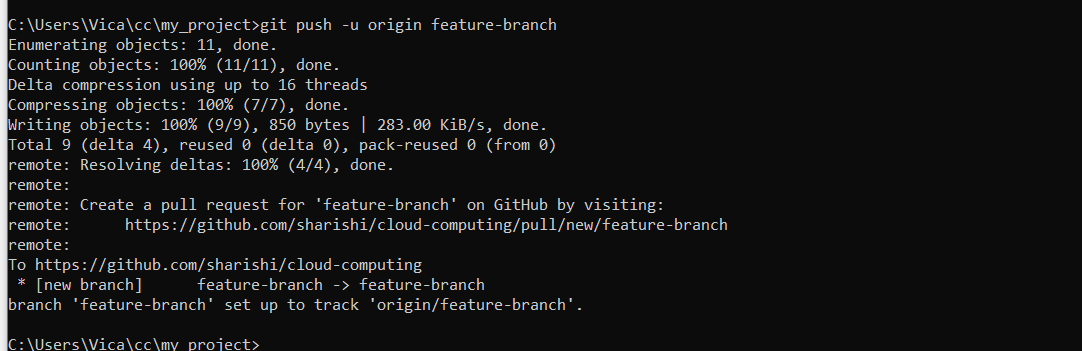
**git rm**

Команда для удаления файлов из рабочего каталога и индекса (стадии), что означает, что файлы будут удалены из следующего коммита.  


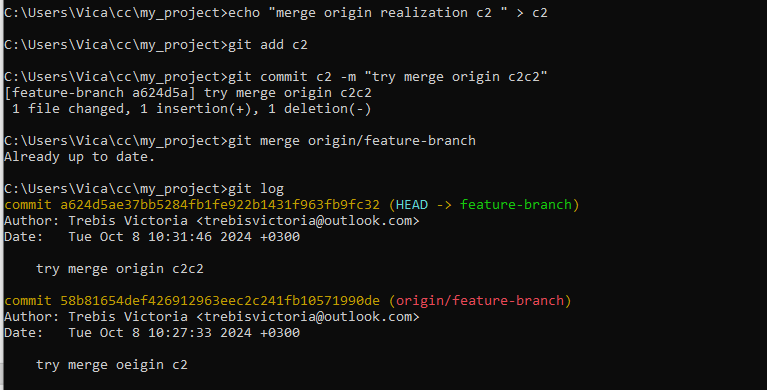


**git merge -** Используется для объединения изменений из одной ветки в другую. Эта команда объединяет ветку feature-branch в текущую ветку.

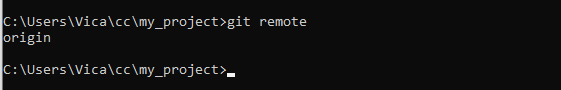


После подключения удаленного репозитория, мы можем отправить наши локальные изменения в него используя **git push -u origin master**  
  
  


Команда git push -u origin feature-branch используется для отправки изменений из нашей локальной ветки feature-branch в удаленный репозиторий, который обозначен как origin

  
**git merge origin/feature-branch**

Это команда для слияния удалённой ветки feature-branch из удалённого репозитория origin в текущую ветку.



**git remote**

Команда для управления удалёнными репозиториями. Она может быть использована для просмотра списка удалённых репозиториев, добавления новых или изменения существующих.

Git не знает, в какую ветку на удалённом репозитории нужно отправить изменения, поэтому нужно установить связь между локальной веткой testing и удалённой веткой. Для этого можно использовать команду с опцией --set-upstream.



**git rm**

Удаление файла из Git

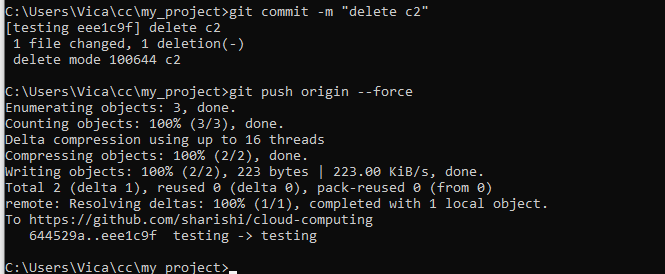


**git switch -c new-branch-name**

Альтернативный и более современный способ переключения на новую ветку (начиная с Git версии 2.23).

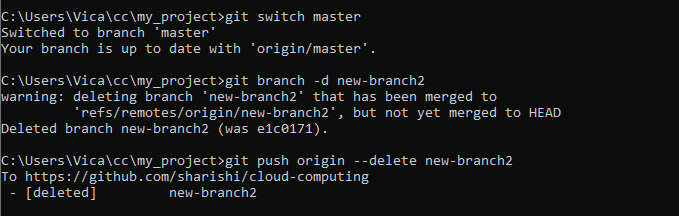
Аналогичен **git checkout -b**, но более понятен в использовании.

**Переключение на новую ветку**:



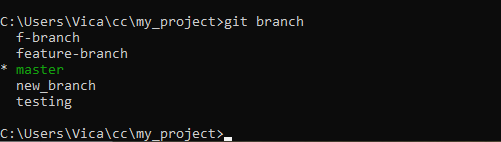
При попытке удалить ветку new-branch2 с помощью git branch -d new-branch2 возникает ошибка, так как ветка используется в рабочем дереве (worktree).

Команда git switch master переключает контекст на ветку master. Консоль сообщает, что ветка master синхронизирована с удаленной веткой origin/master.



После переключения на ветку master команда git branch -d new-branch2 выполняется успешно, но выводит предупреждение, что ветка new-branch2 была объединена с удаленной, но еще не объединена с текущей (HEAD) веткой.

Команда git push origin --delete new-branch2 удаляет ветку new-branch2 из удаленного репозитория origin. Консоль подтверждает успешное удаление.



**Вывод**

В ходе выполнения лабораторной работы я изучила основы работы с системой контроля версий Git, что позволило мне освоить создание и управление репозиториями, а также применение различных команд Git для эффективного контроля версий. Работа с Git значительно упрощает процесс разработки, обеспечивая возможность отслеживания изменений в коде, их возврата к предыдущим состояниям и управления совместной работой над проектом. Освоение инструментов, предоставляемых Git, является ключевым элементом в современном процессе разработки программного обеспечения, так как способствует улучшению качества кода и упрощает взаимодействие между участниками команды.