## **Курс «Контроль версий с помощью GIT»**

* + 1. **Основные концепты работы системы контроля версий**
* Возможность создавать резервную копию и восстанавливаться из нее
* Возможность синхронизировать работу нескольких разработчиков
* Возможность быстро отменить все сделанные изменения
* Возможность отслеживания изменений и авторов этих изменений
* Возможность создания изолированных систем с целью тестирования последних изменений
* Возможность создания веток и слияния этих веток
  + 1. **Типы версионного контроля**

1. Lock-modify-unlock

Основана на принципе, что когда одному разработчику требуется внести изменения, то он закрывает доступ к проекту, вносит изменения и открывает его снова. Другой разработчик может сделать тоже самое.

1. Copy-modify-merge

Любое количество разработчиков может работать и вносить правки в проект в любое время. Далее все объединяется.

К последнему типу версионного контроля и относится GIT.

* + 1. **Установка и конфигурация GIT**

Загрузить гит можно тут: [http://git-scm.com/downloads](https://git-scm.com/downloads" \t "[object Object])

Установить гит можно с настройками по умолчанию.

Генерируем пары ssh ключей. Для этого используем команду:

     ssh-keygen -t rsa -C "vitali\_shulha@epam.com"

Публичный ключ (id\_rsa) нужно отправить владельцу репозитория для получения прав работы. Или загрузить в настройки профиля в [bitbucket](https://bitbucket.org/product/)/[github](https://github.com/)/[gitlab](https://about.gitlab.com/)

Настраиваем имя пользователя и емейл:

   git config --global user.name “Roma Birilo“

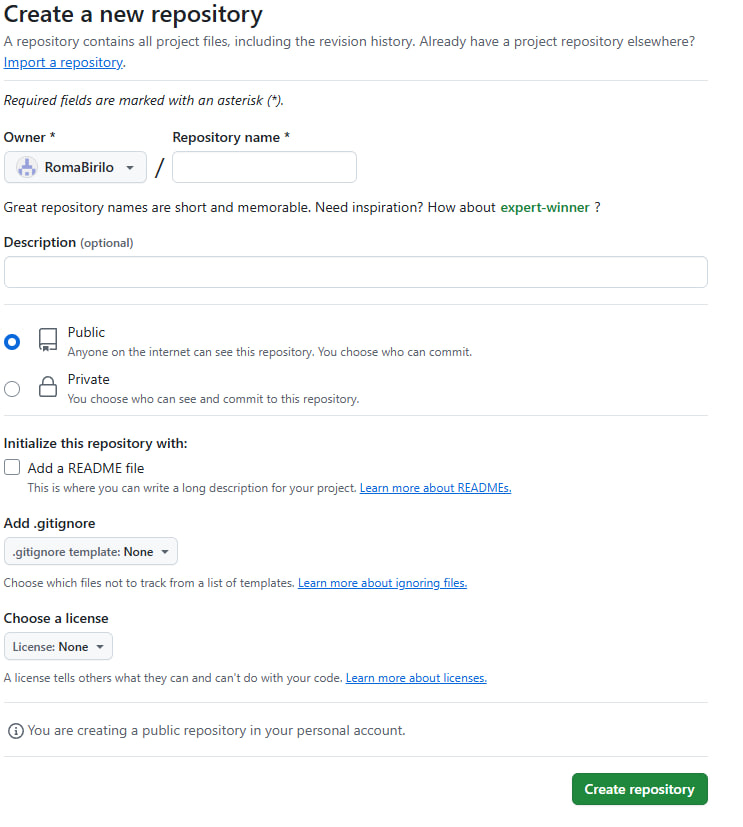
  git config --global user.email “gkdichfjf@gmail.com"

* + 1. **Создание репозитория и его клонирование**

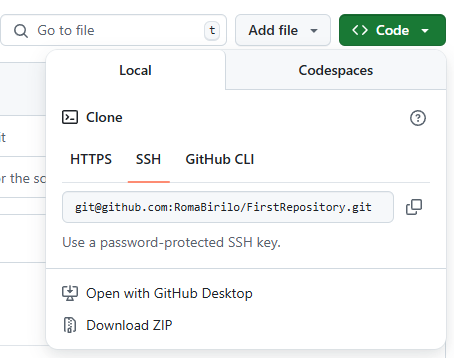
Репозиторий можно создать на платформе [GitHub.com](https://github.com).

Для этого нужно войти в свой аккаунт или создать новый. Затем нажать на «плюс» в правом верхнем углу 

В открывшемся окне вводим название репозитория и нажимаем «Создать репозиторий»



Затем на [GitHub](https://github.com) переходим в созданный репозиторий и копируем SSH ключ

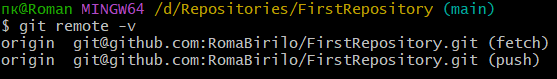


Далее в консоле GitBash вводим команду cd для выбора директории.



Затем вводим команду git clone и вставляем SSH ключimage_2025-07-04_23-16-12

Используем команду git remote –v, чтобы посмотреть, откуда был взят репозиторий.



* + 1. **Создание коммита, отправка на удаленный репозиторий и подтягивание изменений**

После внесения необходимых изменений в файлы используем команду git add, чтобы поставить приготовить изменения, которые будут занесены в коммит.



Далее командой git commit –m ”Название коммита” создаем коммит



Для того, чтобы отправить новые коммиты на удаленный репозиторий,

Воспользуемся командой git push.



Чтобы подтянуть изменения, которые появились на удаленном репозитории можно воспользоваться командой git fetch, чтобы просмотреть изменения и командой git merge, чтобы добавить их. Или можно воспользоваться командой git pull, которая совмещает в себе две предыдущие:

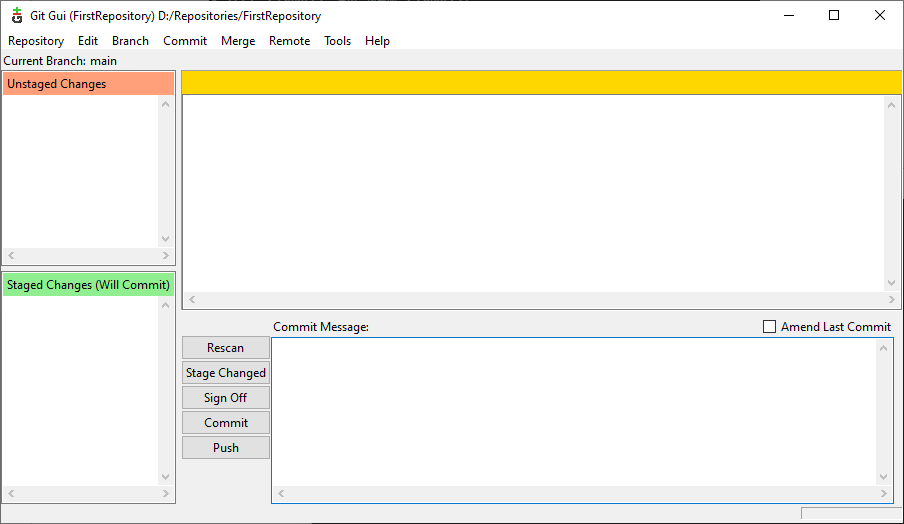


* + 1. **Графические интерфейсы git gui и gitk**

Для более удобного использования Git предоставляет графические интерфейсы такие как git gui и gitk.

Вызываем git gui:



У нас должно появиться следующее окно: 

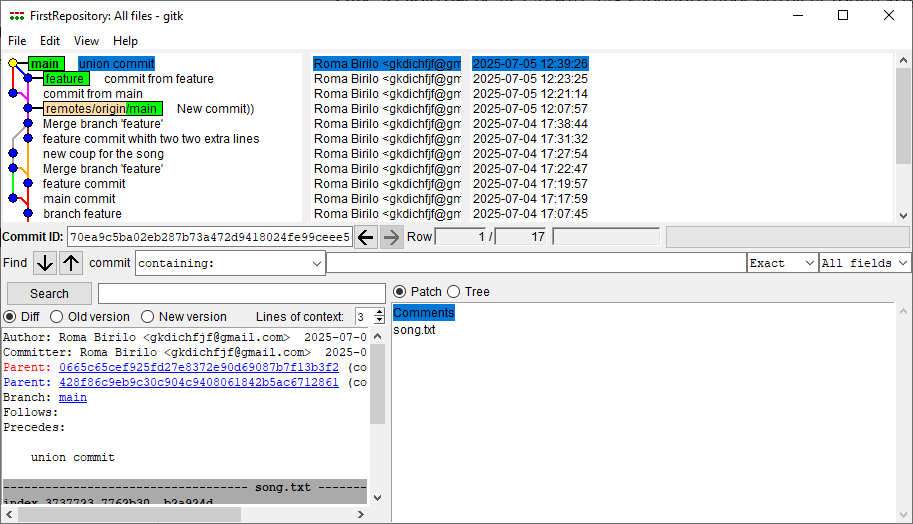
Git gui применяется для более удобного создания коммитов и отправки их на удаленной репозиторий без использования командной строки.

Gitk ,в свою очередь, служит для удобного просмотра истории коммитов

Вызываем gitk:



Появляется такое окно:



* + 1. **Отмена изменений**

В Git можно отменять изменения, которые произошли на разных уровнях.

1. File system

Используем команду git checkout -- file name для того, чтобы отменить изменения, которые были сделаны в этом файле.

Такой способ подходит для тех файлов, которые еще не были добавлены в индекс (Staging area).



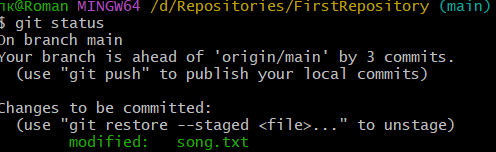
Также можно использовать команду git checkout . , чтобы отменить изменения во всех файлах текущей директории и ниже.

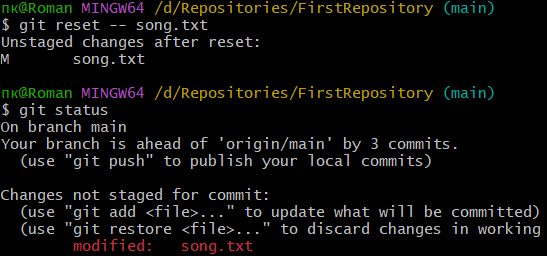


Команда git clean –xdf удалят новые файлы.

1. Staging area

Если файл уже был помещен в staging area, но пондобиллось его достать оттуда, то нужно воспользоваться командой git reset -- file name. И только потом использовать команды из предыдущего пункта.





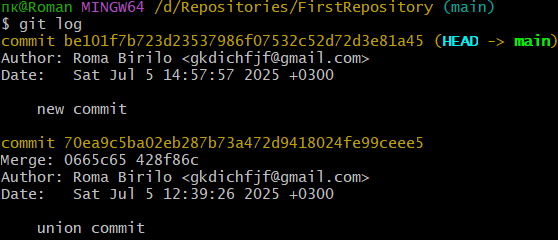
1. Commit

Для того, чтобы добавить что-то в существующий коммит используем команду git commit --amend -m “commit message”.

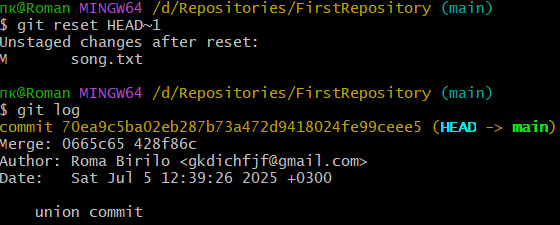


Если же необходимо совсем удалить один или несколько последних коммитов, то используем команду git reset HEAD~n. Вместо n необходимо указать, какое количество коммитов нужно отменить.

До git reset HEAD~1:



После:



Команда git log используется для просмотра истории коммитов.

Также существует набор команд git reset --soft , git reset --mixed, git reset --hard.

Они используются для того, чтобы удалить коммит и прейти в staging area, прейти на уровень файловой системы и удалить коммит, никуда не переходя соответственно.

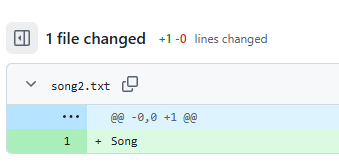
Важно помнить, что снова вернуться к версии проекта до отмены изменений нельзя.

1. Remote repository

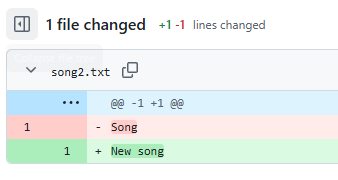
Для отмены изменений, которые находятся в репозитории

Используем комнду git revert <sha1>.

Текст проекта до изменений:



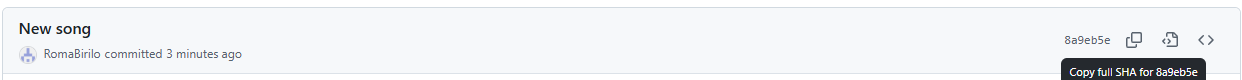
После:



Используем команду git revert <sha1>:



Берем sha1 того коммита, который хотим отменить



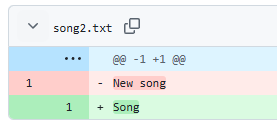
В открывшемся окне нажимаем «:» и прописываем wq:



Эта команда создаст противоположный коммит, относительно того, чей sha1 мы взяли.

Теперь отправляем новый коммит на уделенный репозиторий.

После команды git revert:



Все изменения отменились.

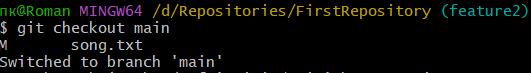
* + 1. **Ветки и слияние веток**

Ветвление позволяет изолировать изменения, позволяя работать параллельно и не вредить рабочему коду. Далее все необходимые ветки можно слить в основную.

Для создания ветки: git checkout –b “branch name”



Для перехода между ветками git checkout “branch name”



Чтобы слить ветки используем команду git merge “branch name”



Выделяется два вида слияние веток:

* Fast-forward. Переносит указатель с главной ветки на конец второстепенной.
* Non fast-forward. Создает новый коммит, в котором второстепенная ветка сливается с главной.

При слиянии веток может возникнуть конфликт. Чаще всего такое может случиться, когда изменения в разных ветках происходят в одном файле. В таком случае необходимо выбрать какой из вариантов файла необходим и создать итоговый коммит.

git rebase переносит коммиты из одной ветки на вершину другой.

git cherry-pick копирует один конкретный коммит из другой ветки и применяет его в текущую ветку, не затрагивая остальную историю.

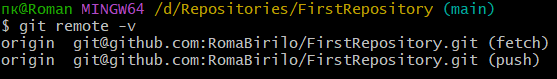
* + 1. **Тэги и стешинг**
* git tag “name” задает имя коммиту
* git tag –list показывает список всех тэгов
* git push --tags отправляет тэги в репозиторий
* git checkout “name” позволяет переключаться между коммитами с тэгом

Стешинг позволяет сохранить свой проект, который еще не готов к кмоммиту во временное хранилище.

* git stash save “discription” сохраняет проект с изменеиями
* git stash list показывает список стешей
* git stash pop возвращает стеш обратно и удаляет его
* git stash apply только возвращает стеш
* git stash drop удаляет все стеши
  + 1. **Команда git remote**

git remote **—** связывает локальный проект с внешним репозиторием.

git remote -v позволяет посмотреть с каким сервером мы работаем



В случае потери удаленного репозитория командой git remote remove “name” удаляем старый репозиторий.



Затем добавляем новый репозиторий командой git remote add “name” “url”.



После чего командой git push --set-upstream origin main переносим все коммиты в новый репозиторий.

