Начало Cd /d/

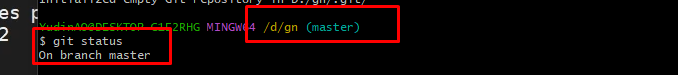
Это команда перехода в нужную папку cd /d/git/pkg/xml

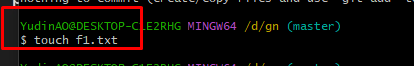
Команды ll ls помогут узнать какие файлы есть в локальной папке

# Git status узнать статус файлов к репозиторию в интернете. Не в локальном , а в глобальном

Чтобы папка начала отслеживаться гитом надо написать

Git init находясь в этой папке





# Добавить любой файл в индекс. В то, зачем будут следить.

Git add . добавить все файлы

Git add file.txt добавить один файл

Если файл без разрешения надо написать

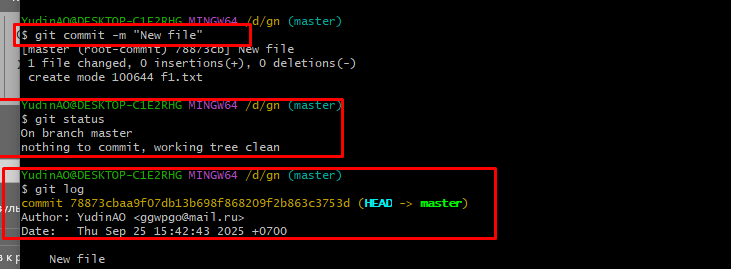
Git add file\_name

# Git commit

Сперва гит адд

Потом гит комиит

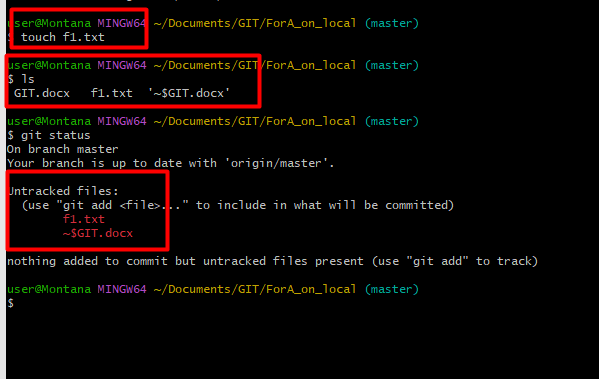
Потом гит пуш из локального репа в глобальный



Закомитил. смотрю логи.

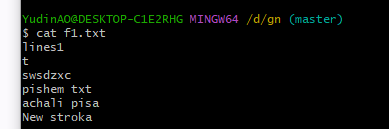
# Добавить файл в репозиторий

Touch name.txt



# Увидеть содержимое файла

Cat f1.txt



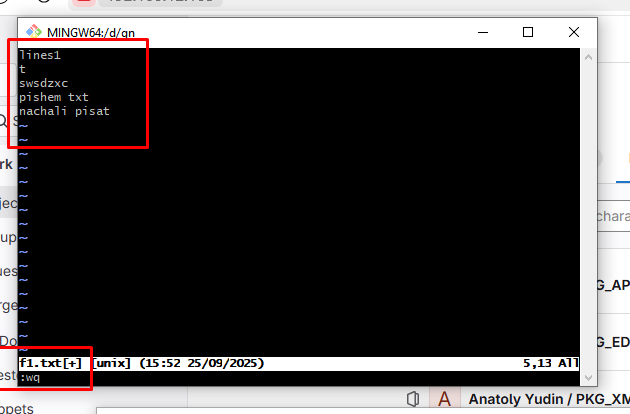
# Заглянуть внутрь файла и изменить его vi f1

Vi f1.txt

Что-то написать в файле

Писать английскими буквами спева нажали esc

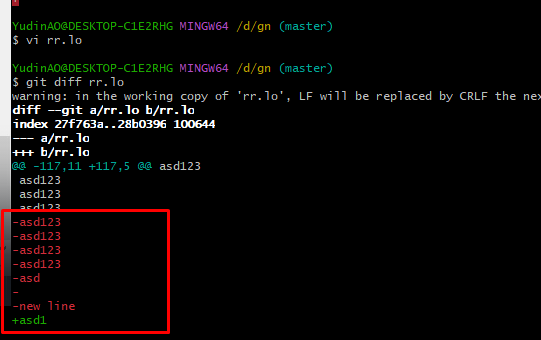
# :wq сохранить изменений



# Git diff изменения с последнего коммита

Т.е. сперва сделал изменения. Вышел из файла, потом написал гит диф и увидел какие изменения получились. Потом написал адд и коммит.

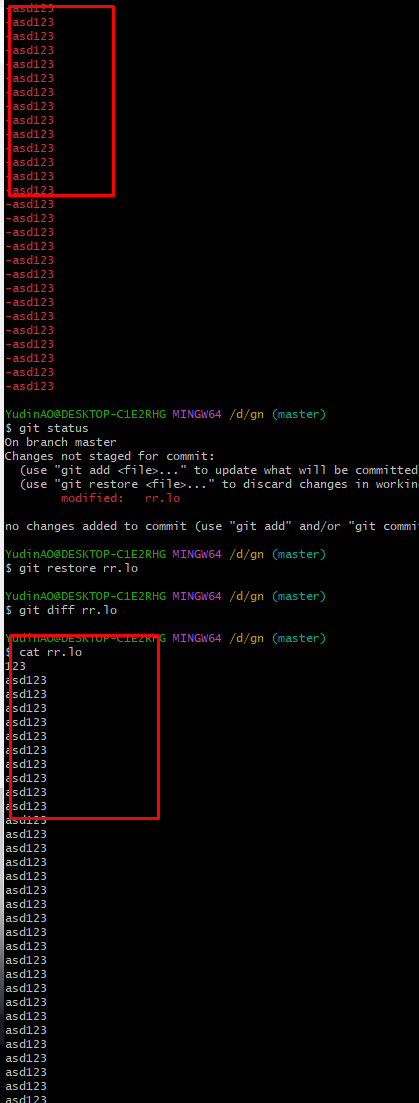
После коммит не видно изменений в файле.



# Git restore



Откатит все сделанные изменения до внесения коммита.



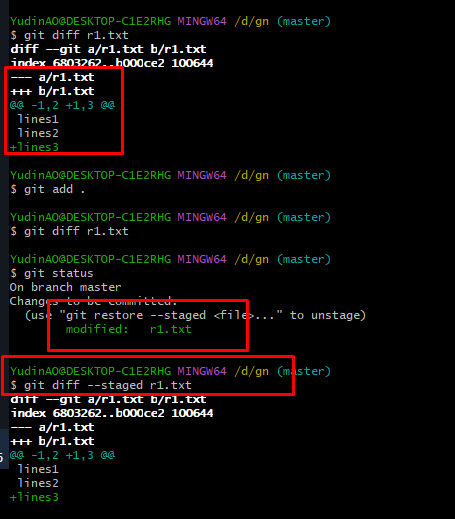
Я удалил столько строчек, потом написал рестор и все вернулось на место и гит дифф не показывает ни каких изменений.



После git add тоже диф ничего не покажет



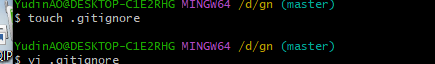
Чтобы увидеть эти изменения после гит адд надо написать гит дифф --стаджед



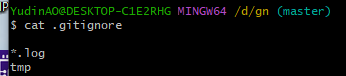
# Git restore –stager r1.txt вернет после git add все изменения в файле.

Это вернет на прошлую стадию, до внесения адд. А потом еще пишешь git resotre r1.txt и вернет весь текст, который изменили

# Git ignore не заносить и не следить за этим файлом гиту

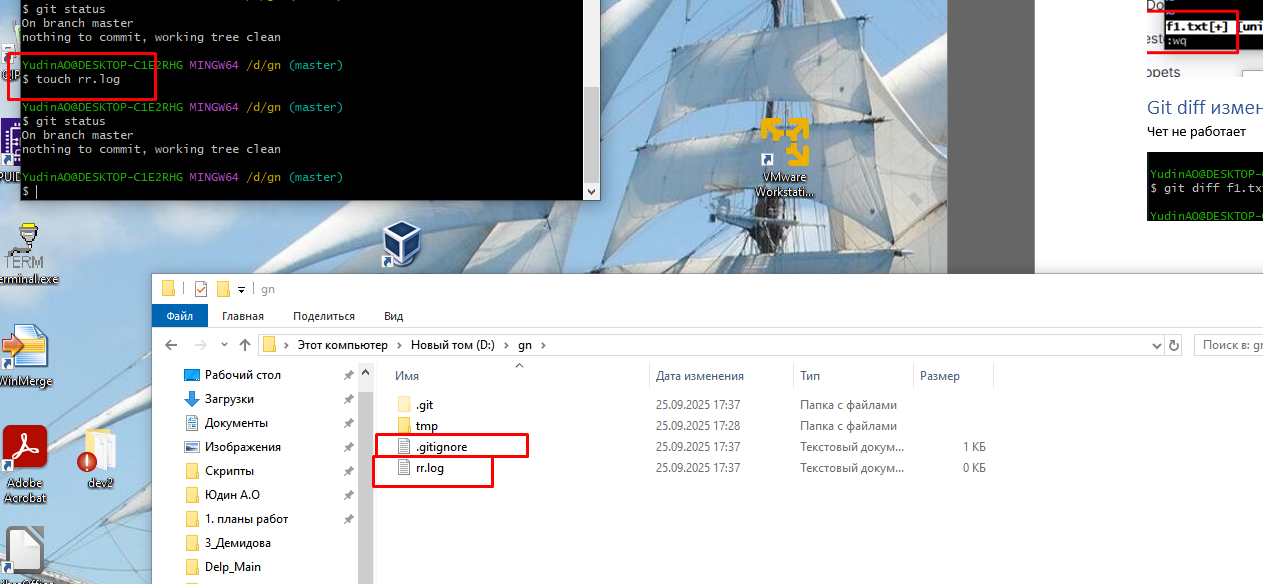


Создал такой файл и прописал в нем исключения



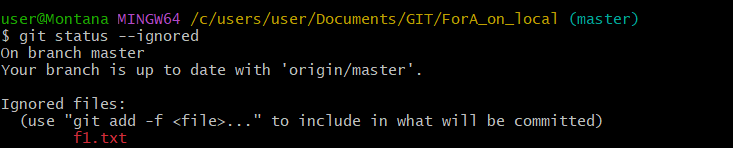
Все на окончания лог и тмп

Сделали адд и гит коммит

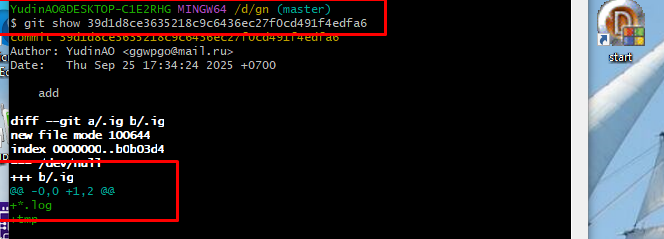


Добавил файл rr.log и его гит не видит и не требует коммита или адд

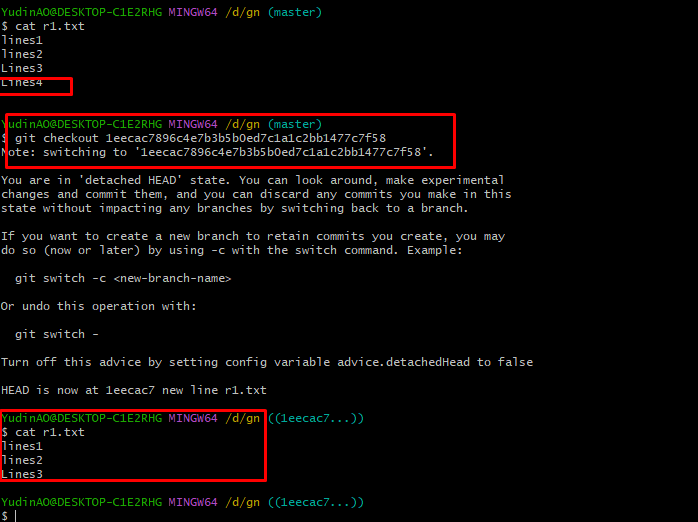
# Отобразить все игнорируемые файлы



# Git show 39d1d8ce3635218c9c6436ec27f0cd491f4edfa6 айди коммита покажет что было сделано в этом коммитн

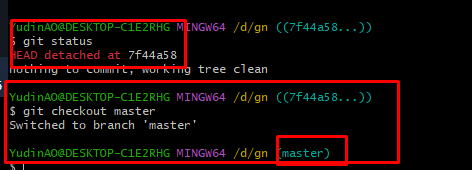


# Git checkout # коммита чтобы увидеть состояние файлов на нужный коммит. Мы не откатываем назад файл, а просто смотрим, как выглядел файл на нужный коммит

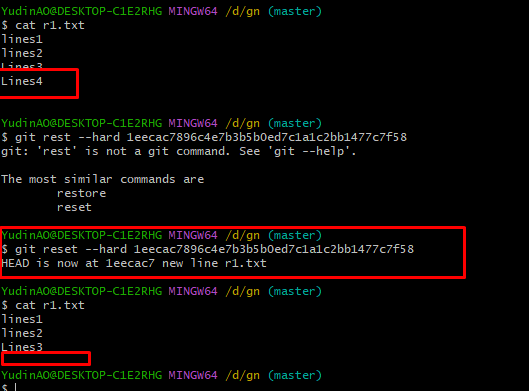


Чтобы вернуться надо написать git checkout и # последний коммит.

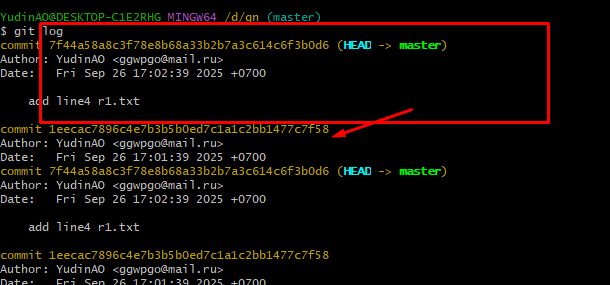
Также хэд нужно вернуть в мастер

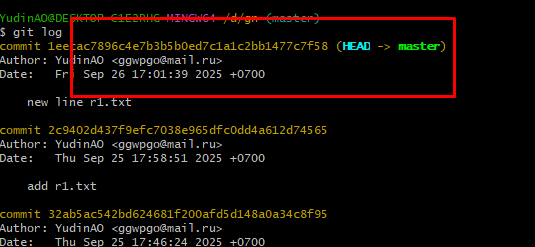


# Git reset –hard #коммита Чтобы полноценно вернутся на нужный коммит нужно

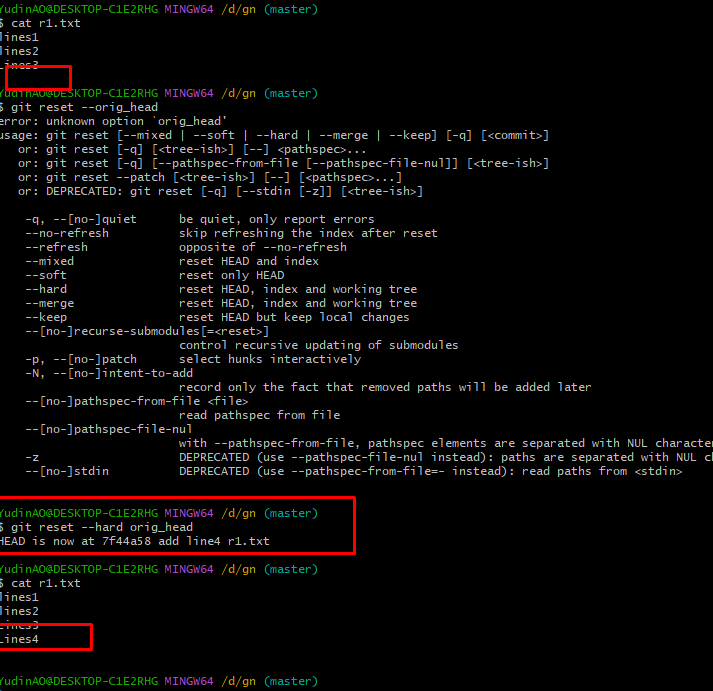


Также коммит удалился

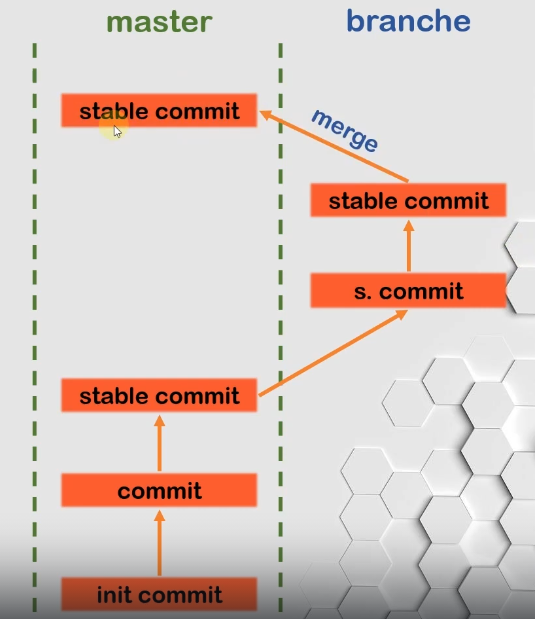




# Git reset –hard orig\_head чтобы вернуться к самому последнему коммиту, если неверно откатились в Git reset –hard #коммита



# Разделение на ветки



# Git branch узнать какие ветки

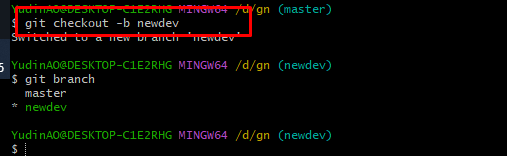
# Git branch DEV создать ветку DEV

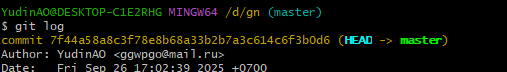
Git switch dev и git checkout dev две команды по переключениям в ветках



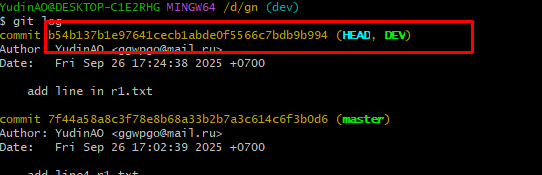
# Git checkout -b namebranch

Создать и переключится в ветку сразу



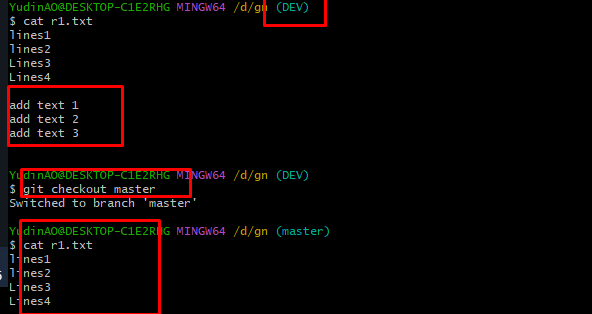


Последний коммит в мастере/



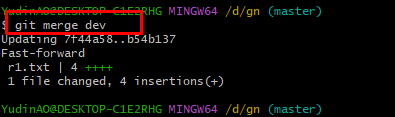
Тут видна разница в коммитах. Последний коммит в деве. Он не отобразиться в мастере

Разница в содержимом в файле, на разных ветках

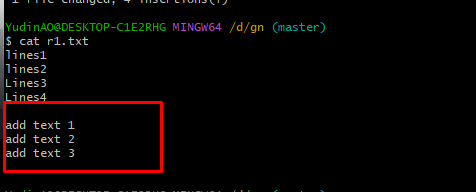


# Merge git merge dev

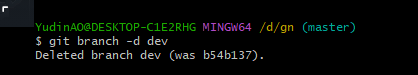
Надо зайти в ветку, в которую хотим залить данные. Допустим в мастер из дева код залить, то заходим в мастер.



Получается в одном файле были изменения и теперь эти изменения есть в мастер ветке, а их не было

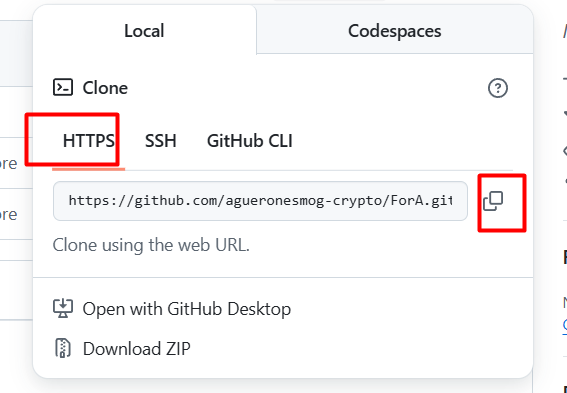


# Git branch -d dev Удалить ветку



# Связать локальный и удаленный репозиторий

Git remote add origin(источник всегда пишется этот псевдоним) и ссылка из гитхаба HTTPS



# Вывести список ssh ключей



# Узнать с каким удаленным репозиторием связан локальный git remote -v

