1. Просмотр истории коммитов

git log

Пример вывода:

commit f639bbde866f63c9d9dbbcae6734b243206cf4b6 (HEAD -> master)

Author: MrZhr4ik <starchik8116@gmail.com>

Date: Thu Jun 5 13:47:56 2025 +0300

Добавлено расписание на среду

commit 4bf2d4df09e9483b9ed2d06ec3384fcaa39de761

Author: MrZhr4ik <starchik8116@gmail.com>

Date: Thu Jun 5 13:42:35 2025 +0300

Добавлено расписание на вторник

commit d5af08f461d0828ad28ae76dd1e14f58047e8c59

Author: MrZhr4ik <starchik8116@gmail.com>

Date: Thu Jun 5 13:36:26 2025 +0300

Добавлено начальное расписание

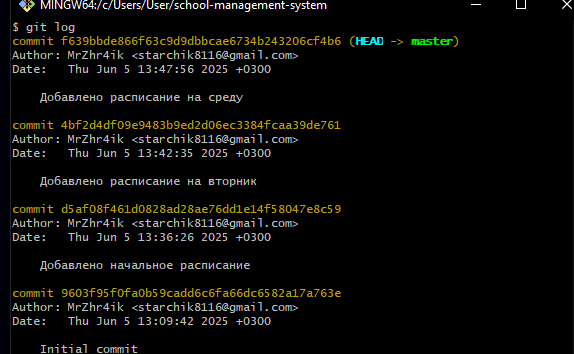
commit 9603f95f0fa0b59cadd6c6fa66dc6582a17a763e

Author: MrZhr4ik <starchik8116@gmail.com>

Date: Thu Jun 5 13:09:42 2025 +0300

Initial commit

Таким образом, Git позволяет отслеживать кто , когда и что добавил или изменил.



2. Краткий формат истории

git log --pretty=oneline

Вывод:

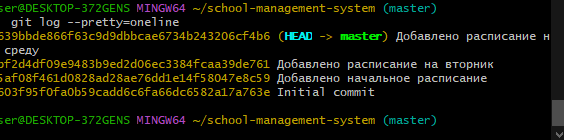
f639bbde866f63c9d9dbbcae6734b243206cf4b6 (HEAD -> master) Добавлено расписание

а среду

4bf2d4df09e9483b9ed2d06ec3384fcaa39de761 Добавлено расписание на вторник

d5af08f461d0828ad28ae76dd1e14f58047e8c59 Добавлено начальное расписание

9603f95f0fa0b59cadd6c6fa66dc6582a17a763e Initial commit



Такой формат удобен для быстрого просмотра истории без лишних деталей

3. Фильтр по времени

git log --since="7 days ago"

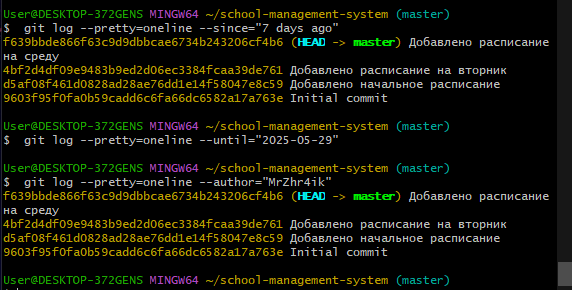
git log --until="2025-05-29"

Эти команды позволяют находить нужные коммиты за определённый период

4. По автору

git log --author="MrZhr4ik"

Это позволяет фильтровать только свои изменения, если работа ведётся в команде



5. По содержимому

git log -S "вторник"

Найдет все коммиты, где была добавлена информация о вторнике

6. Полный лог с деталями

git log --all --pretty=format:"%h %cd | %s%d [%an]" --graph --date=short

Пример:

$ git log --all --pretty=format:"%h %cd | %s%d [%an]" --graph --date=short

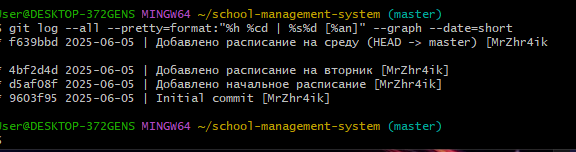
\* f639bbd 2025-06-05 | Добавлено расписание на среду (HEAD -> master) [MrZhr4ik

]

\* 4bf2d4d 2025-06-05 | Добавлено расписание на вторник [MrZhr4ik]

\* d5af08f 2025-06-05 | Добавлено начальное расписание [MrZhr4ik]

\* 9603f95 2025-06-05 | Initial commit [MrZhr4ik]



История выглядит понятно и структурированно

7. Создание алиасов (псевдонимов)

Создадим короткие команды для частых действий:

git config --global alias.co checkout

git config --global alias.ci commit

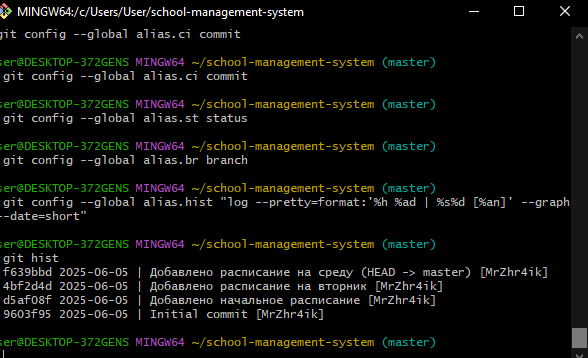
git config --global alias.st status

git config --global alias.br branch

git config --global alias.hist "log --pretty=format:'%h %ad | %s%d [%an]' --graph --date=short"

Теперь можешь использовать:

git hist

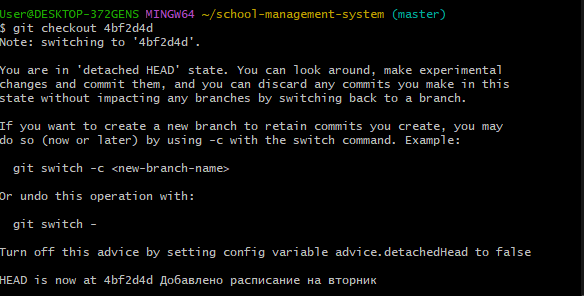


Алиасы значительно облегчают работу с Git

8. Откат к предыдущей версии

Если нужно вернуться к более раннему состоянию:

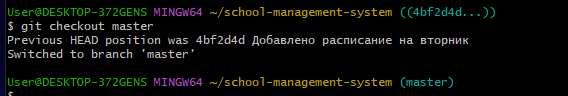
git checkout 4bf2d4d



Эта команда переключает тебя на коммит с описанием «Добавлены уроки на вторник»

Чтобы вернуться к последнему коммиту:

git checkout master



9. Отмена изменений

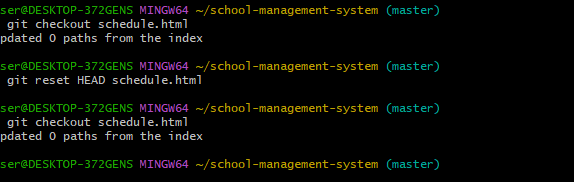
Если изменения не были проиндексированы:

git checkout schedule.html

Если уже индексировал:

git reset HEAD schedule.html

git checkout schedule.html



Эти команды помогают быстро отменить правки до коммита

10. Отмена последнего коммита

Если нужно откатить последнее изменение, но оставить файлы:

git revert HEAD

Команда создаёт новый коммит с обратными изменениями

11. Создание тегов

Для маркировки важных версий можно создать тег:

git tag v1.0

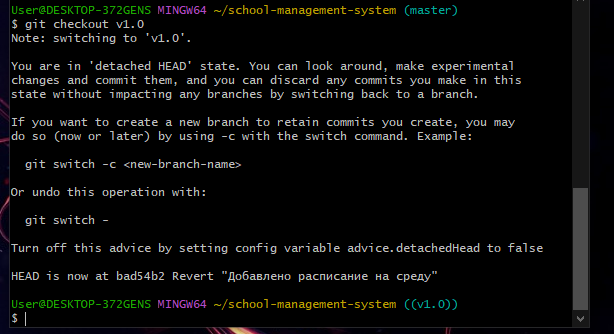
git push origin v1.0

Тег v1.0 будет доступен в репозитории GitHub

12. Переключение между версиями через теги

После создания тега можно переключаться между версиями:

git checkout v1.0



Это удобно при тестировании или восстановлении старых версий