Audion Hub Manager — Git без заучивания команд

English · О программе · Руководство · Консольная команда

Содержание

- Четыре места, где живёт работа
- Слова Git простыми словами
- Риск и режим кнопки
- Канон по разделам
- Частые положения и что делать
- Сообщение коммита
- Учебная лаборатория

Каждая кнопка окна запускает настоящую команду `git` и показывает её целиком. Под командой терминал пишет простыми словами, что она сделала и как её отменить. Все команды программы собраны в одну таблицу — канон Git (`system_core\core\git_canon.py`); таблицы ниже написаны из него и всегда совпадают с тем, что делает окно.

Четыре места, где живёт работа



Раздел ИЗМЕНЕНИЯ показывает эти четыре места полосой над списком файлов: сколько изменено, сколько лежит в индексе, есть ли коммиты, что не отправлено. Нажатие на место показывает только его файлы или открывает его раздел.

Место	Что там	Туда	Обратно
рабочая папка	файлы, как они лежат на диске	правка файла	`git restore` — вернуть к виду из индекса
индекс	то, что войдёт в следующий коммит	`git add` (F5)	`git restore --staged` (F6)
история	коммиты этого компьютера	`git commit` (КОММИТ)	`git reset --soft HEAD~1` — убрать последний
сервер	коммиты, которые видят другие	`git push`	отправленное отменяют `git revert`, а не переписыванием

Слова Git простыми словами

Слово	Простыми словами
репозиторий	папка с историей: рядом с файлами лежит скрытая `.git`
коммит	запись в истории: какие файлы стали какими, кто и зачем
индекс (staging)	стол перед коммитом: на него кладут то, что войдёт в следующую запись
ветка	черновик: отдельная линия работы, которая не мешает другим
`main`	чистовик: главная ветка, из которой выпускают
слияние (merge)	принять черновик в чистовик
конфликт	две ветки поменяли одно место по-разному; Git просит выбрать
remote, `origin`	сервер, куда отправляют и откуда забирают; `origin` — главный
push / pull / fetch	отправить свои коммиты / забрать чужие и дописать / только узнать, что нового
тег	именная метка версии на коммите: `projection-v0.1.2`
stash	ящик для правок, которые мешают перейти на другую ветку
HEAD	где вы сейчас: коммит, на котором стоит рабочая папка
reflog	журнал перемещений HEAD: в нём находятся даже «потерянные» коммиты

Слово	Простыми словами
предложение правок (PR, MR)	просьба принять ветку в `main`; делается на сайте GitHub, GitLab или Forgejo

Риск и режим кнопки

У каждой кнопки точка риска и подсказка с командой, её смыслом и способом отмены.

Риск	Значит
читает	ничего не меняет
меняет	меняет только этот репозиторий
сеть	говорит с сервером
опасно	может потерять работу или переписать историю — окно спросит

Режим	Значит
сразу	команда идёт по нажатию
спросит	сначала вопрос: команда целиком, что сейчас не закоммичено и как отменить; `reset --hard` подтверждается вводом слова RESET
заготовка	команда встаёт в строку терминала с отмеченными местами; её дописывают и запускают сами

Переключатель ОБЪЯСНЯТЬ в заголовке окна включает строку под каждой командой терминала: что она сделала и как её отменить. Когда станет привычно, его выключают.

Канон по разделам

Места в командах: `‹path›` и `‹paths›` — файлы, `‹commit›` — коммит, `‹branch›` — ветка, `‹new_branch›` — имя новой ветки, `‹remote›` — имя сервера, `‹url›` — его адрес, `‹message›` — сообщение, `‹tag›` — метка версии, `‹range›` — цепочка коммитов, `‹dest›` — папка клона, `‹bundle›` — файл резерва. Окно подставляет их из выбранного: файла, коммита, ветки, полей.

ИЗМЕНЕНИЯ

Команда	Что делает	Как отменить	Риск
`git status --short --branch`	Что изменено, что в индексе и насколько ветка впереди или позади remote.	Ничего не меняет.	читает · сразу
`git diff -- ‹path›`	Чем файл в рабочей папке отличается от индекса: что ещё не подготовлено к коммиту.	Ничего не меняет.	читает · сразу
`git diff --cached -- ‹path›`	Что из файла уже лежит в индексе и войдёт в следующий коммит.	Ничего не меняет.	читает · сразу
`git diff HEAD -- ‹path›`	Все изменения файла против последнего коммита — и в индексе, и нет.	Ничего не меняет.	читает · сразу
`git blame -- ‹path›`	Кто и в каком коммите написал каждую строку файла.	Ничего не меняет.	читает · сразу
`git add -- ‹paths›`	Кладёт файлы в индекс — «на стол перед коммитом». Следующий коммит возьмёт именно то, что в индексе.	Отменить: git restore --staged -- ‹файлы›	меняет · сразу
`git add -- ‹paths›`	Кладёт в индекс все изменения слоя, но только файлы, которые разрешает профиль проекта: логи, сборки и секреты не попадут.	Отменить: git restore --staged -- .	меняет · сразу
`git restore --staged -- ‹paths›`	Снимает файлы со стола: они остаются изменёнными в рабочей папке, но в коммит не войдут.	Отменить: git add -- ‹файлы›	меняет · сразу
`git rm --cached -- ‹paths›`	Снимает файлы со стола, пока в репозитории нет ни одного коммита: из индекса они уходят, в рабочей папке остаются.	Отменить: git add -- ‹файлы›	меняет · сразу
`git restore -- ‹paths›`	Возвращает файлы в рабочей папке к виду из индекса. Незакоммиченные правки этих файлов пропадут.	Отменить нельзя: этих правок нет ни в одном коммите.	опасно · спросит

Команда	Что делает	Как отменить	Риск
`git commit -m ‹message›`	Записывает в историю всё, что лежит в индексе, одним снимком с сообщением.	Отменить последний: git reset --soft HEAD~1 (файлы вернутся в индекс).	меняет · сразу
`git commit --only -m ‹message› -- ‹paths›`	Коммит только отмеченных файлов, что бы ни лежало в индексе.	Отменить: git reset --soft HEAD~1	меняет · сразу
`git commit --amend -m ‹message›`	Переписывает последний коммит: новое сообщение и всё, что сейчас в индексе. Если коммит уже отправлен — история разойдётся с remote.	Отменить: git reset --soft HEAD@{1}	опасно · спросит
`git tag -a ‹tag› -m ‹message›`	Ставит именную метку версии на текущий коммит. Push с --follow-tags отправит её вместе с веткой.	Отменить: git tag -d ‹тег›	меняет · сразу

ИСТОРИЯ

Команда	Что делает	Как отменить	Риск
`git log --oneline --decorate -20`	Последние 20 коммитов по одной строке.	Ничего не меняет.	читает · сразу
`git log --graph --oneline --decorate --all -50`	Граф всех веток: где разошлись и где сошлись.	Ничего не меняет.	читает · сразу
`git log --date=short --pretty=format:%h%x09%ad%x09%an%x09%s -40 -- ‹path›`	История одного файла: какие коммиты его меняли.	Ничего не меняет.	читает · сразу
`git reflog --date=local -20`	Журнал перемещений HEAD — спасательный круг: здесь видны даже «потерянные» коммиты.	Ничего не меняет.	читает · сразу

Команда	Что делает	Как отменить	Риск
`git show --stat ‹commit›`	Что изменил коммит: сообщение, автор, файлы.	Ничего не меняет.	читает · сразу
`git tag -n --sort=-creatordate`	Все теги с подписями, новые сверху.	Ничего не меняет.	читает · сразу
`git tag -a ‹tag› ‹commit› -m ‹message›`	Ставит метку версии на выбранный коммит.	Отменить: git tag -d ‹тег›	меняет · сразу
`git switch -c ‹new_branch› ‹commit›`	Создаёт ветку на выбранном коммите и переходит на неё — так спасают «потерянный» коммит из reflog.	Удалить: git branch -d ‹ветка›	меняет · сразу
`git revert --no-edit ‹commit›`	Делает НОВЫЙ коммит, который отменяет изменения выбранного. История не переписывается — годится и для отправленных коммитов.	Отменить: git revert ‹новый коммит›	меняет · сразу
`git cherry-pick ‹commit›`	Копирует изменения выбранного коммита в текущую ветку новым коммитом.	При конфликте: git cherry-pick --abort	меняет · сразу
`git reset --soft HEAD~1`	Убирает последний коммит, а его файлы оставляет в индексе: можно переписать сообщение или разбить коммит.	Отменить: git reset --soft HEAD@{1}	опасно · спросит

Команда	Что делает	Как отменить	Риск
`git reset --hard ‹commit›`	Переносит ветку на выбранный коммит и стирает ВСЕ незакоммиченные изменения рабочей папки и индекса.	Коммиты найдутся в reflog; незакоммиченное пропадёт навсегда.	опасно · спросит

ВЕТКИ

Команда	Что делает	Как отменить	Риск
`git branch -vv`	Ветки, их пары на сервере и насколько они впереди или позади.	Ничего не меняет.	читает · сразу
`git switch ‹branch›`	Переключает рабочую папку на другую ветку. Незакоммиченные правки Git перенесёт или откажется, если они мешают.	Вернуться: git switch -	меняет · сразу
`git switch -c ‹new_branch›`	Создаёт ветку от текущего коммита и переходит на неё.	Удалить: git branch -d ‹ветка›	меняет · сразу
`git switch -`	Возвращает на предыдущую ветку.	Ещё раз — обратно.	меняет · сразу
`git log --left-right --graph --cherry-pick --oneline HEAD...‹branch›`	Какие коммиты есть только в текущей ветке (<) и только в выбранной (>).	Ничего не меняет.	читает · сразу
`git merge --no-ff --no-edit ‹branch›`	Сливает выбранную ветку в текущую отдельным коммитом слияния — в графе видно, откуда пришла работа.	При конфликте: git merge --abort	меняет · сразу
`git branch -d ‹branch›`	Удаляет слитую ветку. Неслитую Git удалить не даст.	Вернуть: git switch -c ‹ветка› ‹хэш из reflog›	меняет · сразу
`git stash push -u -m ‹message›`	Откладывает все незакоммиченные правки и новые файлы в «ящик» и очищает рабочую папку.	Вернуть: git stash pop	меняет · сразу

Команда	Что делает	Как отменить	Риск
`git stash pop`	Достаёт последние отложенные правки обратно в рабочую папку.	Отложить снова: git stash push	меняет · сразу
`git stash list`	Что лежит в ящике отложенного.	Ничего не меняет.	читает · сразу
`git rebase -i ‹range›`	Переписывает цепочку коммитов: склеить, переставить, переименовать. Только для неотправленных коммитов.	Во время: git rebase --abort; после: через reflog.	опасно · заготовка
`git merge --abort`	Прерывает слияние с конфликтами и возвращает всё как было до него.	—	меняет · сразу
`git cherry-pick --abort`	Прерывает перенос коммита с конфликтами и возвращает всё как было до него.	—	меняет · сразу

СИНХРОНИЗАЦИЯ

Команда	Что делает	Как отменить	Риск
`git fetch --all --prune`	Скачивает новости со всех remote, не трогая ваши ветки и файлы. Самая безопасная сетевая команда.	Ничего вашего не меняет.	сеть · сразу
`git fetch --prune ‹remote›`	Узнаёт, что нового на одном сервере, не трогая ваши ветки и файлы.	Ничего вашего не меняет.	сеть · сразу
`git pull --ff-only ‹remote› ‹branch›`	Забирает новые коммиты с сервера, только если их можно просто дописать сверху, без слияния.	Откажет, если ветки разошлись: тогда — ВЕТКИ, слияние.	сеть · сразу
`git push --follow-tags ‹remote› ‹branch›`	Отправляет коммиты текущей ветки и теги на сервер, в её пару. Если имена разные, команда называет обе: ‹ветка›:‹пара›.	Отправленное отменяют новым коммитом (revert), а не переписыванием.	сеть · сразу

Команда	Что делает	Как отменить	Риск
`git push -u ‹remote› ‹branch›`	Отправляет новую ветку и связывает её с парой на сервере: дальше push и pull без имён.	Удалить на сервере: git push ‹remote› --delete ‹ветка›	сеть · сразу
`git push --follow-tags ‹remote› ‹branch›`	Отправляет ветку и теги по очереди во все remote репозитория.	—	сеть · сразу
`git remote -v`	Куда репозиторий умеет отправлять и откуда забирать.	Ничего не меняет.	читает · сразу
`git remote add ‹remote› ‹url›`	Прописывает в репозиторий remote из сохранённого списка.	Убрать: git remote remove ‹имя›	меняет · сразу
`git remote set-url ‹remote› ‹url›`	Меняет адрес remote, который уже есть в репозитории.	Вернуть: git remote set-url ‹имя› ‹прежний адрес›	меняет · сразу
`git remote set-url --add --push ‹remote› ‹url›`	Один remote, несколько адресов отправки: забирать из одного, отправлять в несколько зеркал.	Убрать: git remote set-url --delete --push ‹remote› ‹адрес›	меняет · сразу
`ssh -T -o BatchMode=yes git@github.com`	Проверяет, пускает ли GitHub по вашему SSH-ключу.	Ничего не меняет.	сеть · сразу
`ssh -T -o BatchMode=yes git@gitlab.com`	Проверяет, пускает ли GitLab по вашему SSH-ключу.	Ничего не меняет.	сеть · сразу
`gh run list --limit 15`	Последние прогоны GitHub Actions.	Ничего не меняет.	сеть · сразу
`gh run watch`	Следит за идущим прогоном GitHub Actions до конца.	Ничего не меняет.	сеть · сразу
`glab ci status`	Состояние конвейера GitLab CI.	Ничего не меняет.	сеть · сразу

ПРОЕКТЫ

Команда	Что делает	Как отменить	Риск
`git clone ‹url› ‹dest›`	Скачивает репозиторий целиком в новую папку.	Удалить папку.	сеть · сразу

ОБСЛУЖИВАНИЕ

Команда	Что делает	Как отменить	Риск
`git init -b ‹branch›`	Создаёт пустой репозиторий (папку .git) в выбранном слое; первая ветка — ветка проекта из реестра, обычно main.	Удалить папку .git.	меняет · сразу
`git rev-parse --show-toplevel`	Корень репозитория, в котором идут команды.	Ничего не меняет.	читает · сразу
`git bundle create ‹bundle› ‹branch› --tags`	Вся история ветки и теги одним файлом — резерв без сервера. Поднимается командой git clone ‹файл›.	Удалить файл.	меняет · сразу
`git clean -nd`	Показывает, какие неотслеживаемые файлы удалил бы git clean. Сам ничего не удаляет.	Ничего не меняет.	читает · сразу
`git gc`	Уплотняет базу репозитория. Безопасно.	—	меняет · сразу
`git fsck --full`	Проверяет целостность базы репозитория.	Ничего не меняет.	читает · сразу
`git config --list --show-origin`	Все настройки Git и из какого файла каждая.	Ничего не меняет.	читает · сразу
`git config --show-origin --get-regexp user\.`	Под каким именем и почтой будут подписаны коммиты.	Ничего не меняет.	читает · сразу

Частые положения и что делать

Сохранить работу. ИЗМЕНЕНИЯ: отметить файлы пробелом, F5 — в индекс, внизу тема коммита, КОММИТ (Ctrl+Enter). Файлы вне профиля проекта в индекс не идут — так в историю не

попадают сборки, логи и секреты.

Закоммитил не то, ещё не отправил. ИСТОРИЯ → «убрать последний (soft)»: коммит уходит, файлы остаются в индексе; поправить и закоммитить снова. Или галочка «исправить прошлый (--amend)» в блоке КОММИТ.

Испортил файл, нужен вид из последнего коммита. ИЗМЕНЕНИЯ → «вернуть» (F8). Окно спросит: правки этого файла пропадут навсегда.

Плохой коммит уже на сервере. ИСТОРИЯ → выбрать коммит → revert. Появится новый коммит, который отменяет изменения выбранного; история не переписывается, другие ничего не теряют.

Потерял коммит после reset. ИСТОРИЯ → вкладка REFLOG, выбрать коммит, вписать имя ветки внизу → «ветка отсюда».

Слияние остановилось на конфликте. В ИЗМЕНЕНИЯХ файлы с буквой U: поправить их, F5 — в индекс, КОММИТ. Передумали — ВЕТКИ → «если застряли» → `git merge --abort`, всё вернётся как было до слияния.

pull отказал: ветки разошлись. ВЕТКИ: выбрать ветку сервера (`origin/main`) → «влить в текущую», разобрать конфликт, если будет, затем PUSH. Программа нарочно забирает только перемоткой (`pull --ff-only`): слияние вы делаете сами и видите.

Правки мешают перейти на другую ветку. ВЕТКИ → «отложить» (stash), перейти, поработать, вернуться, «достать».

Выпустить версию. Блок КОММИТ: серия (`projection`), окно предложит следующий номер по тегам серии; patch, minor или major; галочка «поставить тег». После коммита — PUSH: `--follow-tags` отправит тег вместе с веткой.

Резерв без сервера. ОБСЛУЖИВАНИЕ → `git bundle create`: вся история ветки и теги одним файлом. Поднимается командой `git clone ‹файл›`.

Новая ветка для задачи. ВЕТКИ: имя внизу (`feature/отчёт`) → «создать и перейти». Первый push — `push -u`: он свяжет ветку с парой на сервере, дальше push и pull без имён.

Сообщение коммита

Блок КОММИТ собирает сообщение из частей и показывает его целиком в рамке:

```
тип(область): серия версия — тема
docs(readme): projection v0.1.3 — раздел о зеркале
```

Тип	Когда
docs	документы, тексты
code	код, новое умение
fix	исправление ошибки
ui	вид окна
test	проверки
chore	уборка, сборка, зависимости
audit	разбор и проверка

Область — часть проекта (`readme`, `mirror`, `ui`), можно пусто. Серия и версия необязательны: с ними сообщение несёт номер, а галочка «поставить тег» ставит метку `серия-версия` на этот коммит. Номер окно предлагает следующий по тегам серии; patch — исправление (0.1.1 → 0.1.2), minor — новое умение (0.1 → 0.2), major — несовместимое (0 → 1).

Учебная лаборатория

Десять маленьких репозиториев в известных состояниях. Их строит ПРОЕКТЫ → УЧЕБНАЯ ЛАБОРАТОРИЯ → «создать заново» или `hub_cli.cmd lab create --replace --register`: прежние уходят в Корзину, новые вносятся в реестр (`lab_01` … `lab_10`). Авторы и даты в них постоянные, поэтому коммиты получают те же хэши на любой машине. Ломайте смело.

	Репозиторий	Что в нём	Что попробовать
01	first-commit	файлы, репозитория ещё нет	`git init`, add, commit
02	index	изменены, новые, удалён, переименован, файл наполовину в индексе	status, три вида diff, add, `restore --staged`, restore, коммит части файлов
03	branch-merge	ветка `feature/login` позади `main`, `fix/typo` уже слита	switch, `switch -c`, `switch -`, сравнить, `merge --no-ff`, `branch -d`

	Репозиторий	Что в нём	Что попробовать
04	conflict	слияние остановилось на конфликте в `style.css`	разобрать конфликт или `merge --abort`
05	behind-remote	свой «сервер» в `_servers`; у вас 2 неотправленных коммита, на сервере 1 новый	fetch, `pull --ff-only` (на `main` откажет, на `docs` пройдёт), push, `push -u`
06	stash	правка `notes.txt` мешает перейти на ветку `other`; один stash уже лежит	`stash push`, switch, `stash list`, `stash pop`
07	tags	теги `projection-v0.1.0` и `v0.1.1`, после них ещё коммит	следующая версия, `tag -a`, `tag -n`
08	lost-commit	коммит «step two» стёрт reset'ом	reflog, ветка от потерянного коммита
09	revert-reset	третий коммит сломал расчёт	revert, `reset --soft`, `reset --hard`
10	source-mirror	проект (`source`) и его GIT-КОПИЯ (`git-copy`) разошлись	план зеркала, применить, сверить, коммит GIT-КОПИИ

Тесты программы гоняют каждую команду канона на свежей копии лаборатории и проверяют, что из неё вышло; пять сетевых (`ssh -T`, `gh`, `glab`) — по форме команды.