Audion Hub Manager — руководство

English · О программе · Git-вики · Консольная команда · История

Содержание

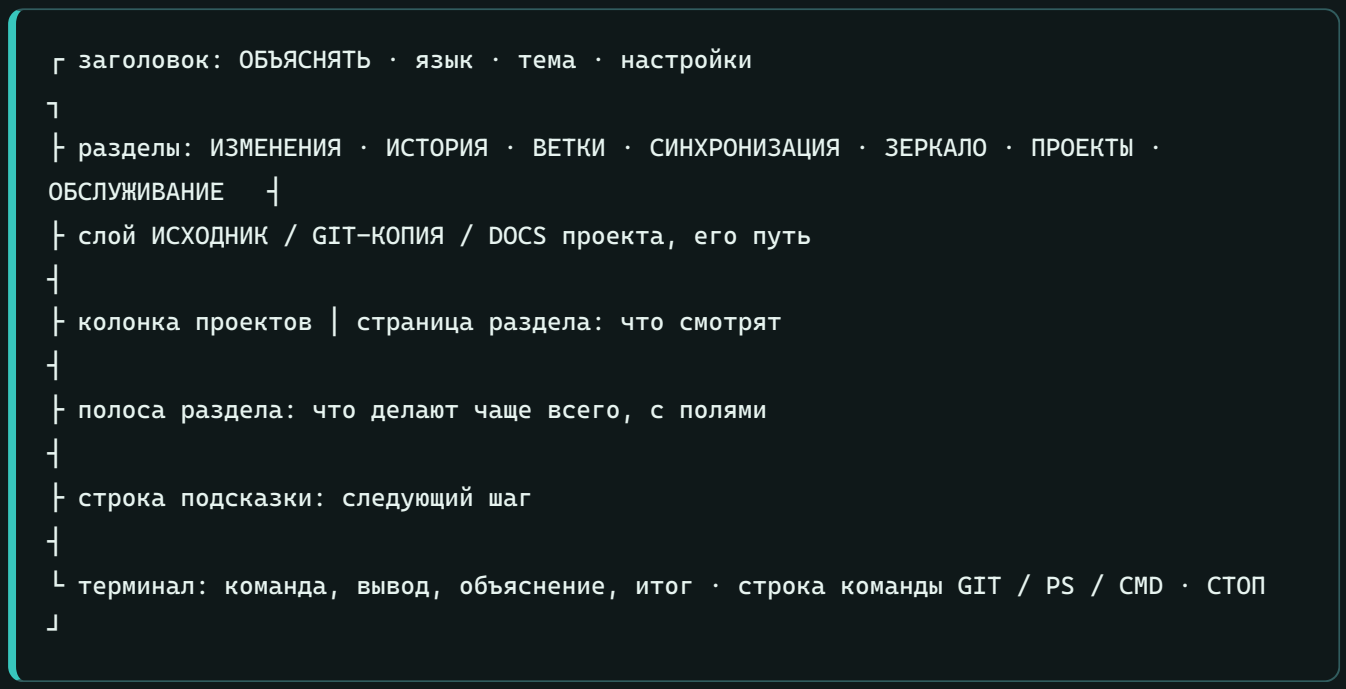
- Первый запуск
- Окно
- ИЗМЕНЕНИЯ
- ИСТОРИЯ
- ВЕТКИ
- СИНХРОНИЗАЦИЯ
- ЗЕРКАЛО
- ПРОЕКТЫ
- ОБСЛУЖИВАНИЕ
- Обычный день
- Клавиши
- Где что лежит

Первый запуск

`Start.exe` открывает окно. Если реестр проектов пуст, строка подсказки зовёт в ПРОЕКТЫ: там можно найти свои проекты в папке или построить учебную лабораторию. Если папка `runtime\` пуста, окно скажет, что консольная команда недоступна: окружение Python собирает `builder_main.cmd` → [01].

Чтобы сначала потренироваться: ПРОЕКТЫ → УЧЕБНАЯ ЛАБОРАТОРИЯ → «создать заново», затем `lab_01` в списке проектов.

Окно



Колонка проектов слева — проекты реестра. Щелчок по проекту делает его текущим: Git-команды идут к нему. Проект — запись реестра: она переключает весь комплект разом — ИСХОДНИК, GIT-КОПИЮ, DOCS, профиль зеркала и ветку по умолчанию. Квадрат у проекта отмечает его для работы над несколькими сразу, ALL и CLEAR отмечают все или ни одного; бейдж в шапке — «отмечено | всего».

Внизу колонки — **КОММИТ ОТМЕЧЕННЫХ**: тема одна на всех, и каждый отмеченный проект по очереди проходит контрольную точку в своей GIT-КОПИИ — зеркало ИСХОДНИК → GIT-КОПИЯ, сверка BLAKE3, проверка секретов, проверка путей профилем, add и commit. Каждая команда идёт в терминал, как с кнопки, с именем проекта в заголовке. Проект, где шаг сказал «нет», останавливается с причиной, остальные идут дальше; GIT-КОПИЯ без изменений — не ошибка. GIT-КОПИЮ без своего репозитория окно пропускает: её первый коммит делают по шагам, в ЗЕРКАЛЕ и ИЗМЕНЕНИЯХ. СТОП останавливает идущую команду, а следующие проекты не начинаются. Итог — строкой: сделано, без изменений, остановлено.

Справа от бейджа — **корень проектов**, папка, где лежат живые проекты: на неё смотрят панель папок и поиск проектов, и в ней лежит реестр проектов `hub_projects.json` — у каждого корня свой. Меню корня:

Пункт	Что
Сделать корнем	папка, выбранная в панели папок раздела ПРОЕКТЫ, становится корнем; колонка показывает её реестр

Пункт	Что
Вернуть корень по умолчанию	папка рядом с программой и её реестр
Импорт реестра из файла…	проекты из файла дописываются в реестр корня: пути читаются от места файла и пишутся от корня; проекты, что в реестре уже есть, остаются как есть
Экспорт реестра в файл…	реестр корня — в файл, каждый путь полностью: файл читается одинаково, куда бы он ни уехал

Выбор корня хранится для этой машины в `config\storage_layout.local.json`. Если у проектов нового корня GIT-КОПИЯ лежит не в корне GIT-КОПИЙ, терминал называет их строкой под командой. Шеврон «‹» сворачивает колонку, «›» у края страницы раскрывает её. В окне от 1550 точек колонка стоит рядом со страницей и раздвигает её, свёрнута она или нет — окно помнит; в окне уже — свёрнута и раскрывается поверх страницы, а закрывается выбором проекта, шевроном, Esc или щелчком мимо неё. Длинное имя или сообщение не обрывается многоточием, а гаснет к концу; целиком оно — в подсказке.

Слой — где идут Git-команды:

Слой	Что это
ИСХОДНИК	живой проект целиком, источник истины
GIT-КОПИЯ	чистое зеркало проекта по профилю, свой `.git`: чистая история для сервера
DOCS	слой для чтения: Markdown и тексты

Рядом со слоем — его путь и состояние: «N изм.», «без git», «нет папки», «внутри чужого» (папка лежит внутри другого репозитория, и команды ушли бы туда). Кнопки рядом открывают слой в Проводнике, в VS Code и во внешнем терминале с `git status`.

ОБЪЯСНЯТЬ — строка под каждой командой терминала: что она сделала и как её отменить. **Тема** — Audion Vault Dark (по умолчанию), Audion Code Dark, Audion Dark, Audion Light. **Настройки** — размеры текста окна и терминала, подтверждение `reset --hard` вводом слова RESET, пути к настройкам и журналам. **О программе** — номер, папки, канон Git, лицензия.

Терминал показывает каждую команду целиком, её вывод в цветах Git и итог: «Готово» или ошибку с кодом выхода. Строка команды работает в трёх режимах: GIT (пишут `status`, `log --oneline` или целиком `git …`), PS (PowerShell 7 программы), CMD (`cmd.exe` в UTF-8). Все три идут в корне репозитория выбранного слоя. Стрелки — прежние команды, список прежних — с

закреплением. Опасную команду, набранную руками, окно тоже спросит: Git-команду оно судит по её доводам и в строке PS или CMD (`reset --hard`, `push --force`, `restore`, `clean -f`, `branch -D`, `rebase`, `commit --amend`...), а `reset --hard` ждёт слова RESET, как с кнопки. Пока идёт одна команда, следующая ждёт очереди; СТОП останавливает идущую вместе со всем, что она запустила. Кнопки терминала: папка `logs`, весь текст в буфер обмена, очистить, развернуть на всё окно.

У каждой кнопки — точка риска (читает, меняет, сеть, опасно) и подсказка: команда, что она делает, как отменить.

ИЗМЕНЕНИЯ

Полоса над списком — четыре места Git: РАБОЧАЯ ПАПКА, ИНДЕКС, ИСТОРИЯ, СЕРВЕР, со счётчиками. Нажатие на рабочую папку или индекс оставляет в списке только их файлы, на историю и сервер — открывает раздел.

Список файлов — в двух видах:

- **ИЗМЕНЕНИЯ** — только то, что отличается от последнего коммита, группами: КОНФЛИКТЫ, В ИНДЕКСЕ, ИЗМЕНЕНЫ, НОВЫЕ;
- **ВСЕ ФАЙЛЫ** — папки слоя целиком, с буквой Git у изменённых: Enter — в папку, Backspace — наверх.

Пробел или Insert отмечает файл; меню отметок берёт все изменённые, все новые, все в индексе, `*.md` или снимает всё. Поле «найти» сужает список. Файл вне профиля проекта помечен «вне профиля»: в индекс и в GIT-КОПИЮ он не идёт.

Справа — РАЗНИЦА выбранного файла: НЕ В ИНДЕКСЕ, В ИНДЕКСЕ или ПРОТИВ HEAD; добавленное — зелёным, убранное — коралловым. Оттуда же — история файла и VS Code.

Блок **КОММИТ** собирает сообщение: ТИП (docs, code, fix, ui, test, chore, audit), ОБЛАСТЬ, ВЕРСИЯ (серия, номер, patch / minor / major, «поставить тег»), галочки «только отмеченные» и «исправить прошлый (--amend)». Сообщение целиком видно в рамке. Тему пишут в полосе внизу.

Полоса	Клавиша	Что
обновить	F2	`git status` в терминал
в индекс	F5	отмеченные файлы, без отметок — выбранный; файлы вне профиля не берёт

Полоса	Клавиша	Что
из индекса	F6	снять с индекса, правки остаются; до первого коммита — `git rm --cached`, сравнивать ещё не с чем
вернуть	F8	вернуть файл к виду из индекса — опасно, окно спросит
тема коммита		что сделано и зачем
КОММИТ	Ctrl+Enter	записать индекс в историю; перед записью окно проверяет файлы индекса профилем
push		отправить на сервер

ИСТОРИЯ

Вкладки страницы: ГРАФ (все ветки или только текущая), ФАЙЛ (история файла, выбранного в ИЗМЕНЕНИЯХ), ТЕГИ, REFLOG (журнал перемещений, где видны и «потерянные» коммиты). Поле «искать в сообщениях» оставляет коммиты с этим текстом в сообщении, авторе или хэше.

Выбранный коммит показан справа: автор, время, родители, файлы. Действия разложены по последствиям:

- НИЧЕГО НЕ МЕНЯЕТ — показать коммит;
- НОВЫЙ КОММИТ ПОВЕРХ — revert, cherry-pick в текущую ветку;
- ПЕРЕПИСЫВАЕТ ИСТОРИЮ — убрать последний (soft), сбросить сюда (hard): окно спросит.

Полоса: имя ветки или тега для выбранного коммита, «показать» (F3), «лента», reflog, «ветка отсюда», «тег здесь», revert.

ВЕТКИ

Страница: ЛОКАЛЬНЫЕ, НА СЕРВЕРЕ, ТЕГИ, ОТЛОЖЕНО (STASH). У выбранной ветки — её положение против пары на сервере («коммитов только у вас: 2», «ветки разошлись: нужно слияние») и действия: перейти, сравнить с текущей, влить в текущую, удалить, `push -u`. У ветки сервера — «влить в текущую»; чтобы работать в ней, её имя без `origin/` вписывают внизу и нажимают «создать и перейти» — Git свяжет новую ветку с серверной.

Блок «ЕСЛИ ЗАСТРЯЛИ»: `git merge --abort` и `git cherry-pick --abort` возвращают всё как было до слияния или переноса; `rebase -i` — заготовка для переписывания своих неотправленных коммитов.

Полоса: имя новой ветки, «создать и перейти» (F7), «назад», подпись отложенного, «отложить», «достать»; `git branch -vv`, `git stash list`.

СИНХРОНИЗАЦИЯ

Список remote репозитория: откуда забирает и куда отправляет каждый. Блок **НОВЫЙ REMOTE** собирает адрес: платформа (GitHub, GitLab, Codeberg, Forgejo, Gitea, свой сервер), вид (SSH — вход по ключу, HTTPS — через Git Credential Manager или `gh` / `glab`), имя remote, владелец, репозиторий; у своих серверов — адрес и порт SSH. Готовый адрес можно «прописать в репозиторий», добавить к `origin` вторым адресом отправки или «сохранить в remotes.json» — список программы, из которого его прописывают в любой репозиторий. Адрес с паролем или токеном внутри не сохраняется.

Блок **ДОСТУП**: «проверить всё» — Git, имя и почта коммитов, `gh`, `glab`, вход по SSH на GitHub и GitLab, адреса remote; кто подписывает коммиты; `ssh -T` на GitHub и GitLab; вход через GitHub CLI и GitLab CLI во внешнем терминале; Диспетчер учётных данных Windows. Пароли окно не спрашивает и не хранит.

Свой сервер Forgejo или Gitea. Личный токен доступа создают на сервере: Settings → Applications → Generate New Token. Окно проверяет токен запросом к серверу и отдаёт хранилищу Git (Credential Manager); в файлы программы и в журнал он не пишется. Дальше `git push` по HTTPS берёт его сам. Кнопки: «сервер отвечает?», «сохранить токен», «кто я», «мои репозитории», «забыть токен» (убирает с этой машины; на сервере токен действует, пока его там не отзовут).

CI: `gh run list`, `gh run watch`, `glab ci status`.

Полоса: «fetch всё» (F2), pull, «во все remote», PUSH со счётчиком неотправленного. pull забирает только перемоткой: если ветки разошлись, он откажет — тогда ВЕТКИ → слияние, потом PUSH. PUSH отправляет текущую ветку в её пару на сервере; если имена у них разные, команда называет обе (`topic:docs`).

ЗЕРКАЛО

Зеркало приводит GIT-КОПИЮ к ИСХОДНИКУ по профилю проекта. ИСХОДНИК при этом не меняется, `.git` копии защищён.

Блок ПРОФИЛЬ показывает, что профиль пропускает и что нет, способ сравнения и вид зеркала («точное зеркало: лишнее удаляется»). «план» (F2) строит таблицу: НОВЫЙ, ОБНОВИТЬ, УДАЛИТЬ, БЕЗ ИЗМЕНЕНИЙ, ВНЕ ПРОФИЛЯ — с путём и причиной; ничего не пишет. ПРИМЕНИТЬ выполняет план; с галочкой «сухой прогон» только показывает, что записал бы. «сверить BLAKE3» сравнивает ИСХОДНИК и GIT-КОПИЮ по суммам. Отчёты ложатся в `logs\<проект>\`.

После применения: переключить слой на GIT-КОПИЮ и закоммитить её в ИЗМЕНЕНИЯХ.

ПРОЕКТЫ

Слева — ПАПКИ, файловая панель этой машины. Сверху места: ДИСКИ, корень проектов, GIT-КОПИИ (`Hub Data`), DOCS, если папка есть. Enter или двойной щелчок — в папку, Backspace — наверх, поле «найти» сужает список. У папки видно, чем она приходится реестру — ИСХОДНИК, GIT-КОПИЯ или DOCS какого-то проекта, — или что в ней свой репозиторий (`git`), которого в реестре нет. Выбранная папка — строка под курсором, а если под ним `..` — папка, которую смотрят; её путь — в строке под списком.

РЕЕСТР — таблица проектов: ветка, состояние, GIT-КОПИЯ, слои. ЗАПИСЬ — папки ИСХОДНИКА, GIT-КОПИИ и DOCS и ветка по умолчанию выбранного проекта; кнопка у поля берёт в него папку, выбранную в панели. «Сохранить» пишет их в реестр корня проектов (пути внутри корня — от него, и корень переезжает вместе со своим реестром), «убрать из реестра» вынимает запись, а папки не трогает.

Полоса:

Кнопка	Что
статус всех (F2)	`git status` по всем проектам — в GIT-КОПИИ, если в ней свой `.git`, иначе в ИСХОДНИКЕ: чисто, изменено, ошибки
планы зеркал	план зеркала каждого проекта
секреты	проверка секретов и тяжёлых файлов во всех проектах
сверка	сверка ИСХОДНИКА и GIT-КОПИИ всех проектов по BLAKE3

Кнопка	Что
найти в папке	находит проекты в папке, выбранной в панели, и дописывает в реестр; окно сначала называет папку; папки не трогает
клонировать	адрес репозитория → клон в папку, выбранную в панели; окно сначала называет адрес и место; «в реестр» вносит его
workspace	общий `.code-workspace` для VS Code со всеми проектами
почистить реестр	убирает записи без папки ИСХОДНИКА и дубли; перед записью показывает, что уйдёт

«Найти в папке»: папка, которая сама проект (у неё есть `.git`, `README.md`, `pyproject.toml` и подобное), находит себя; общая папка без таких признаков — проекты в подпапках. GIT-КОПИЯ найденного проекта предлагается в папке `Hub Data` рядом с папкой программы; её путь правят в ЗАПИСИ.

УЧЕБНАЯ ЛАБОРАТОРИЯ — десять учебных репозиториев в `examples\lab`; «создать заново» строит их (прежние — в Корзину) и вносит в реестр, «открыть LAB.md» — что в каждом попробовать. Подробно — в Git-вики.

ОБСЛУЖИВАНИЕ

Блок	Что
ПРОВЕРКИ	«секреты» (F2) — похожие на секреты файлы (`.env`, `*.pem`), ключи и токены в тексте, бинарники больше 25 МБ, с пометкой, уйдёт ли файл с push; «сверка» зеркала; «хранилище» — корни программы, GIT-КОПИЙ, DOCS и проектов; движок хэшей BLAKE3; папка отчётов
ОТЧЁТЫ	итог последних проверок
РЕЗЕРВ	`git bundle create` — вся история ветки и теги одним файлом
ЗДОРОВЬЕ РЕПОЗИТОРИЯ	`git fsck --full`, `git gc`, что удалил бы `git clean`
НАСТРОЙКИ GIT	кто подписывает коммиты, все настройки с файлами, корень репозитория

Там же `git init` — для слоя, который ещё не репозиторий.

Обычный день

Одна машина. ИЗМЕНЕНИЯ → отметить → F5 → тема → КОММИТ → push.

Проект с GIT-КОПИЕЙ. ЗЕРКАЛО → план → ПРИМЕНИТЬ → слой GIT-КОПИЯ → ИЗМЕНЕНИЯ → отметить всё изменённое и новое → F5 → тема и версия → КОММИТ с тегом → PUSH.

Две машины.

```
машина A: коммит → тег → push
машина B: fetch → pull (перемоткой)
```

fetch только узнаёт, что нового, ничего вашего не трогая; pull дописывает новые коммиты, когда вы этого хотите.

Клавиши

Клавиша	Где	Что
F2	каждый раздел	главное чтение раздела: status, fetch, план, статус всех, секреты
F3	ИСТОРИЯ	показать коммит
F5 / F6 / F8	ИЗМЕНЕНИЯ	в индекс / из индекса / вернуть
F7	ВЕТКИ	создать ветку и перейти
Ctrl+Enter	ИЗМЕНЕНИЯ	КОММИТ
Пробел, Insert	список файлов	отметить
Enter, Backspace	ВСЕ ФАЙЛЫ	в папку, наверх
Enter, ↑ ↓	строка команды	выполнить, прежние команды
Esc		закрыть настройки или «О программе», снять фильтр

Где что лежит

Файл	Что
`<корень проектов>\hub_projects.json`	реестр проектов этого корня; без него программа заводит его: для корня по умолчанию — из `config\projects.example.json`, для своего корня — пустым
`config\storage_layout.local.json`	корень проектов, выбранный на этой машине
`config\projection_profiles.json`	профили зеркала
`config\remotes.json`	сохранённые remote, без паролей
`config\forgejo_hosts.json`	серверы Forgejo и Gitea этой машины, без токенов; без файла список пуст
`config\window_settings.json`	тема, язык, шрифты и размеры, раздел и слой, ОБЪЯСНЯТЬ, тип, область и серия коммита
`config\terminal_commands.json`	прежние команды строки терминала
`logs\`	журнал окна; `logs\<проект>\` — отчёты зеркала и проверок