АСТРА ИИ КОД

(ASTRA AI Code)

Руководство администратора

Версия 1.0.0

Москва
2026

СОДЕРЖАНИЕ

1. Общие положения

Раздел вводит базовую терминологию продукта АСТРА ИИ Код (ASTRA AI Code) (далее – AstraCode), применяемую во всём руководстве: компоненты системы, клиентские приложения и механизмы безопасности.

Область действия руководства. Настоящий документ описывает все три клиентских приложения продуктовой линейки АСТРА ИИ Код (ASTRA AI Code): АстраCLI (терминальный клиент), Plugin (расширение для IDE) и Desktop (настольное приложение). Все три клиента разделяют общее ядро — исполняемый файл на Rust и общий протокол внутреннего сервера обработки команд (app-server); различаются интерфейсом и способом распространения.

1.1. Термины и определения

Ниже приведены термины и определения, используемые во всём документе.

Термин	Определение
АСТРА ИИ Код (ASTRA AI Code)	Программный продукт — ИИ-агент для разработки кода, рассчитанный на локальное развёртывание. Агент исполняется на рабочей станции пользователя, обращается к выбираемой языковой модели и применяет изоляцию выполнения, разграничение прав и режимы подтверждения операций.
Ядро (агентский цикл)	Базовое ядро AstraCode, реализующее цикл «приём запроса → обращение к модели → выполнение инструментов → возврат результата» и управляющее состоянием Session → Task → Turn. Единое для всех трёх клиентов; поставляется как исполняемый файл astracode.
АстраCLI	Клиент продукта с текстовым (терминальным) интерфейсом (TUI). Единственный исполняемый файл; режимы: TUI, exec, review, app-server, mcp-server, mcp, plugin, license, update.
Plugin (AstraCode Chat)	Клиент в виде расширения для Visual Studio Code и совместимых сред (Cursor). Поставляется как платформо-зависимый пакет VSIX со встроенным исполняемым файлом ядра; графический интерфейс — встроенное окно браузера.
Desktop	Настольное графическое приложение на платформе Electron. Исполняемый файл ядра встроен в установщик; не требует Node.js, npm, Git или WSL.
Агент	Исполнительный контур AstraCode, который в рамках сессии ведёт диалог с языковой моделью, вызывает инструменты и выполняет действия над кодом и файлами пользователя. Жизненный цикл: Session → Task → Turn.
config.toml	Главный файл пользовательской конфигурации; значения объединяются по слоям: встроенные → managed → system → config.toml → проект → параметры командной строки.

Термин	Определение
safeStorage	Механизм шифрования секретов в Desktop через системное хранилище учётных данных (Astra Linux).
VSIX	Формат пакета расширения Visual Studio Code; Plugin распространяется как платформо-зависимые VSIX со встроенным исполняемым файлом.

1.3. Сокращения

Сокращение	Расшифровка
API	Application Programming Interface
CLI	Command Line Interface
TUI	Terminal User Interface
GUI	Graphical User Interface
IDE	Integrated Development Environment
IPC	Inter-Process Communication
MCP	Model Context Protocol
WAL	Write-Ahead Logging
UDS	Unix Domain Socket
SSE	Server-Sent Events
JSON-RPC	JSON Remote Procedure Call
MDM	Mobile Device Management
CA	Certificate Authority
TTL	Time To Live
SHA	Secure Hash Algorithm

2. Введение

AstraCode — ИИ-ассистент разработки на базе больших языковых моделей. Продукт создан для локального развёртывания: агент исполняется на рабочей станции пользователя, обращается к выбранной модели и выполняет задачи программирования непосредственно в проекте.

2.1. Продуктовая линейка

Продукт поставляется в виде трёх клиентских приложений, разделяющих общее ядро:

Клиент	Интерфейс	Платформа	Распространение
АстраCLI	Текстовый (TUI)	Astra Linux x86_64	Сборка из исходного кода
Plugin	Графический (встроенное окно браузера в VS Code/Cursor)	Astra Linux x64	Платформо-зависимые VSIX со встроенным исполняемым файлом
Desktop	Графический (настольное приложение)	Astra Linux x64	DEB/AppImage (Astra Linux)

Все три клиента запускают ядро astracode в режиме app-server и обмениваются с ним командами по протоколу JSON-RPC 2.0. АстраCLI может работать во внутрипроцессном режиме (TUI запускает ядро в том же процессе); Plugin и Desktop порождают ядро как дочерний процесс и связываются по stdio.

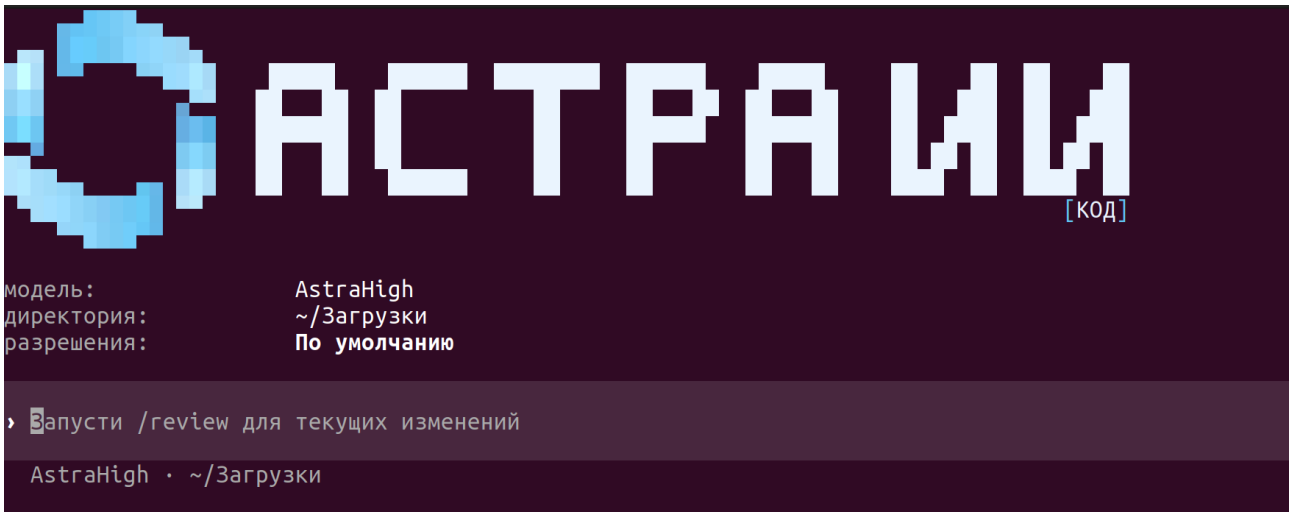

Рисунок 1 - Интерфейс АстраCLI

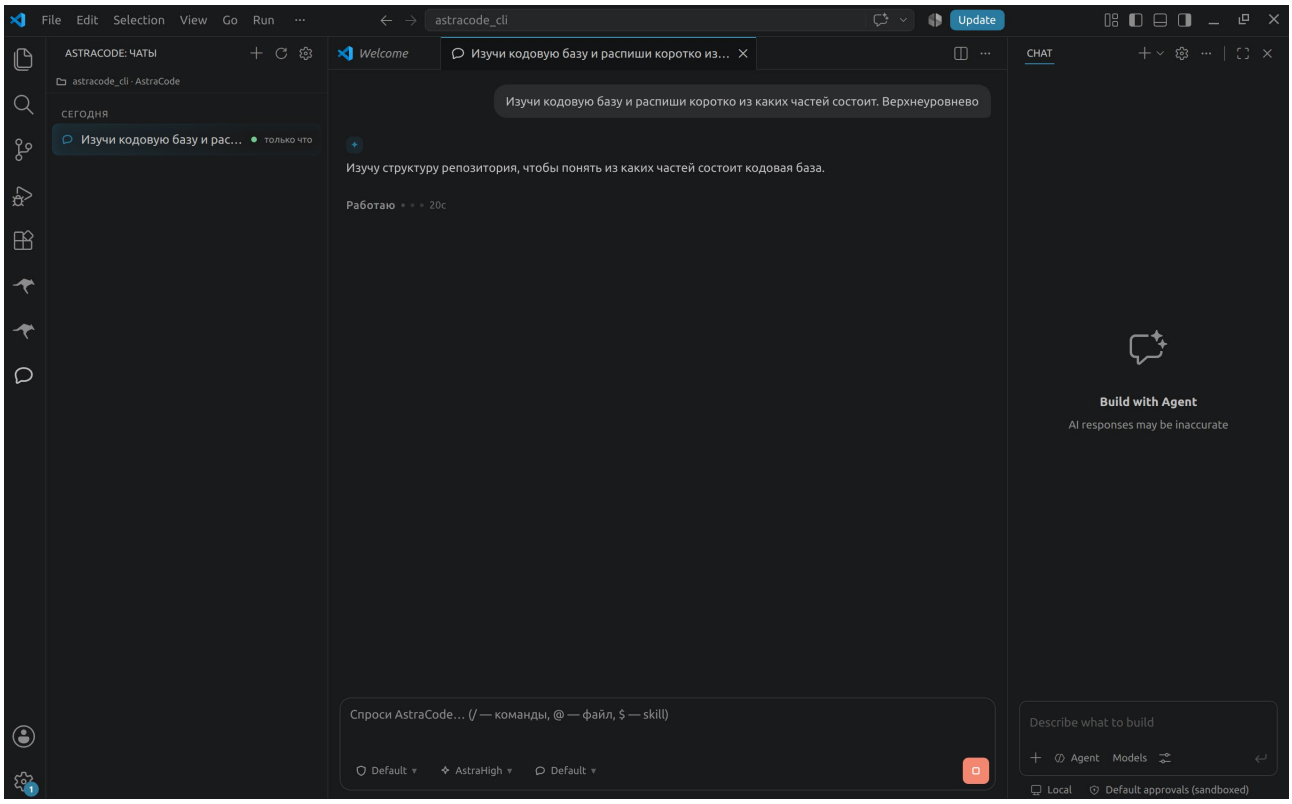

Рисунок 2 - Интерфейс Plugin

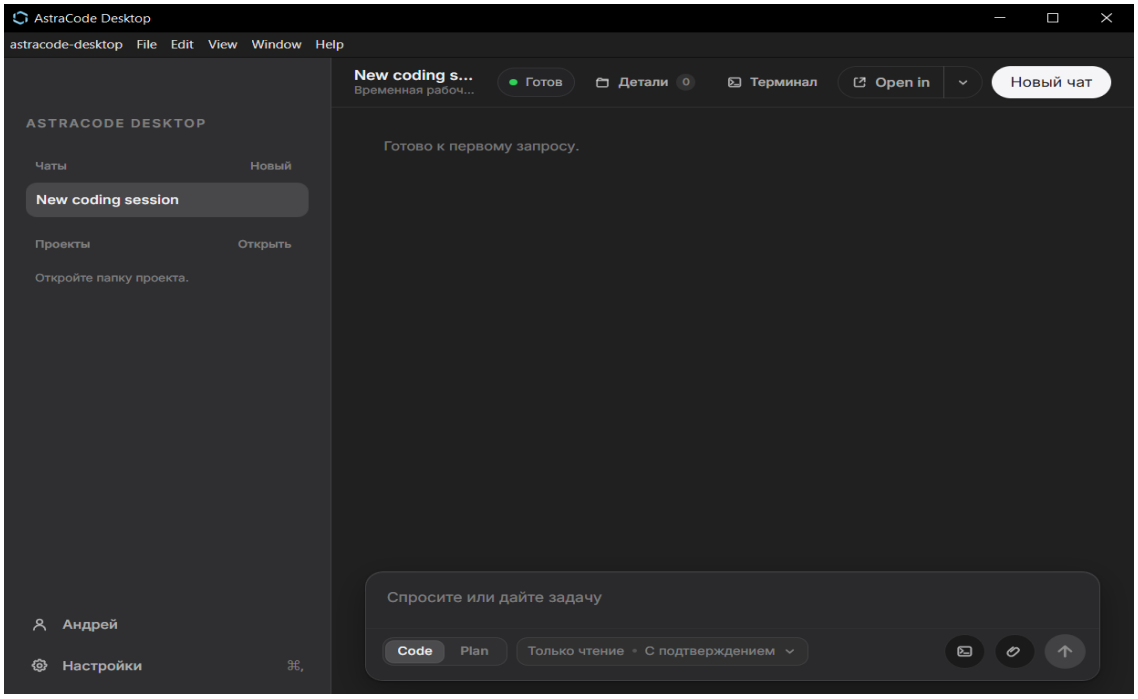

Рисунок 3 - Интерфейс Desktop: экран Code с открытым диалогом

2.2. Назначение продукта

AstraCode применяется для автоматизации задач программирования и работы с кодом:

- анализ структуры проекта и навигация по репозиторию;
- чтение, создание и изменение файлов исходного кода;
- выполнение shell-команд, запуск тестов и сборок, интерпретация результатов;
- проведение code review изменений перед фиксацией;
- работа с долгими задачами (цели), которые сохраняются между сессиями;
- интеграция с внешними системами через MCP-серверы.

Агент взаимодействует с проектом через набор встроенных инструментов (около 30), а модель определяет последовательность вызовов этих инструментов для решения поставленной задачи. Все действия, выходящие за рамки разрешённой политики, требуют подтверждения пользователя.

2.3. Подключение поставщиков моделей

Для работы агенту требуется поставщик модели. AstraCode подключается к поставщикам трёх категорий:

- **Облачные** — публичные API (OpenAI и др.).
- **Локальные** — vLLM, SGLang, LiteLLM и иные серверы, запущенные на оборудовании организации или пользователя.
- **Корпоративные** — внутренний шлюз моделей организации, доступный по сетевому адресу.

Внутренне AstraCode использует Responses API; для поставщиков, поддерживающих только Chat Completions, применяется встроенный мост (bridge), транслирующий запросы автоматически. Адрес, идентификатор и режим аутентификации задаются в config.toml; ключи хранятся зашифрованными локально и не передаются в открытом виде.

2.4. Уровни подготовки пользователей

Роль	Задачи
Администратор	Развёртывание на рабочих станциях, подготовка шаблонов config.toml, управление лицензиями, настройка политик безопасности, аудит, первичная диагностика.
Разработчик	Ведение сессий, управление моделью и провайдером, настройка MCP и плагинов, управление политиками подтверждений.
Конечный пользователь	Запуск в подготовленной среде, постановка задач, подтверждение/отклонение действий, базовые команды.

2.5. Задачи администратора

С точки зрения администратора, AstraCode — это приложение, которое необходимо установить на рабочие станции, настроить на корпоративный шлюз моделей, обеспечить лицензиями и задать политики безопасности. Конкретные задачи:

- развёртывание на рабочих устройствах (вручную или средствами распределённой установки);
- подготовка и распространение шаблона config.toml с параметрами провайдера, политиками песочницы и подтверждений;
- управление лицензиями: выпуск ключей, доставка, контроль активации;
- настройка доверенных сертификатов (ASTRACODE_CA_CERTIFICATE);
- задание политик безопасности через управляемый слой (managed_config.toml);
- аудит действий на устройстве через журналы.

3. Установка и первоначальная настройка

3.1. Системные требования

Общие требования

Параметр	Требование
Сетевой доступ	к выбранному провайдеру моделей (облачному или корпоративному шлюзу)

По клиентам

Клиент	Платформа	Минимум
CLI	Astra Linux x86_64 (glibc)	glibc 2.31+ (Astra Linux 1.7+)
Plugin	Astra Linux x86_64	VS Code 1.120.0+; glibc 2.31+
Desktop	Astra Linux x64	glibc 2.31+; RAM ⩾ 4 ГБ (рекомендуется 8 ГБ); диск ⩾ 500 МБ

3.2. Установка АстраCLI

```
# Astra Linux x86_64
tar xzf /tmp/astracode.tar.gz -C /tmp/
sudo mv /tmp/astracode /usr/local/bin/
astracode --version
```

Проверка установки

```
astracode --version       # версия и канал сборки
astracode mcp list        # список зарегистрированных MCP-серверов
```

3.3. Установка Plugin

Расширение поставляется как платформо-зависимый пакет VSIX со встроенным исполняемым файлом ядра. Отдельная установка консольного клиента не требуется.

Файлы поставки:

- astracode-<platform>-<version>.vsix;

Поддерживаемые платформы: linux-x64.

```
# Установка из командной строки
code --install-extension /path/to/astracode-linux-x64-0.5.0.vsix
```

Установка через интерфейс: Extensions → «…» → **Install from VSIX…** → выбрать файл.

После установки расширение размещается в ~/.vscode/extensions/astracode.astracode-<version>/; встроенный исполняемый файл — в <extension>/bin/astracode.

Разрешение исполняемого файла при работе

Поиск исполняемого файла ядра выполняется по трём ступеням:

1. Явная настройка astracode.binaryPath.
2. Зашитый <extension>/bin/astracode.
3. astracode из PATH (резервный вариант для режима разработки).

Выбранная ступень и путь записываются в журнал.

3.4. Установка Desktop

Приложение распространяется в виде установочных пакетов DEB и AppImage для Astra Linux; исполняемый файл ядра встроен в установщик в каталоге resources/bin/<платформа>-<архитектура>/. Для конечного пользователя установка Node.js, npm и Git не требуется.

Astra Linux (DEB/AppImage)

```
# DEB
sudo apt install ./AstraCode-Desktop-0.9.0-x64.deb

# AppImage (без установки)
chmod +x AstraCode-Desktop-0.9.0-x86_64.AppImage
./AstraCode-Desktop-0.9.0-x86_64.AppImage
```

Удаление установщика не затрагивает пользовательские данные (userData).

3.5. Первый запуск и настройка провайдера

Общий механизм

При первом запуске ядро создаёт каталог ~/.astracode/ (или в ASTRACODE_HOME), генерирует уникальный installation_id и, если провайдер не настроен, предлагает выбрать его. Режим песочницы по умолчанию зависит от доверия к рабочему каталогу: для нового (недоверенного) каталога — read-only; после принятия решения о доверии — workspace-write.

Подключение к корпоративному шлюзу (общее для всех клиентов)

```
default_provider = "company-llm"

[[providers]]
id = "company-llm"
display_name = "Корпоративный шлюз"
base_url = "https://<адрес шлюза организации>/v1"
```

Поля секции [[providers]]:

- id — стабильный идентификатор; ключ для default_provider и хранилища секретов.
- display_name — человекочитаемое имя для списка выбора.
- base_url — корневой URL; должен оканчиваться на /v1.
- model_override (необяз.) — принудительная подстановка идентификатора модели.
- extra_body (необяз.) — дополнительные поля в тело запроса к серверу модели.
- strip_strict (необяз.) — удаляет "strict": true из схем вызова функций (для Z.AI/GLM).
- forward_reasoning_summaries (необяз.) — пробрасывает reasoning_content как reasoning-summary.

Ввод ключа доступа

Ключ вводится через интерфейс клиента или через CLI (читает со стандартного ввода):

```
echo "sk-..." | astracode provider-secret set company-llm
```

Подкоманды provider-secret:

```
astracode provider-secret status company-llm      # наличие ключа
astracode provider-secret delete company-llm      # удалить ключ
astracode provider-secret probe company-llm       # проверить адрес /models
astracode provider-secret probe-capabilities company-llm --model <id>
```

Ключ **никогда не пишется в config.toml** — только идентификатор провайдера. Сам ключ шифруется и помещается в ~/.astracode/secrets/local.age.

Специфика первого запуска по клиентам

АстраCLI: открывает диалог /model в TUI с выбором провайдера и полем ввода ключа.

Plugin: при активации расширения, если провайдер не настроен, интерфейс предлагает добавить его в панели настроек («AstraCode: Open Settings»). Запуск ядра и короткие CLI-вызовы (provider-secret) используют единое значение ASTRACODE_HOME — иначе ключ записывается в одно хранилище, а app-server ищет его в другом.

Desktop: запускает мастер первого запуска с шагами: выбор языка → подключение провайдер → выбор модели → выбор политик (read-only / workspace-write / danger-full-access) → выбор папки проекта. Перед мастером показывается экран выбора аккаунта для создания локального профиля.

3.6. Массовое развёртывание

Раздача шаблона конфигурации

Администратор готовит и распространяет шаблон config.toml — **без секретов**:

```
default_provider = "company-llm"
approval_policy = "on-request"
sandbox_mode = "workspace-write"

[[providers]]
id = "company-llm"
display_name = "Корпоративный шлюз"
base_url = "https://<адрес шлюза организации>/v1"

[sandbox_workspace_write]
network_access = false
writable_roots = ["/home/<user>/work"]

[shell_environment_policy]
inherit = "minimal"
allow = ["PATH", "HOME", "USER", "LANG", "TERM"]
deny = ["SECRET_*", "AWS_*", "*_API_KEY", "GITHUB_TOKEN"]

[tui]
locale = "ru"
```

Шаблон размещается в ~/.astracode/config.toml.

Ключи API не переносятся между устройствами

Ключи шифруются относительно учётной записи ОС конкретного пользователя (мастер-ключ хранится в системное хранилище учётных данных и привязан к пути ASTRACODE_HOME). Поэтому:

- каждый пользователь вводит свой ключ на своём устройстве;
- передавать secrets/local.age бесполезно: без мастер-ключа целевой учётной записи он не расшифровывается;
- в шаблоне config.toml никаких ключей быть не должно.

Специфика массового развёртывания по клиентам

АстраCLI: распространение исполняемого файла; шаблон config.toml — через систему управления конфигурацией.

Plugin: распространение VSIX-файлов; установка командой code --install-extension.

Desktop: распространение установочных пакетов DEB/AppImage; установка через apt install или запуск AppImage.

Управляемый слой (MDM)

Конфигурация считывается из иерархии слоёв: встроенные значения → управляемый слой → системный → пользовательский → проектный → параметры командной строки. Управляемый слой (managed_config.toml, распространяемый через средства управления устройствами) имеет более высокий приоритет, чем пользовательские настройки, и позволяет зафиксировать политики, которые пользователь не сможет переопределить.

Расположение управляемого слоя:

Платформа	Путь
Astra Linux	/etc/astracode/managed_config.toml

4. Настройка провайдеров и моделей

Глава описывает подключение поставщиков моделей, работу с локальными и корпоративными развёртываниями, выбор активной модели и обслуживание кэша каталога моделей — общие принципы, единые для всех клиентов, с указанием особенностей по клиентам.

4.1. Подключение к поставщикам моделей

Поставщики описываются в ~/.astracode/config.toml плоской таблицей [[providers]] — по одной записи на каждый сервер:

```
default_provider = "local-vllm"

[[providers]]
id = "local-vllm"
display_name = "Local vLLM"
base_url = "http://111.111.11.11:8000/v1"
```

Все провайдеры считаются говорящими по стандарту OpenAI Chat Completions на <base_url>/chat/completions и предоставляющими /v1/models. Трансляцию в Responses API выполняет встроенный мост: прокси поднимается ядром на свободном localhost-порту и завершается при остановке.

При загрузке конфигурации массив валидируется: id, display_name и base_url не должны быть пустыми, base_url должен быть корректным URL, дубликаты id запрещены.

Несколько провайдеров сосуществуют в одном config.toml; переключение — через /model или интерфейс клиента. Последний выбор сохраняется в [ui_state]:

```
[ui_state]
last_used_provider = "AstraCode"
last_used_model = "AstraHigh"
```

Удаление провайдера убирает запись из конфигурации и стирает его API-ключ из secrets; операция необратима.

4.2. Локальные и корпоративные развёртывания

Типовые base_url локальных провайдеров:

Сервер	base_url
vLLM	http://localhost:8000/v1
LiteLLM	http://localhost:4000/v1

Если сервер не требует ключа, вводится заглушка: echo "dummy" | astracode provider-secret set local-llama.

Прокси:

```
export HTTPS_PROXY="http://proxy.company.com:3128"
export HTTP_PROXY="http://proxy.company.com:3128"
export NO_PROXY="localhost,127.0.0.1,*.internal"
```

4.3. Выбор активной модели

Активный поставщик задаётся параметром default_provider. Конкретная модель выбирается командой /model или в интерфейсе клиента; список моделей загружается из /v1/models и объединяется со встроенным каталогом. Выбор сохраняется.

4.4. Кэш каталога моделей

Менеджер моделей кэширует каталог на диске в ASTRACODE_HOME. Срок свежести кэша ограничен. Кэш считается актуальным при совпадении версии клиента с текущей версией AstraCode и возрасте записи в пределах срока.

4.5. Управление провайдерами через интерфейс клиента

АстраCLI

Команда /model в TUI открывает трёхколонный выбор: провайдер → статус → модели. Клавиши: Enter — активировать, r — обновить, d — удалить, + Add provider — добавить.

Plugin

Панель настроек («AstraCode: Open Settings») предоставляет форму добавления/редактирования провайдеров. Активный провайдер зеркалируется в настройки VS Code — это зеркало для интерфейса; источником правды остаётся config.toml. Смена активного провайдера перезапускает ядро.

Plugin записывает конфигурацию через ядро, а не прямым редактированием файла. Для предотвращения потери данных при параллельном редактировании применяется оптимистичная блокировка: при чтении получается версия конфигурации, передаваемая при записи; ядро отклоняет запись, если файл изменился; plugin перечитывает и повторяет попытку.

Дополнительные механизмы:

- отслеживание внешних изменений config.toml (например, из TUI) с фильтрацией ложных срабатываний;
- автоматическое исправление: если активный указатель провайдера ссылается на несуществующую запись, plugin переписывает указатель на существующего провайдера или удаляет его, сохраняя остальные настройки.

Desktop

Настройки провайдеров — в «Параметры» → вкладка «Код». Форма с полями: имя, base URL, модель, API-ключ, расширенные параметры. При сохранении изменения передаются во встроенный сервер, который перезапускается при необходимости. Ключ сохраняется через astracode provider-secret set.

Desktop кэширует проверку возможностей моделей в локальной БД с ограниченным сроком действия. Маскированный показ ключа — первые четыре и последние четыре символа.

5. Безопасность

Все клиенты дают модели доступ к коду и системе. Без защиты это рискованно, поэтому встроена многоуровневая модель безопасности из трёх независимых слоёв: песочница, подтверждения и секреты. Дополнительный слой — контроль лицензии. Эти механизмы реализованы в общем ядре и едины для всех клиентов.

5.1. Песочница: три режима

Режим	Поведение
read-only	Самый строгий. Чтение любых файлов

Режим	Поведение
	разрешено, запись — никуда, сеть отключена.
workspace-write (по умолчанию для доверенного каталога)	Чтение — везде. Запись — только в рабочий каталог и /tmp. Сеть управляется отдельно (network_access = false по умолчанию).
danger-full-access	Полностью отключает песочницу. Только для явно доверенных сценариев; переход требует подтверждения.

Реализация по платформам:

- **Astra Linux:** bwrap (namespaces user/PID/network/mount) + seccomp (блокировка setuid/setgid/capset; при отключённой сети — создание сокетов); резервный — landlock ([features] use_legacy_landlock = true).

Конфигурация:

```
sandbox_mode = "workspace-write"

[sandbox_workspace_write]
writable_roots = ["/home/me/work"]
network_access = false
exclude_tmpdir_env_var = false
exclude_slash_tmp = false
```

5.2. Подтверждения: политики

Политика	Поведение
untrusted	Каждое действие, способное писать на диск или в сеть, требует подтверждения.
on-request (по умолчанию, рекомендуется)	Модель сама решает, когда запросить подтверждение.
on-failure	Устаревшая. Подтверждение — только при блокировке песочницей.
never	Без запросов; для автоматизированных запусков.

Рецензент подтверждений:

```
approval_policy = "on-request"
approvals_reviewer = "user"      # user | auto_review
```

auto_review (синоним guardian_subagent) включает автоматическую проверку подчинённым агентом на второй модели. Действия, отклонённые авто-проверкой, можно поштучно одобрить через /autoreview.

Специфика интерфейса:

- **CLI:** нумерованный список в TUI (Yes / Yes, don’t ask again / No).
- **Plugin:** карточки подтверждений непосредственно в ленте чата (Enter — разрешить, Esc — отклонить).
- **Desktop:** карточки подтверждений с кнопками одобрить/отклонить/разрешить один раз; при незакрытом окне — системное уведомление.

5.3. Защищённые пути

Даже в workspace-write или danger-full-access остаются защиты на уровне файловой системы:

- .git/ — никогда (защита истории git).
- .astracode/ — никогда (защита настроек).
- .agents/ — никогда (защита устаревших конфигураций).

Это жёсткие ограничения: песочница применяет их, даже если подтверждение разрешает действие.

5.4. Секреты: локальное шифрование

API-ключи провайдеров, OAuth-токены MCP и иные учётные данные хранятся в ~/.astracode/secrets/local.age — зашифрованном файле (схема age с функцией формирования ключа scrypt).

Мастер-ключ хранится в системном хранилище учётных данных (Astra Linux) с привязкой к пути ASTRACODE_HOME.

Следствия:

- Без входа в ОС расшифровать local.age нельзя.
- Копирование local.age на другое устройство бесполезно.
- Обновление не ломает доступ к секретам.
- В config.toml ключи не хранятся.

При недоступности системного хранилища (сервер без графического интерфейса) ключ записывается в ~/.astracode/secrets/local.key с ограничением прав. Переменная ASTRACODE_PROVIDER_SECRETS_KEY_STORAGE принудительно выбирает способ хранения.

Desktop дополнительно: для собственных секретов приложения применяется системное шифрование (safeStorage, Astra Linux).

Секреты делятся по области: **глобальные** и **привязанные к окружению** (к git-репозиторий или cwd).

5.5. Фильтрация переменных окружения

```
[shell_environment_policy]
inherit = "minimal"            # all | minimal | none
allow = ["PATH", "HOME", "USER", "LANG", "TERM"]
deny = ["SECRET_*", "AWS_*", "*_API_KEY", "GITHUB_TOKEN"]
```

inherit = "minimal" оставляет только базовые переменные; deny — маски имён, блокируемые даже при попадании в allow.

5.6. Журнал аудита

Все решения о подтверждениях и эскалации фиксируются в ~/.astracode/log/astracode-tui.log с префиксом audit:::

```
audit::approval granted tool=local_shell cmd="rm tmp.txt" reason="user_approved"
audit::escalation requested cmd="sudo systemctl restart x" → denied
```

5.7. Контроль лицензии

Исполняемый файл ядра требует действительный подписанный токен лицензии в ~/.astracode/license_state.json либо запись в системном хранилище учётных данных.

```
astracode license activate --file /path/to/license.key
astracode license status
```

Альтернативный способ — поместить license.key в каталог ASTRACODE_HOME (активация размещением файла).

Каналы сборок: licensed (требует ключ) и nolicense (без проверки); канал виден по суффиксу -nolicense в выводе --version. Подкоманды Update и License освобождены от проверки.

5.8. Модель угроз

От чего защищает: случайные деструктивные команды модели; утечка API-ключей через дочерние процессы; чтение/запись вне проекта без подтверждения; сетевые запросы при network_access = false; перезапись настроек ядра и git-истории.

От чего не защищает: сознательно вредоносная модель; эксплойты в bwrap; атаки по сторонним каналам; социальная инженерия в диалогах подтверждения. Песочница — эшелонированная защита, а не абсолютная защита.

6. Администрирование

6.1. Управление пользователями и доступом

AstraCode не имеет собственной системы учётных записей: клиент запускается от имени пользователя ОС и хранит данные в ~/.astracode/. «Пользователем» с точки зрения администрирования является рабочая станция с установленным клиентом; разграничение доступа сводится к настройке параметров безопасности на устройстве.

Параметры доступа регулируются двумя независимыми параметрами: sandbox_mode и approval_policy (см. главу 5). Значения можно переопределить через параметры командной строки (--sandbox, --approval) или ключ -c:

```
astracode --sandbox read-only --approval untrusted
astracode -c sandbox_mode=workspace-write -c 'tui.locale="ru"'
```

Поскольку пользователь имеет локальный доступ к config.toml, он способен изменить локальные значения; централизованно «заморозить» их можно только через управляемый слой (см. 6.4).

6.2. Лицензирование и активация

Активация и проверка состояния

```
astracode license activate --file /путь/к/license.key
astracode license status
```

Команды license status и license activate работают без активной лицензии. Команда astracode update также лицензионно-освобождена.

Хранение состояния активации

Запись об активации сохраняется:

- в системном хранилище учётных данных — предпочтительно;
- либо в файле ~/.astracode/license_state.json (резервный вариант).

Режим хранения выбирается переменной ASTRACODE_LICENSE_KEY_STORAGE; принудительный отказ от системного хранилища — ASTRACODE_LICENSE_DISABLE_KEYRING=1.

Каналы сборок

Каждая сборка распространяется в одном из двух каналов: licensed (требует ключ) и nolicense (без проверки). Само клиентское приложение лицензию не проверяет — это делает ядро. Канал определяется на этапе сборки и виден по суффиксу -nolicense в выводе --version и в имени файла артефакта.

6.3. Журналирование и аудит

Журналы ядра (общие)

Текстовые журналы — ~/.astracode/log/astracode-tui.log. Формат: [timestamp] [level] [module] message. Ротация по размеру. Расположение переопределяется log_dir в config.toml.

База журналов — ~/.astracode/logs_2.sqlite. События запросов к LLM, вызовы инструментов, ошибки. Секретов не содержит.

БД состояния — ~/.astracode/state_5.sqlite. Метаданные потоков диалога, цели, агентные задачи, аудит подтверждений.

Журналы сессий — ~/.astracode/sessions/<год>/<месяц>/<день>/*.jsonl. Полная последовательность событий сессии.

Журналы клиентов

Plugin: журнал расширения — Output-канал «AstraCode» и файл в каталоге расширения (с ограничением прав и ротацией по размеру).

Desktop: журнал обновлений — <userData>/updater.log. Последние строки потока ошибок встроенного сервера с маскированием секретов.

Сбор диагностики

Источник	Расположение	Назначение
Текстовый журнал	~/.astracode/log/astracode-tui.log	Трассировка, аудит, ошибки MCP
БД журналов работы	~/.astracode/logs_2.sqlite	Журнал событий работы ядра
БД состояния	~/.astracode/state_5.sqlite	Метаданные потоков диалога, цели, аудит
Журналы сессий	~/.astracode/sessions/…/*.jsonl	Полная последовательность событий сессии

Команда /debug-config в TUI показывает слои конфигурации с источниками значений. Телеметрия отключается переменной ASTRACODE_NO_TELEMETRY.

Plugin: «AstraCode: Collect Diagnostics» собирает архив: журналы сессий, журнал расширения, сведения об окружении и манифест. Перед передачей проверить содержимое — журналы могут содержать код и запросы.

Desktop: «AstraCode: Collect Diagnostics»; диагностика в «Параметры» → «Код» → «Диагностика» (сборка, обновления, песочница, журнал ошибок).

6.4. Управление конфигурацией

Каталог ASTRACODE_HOME

```
~/.astracode/
├── config.toml            # пользовательский конфиг
└── managed_config.toml    # управляемый слой (опционально)
```

```
├── license_state.json      # состояние активации (резервный файл)
├── secrets/
│   ├── local.age           # зашифрованное хранилище (age + scrypt)
│   └── local.key           # резервный мастер-ключ (0o600)
├── state_5.sqlite          # БД состояния
├── logs_2.sqlite           # БД журналов
├── sessions/               # журналы сессий (по дате)
├── memories/               # память (под git)
├── skills/                 # умения (.system/ — встроенные)
├── log/                    # текстовые журналы
└── ...
```

Слои конфигурации

Конфигурация собирается из слоёв (от низшего к высшему):

1. Встроенные значения по умолчанию.
2. Управляемый слой (managed_config.toml).
3. Системный слой (/etc/astracode/config.toml).
4. Пользовательский слой (${ASTRACODE_HOME}/config.toml).
5. Проектный слой (./.astracode/config.toml).
6. Параметры командной строки и ключи -c (наивысший приоритет).

Управляемый слой позволяет администратору зафиксировать политики, которые пользователь не сможет переопределить.

Общие секреты для всех клиентов

При едином ASTRACODE_HOME все три клиента используют общую конфигурацию и общее хранилище secrets/local.age. Ключ, введённый в одном клиенте, доступен в остальных без повторного ввода.

Резервное копирование

Что подлежит резервному копированию: config.toml, secrets/local.age (но без мастер-ключа из системного хранилища бесполезен), memories/, skills/<custom>/, state_5.sqlite.

```
tar czf astracode-backup-$(date +%F).tar.gz \
  ~/.astracode/config.toml \
  ~/.astracode/secrets/ \
  ~/.astracode/memories/ \
  ~/.astracode/skills/
```

Не требует резервирования: sessions/, logs_2.sqlite, log/, skills/.system/, tmp/, shell_snapshots/.

Desktop: резервное копирование — копирование каталога userData целиком. При копировании на работающем приложении следует использовать стандартные средства резервного копирования SQLite. Секреты (API-ключи) при переносе между устройствами не сохраняются — они зашифрованы мастер-ключом, привязанным к учётной записи ОС.

6.5. Обновление

АстраCLI

Встроенная команда (рекомендуется):

```
astracode update
```

Ядро скачивает свежую сборку, проверяет контрольную сумму, распаковывает и атомарно заменяет текущую версию. Работает без активной лицензии.

В TUI доступна команда /update: после установки ядро перезапускается с новой версией, и пользователь оказывается в обновлённом TUI без перезапуска оболочки.

Сохранность секретов: формат шифрования не меняется между версиями; мастер-ключ зависит от ASTRACODE_HOME, а не от версии исполняемый файла.

Plugin

Обновление — переустановка VSIX:

```
code --install-extension /path/to/astracode-<platform>-<new_version>.vsix
```

с последующей перезагрузкой окна Секреты сохраняются.

Desktop

Автоматическое обновление по ленте HTTPS. Принципы:

- недоступность сервера обновлений нефатальна;
- ничего не скачивается без ведома пользователя — сначала вопрос, затем загрузка;
- адрес ленты — одно значение сборки (ASTRACODE_PUBLISH_BASE_URL).

Обновление не скачивается и не устанавливается автоматически без подтверждения. Переменная ASTRACODE_UPDATER_AUTO=1 включает автоматический режим. Адрес с хостом .invalid трактуется как «обновления не настроены». Журнал — <userData>/updater.log.

6.6. Удаление

АстраCLI

```
sudo rm /usr/local/bin/astracode        # исполняемый файл
rm -rf ~/.astracode                     # пользовательские данные (НЕОБРАТИМО)
```

Plugin

```
code --uninstall-extension astracode.astracode
```

После удаления ~/.astracode/ сохраняется. Удаление данных — отдельная задача: rm -rf ~/.astracode.

Desktop

Удаление через систему (Пуск → Удалить программу, apt remove, удаление DMG-приложения). Пользовательские данные в userData сохраняются до явного удаления.

6.7. Desktop: встроенный терминал

Встроенный эмулятор терминала с поддержкой полноэкранных консольных программ (vim, htop) и ввода в строке чата. Выбор оболочки: $SHELL → bash → sh (Astra Linux).

7. Типовые проблемы и их решения

7.1. Общие проблемы

Симптом	Причина	Решение
command not found (CLI)	Исполняемый файл не в PATH	which astracode; проверить /usr/local/bin/astracode; export PATH
bwrap: Permission denied	Astra Linux отключил unprivileged user	sudo sysctl kernel.unprivileged_userns_clone=1;

Симптом	Причина	Решение
	namespaces	или [features] use_legacy_landlock = true
bwrap: No such file or directory	Системный bwrap не установлен	sudo apt install bubblewrap
Failed to decrypt secrets file	системное хранилище учётных данных потерял мастер-ключ	rm ~/.astracode/secrets/local.age; ввести ключи заново
Could not find suitable model provider	Провайдер не настроен	/model → выбрать или добавить провайдера
401 Unauthorized	API-ключ недействителен или неверный	Удалить и заново добавить провайдера; provider-secret set
Ошибка самоподписанного сертификата	Локальный провайдер с собственным CA	export ASTRACODE_CA_CERTIFICATE=...
Каждое действие спрашивает разрешение	approval_policy = "untrusted"	/approvals → предустановка «По умолчанию» (on-request)
Сообщение об отсутствии лицензии	Сборка с контролем без активации	astracode license activate --file <файл>; или сборка nolicense

7.2. Специфичные проблемы Plugin

Симптом	Причина	Решение
Исполняемый файл ядра не найден	VSIX не для той платформы; неверный astracode.binaryPath	Установить VSIX нужной платформы; проверить настройку
Timeout инициализации соединения	Несовместимые версии расширения и ядра	Установить согласованную пару VSIX
Доступ к провайдеру отклонён при наличии ключа	Ключ записан в другое хранилище (разный ASTRACODE_HOME)	Обеспечить единый ASTRACODE_HOME; повторно ввести ключ
Расширение не активируется	VS Code ниже 1.120.0; VSIX не для той платформы	Проверить версию VS Code; установить правильный VSIX
Нет сети у агента	approvalPolicy не on-request	Установить astracode.approvalPolicy: on-request

7.3. Специфичные проблемы Desktop

Симптом	Причина	Решение
Сервер не запускается, binary_not_found	Встроенный исполняемый файл отсутствует/повреждён	Переустановить; проверить SHA-256 установщика
Сервер не запускается, ready_timeout	Сервер не отвечает вовремя	Проверить поток ошибок в «Диагностике»; проверить нагрузку процессора/памяти

Симптом	Причина	Решение
Ключ провайдера не сохраняется	Системное шифрование недоступно	Настроить системное хранилище (libsecret)
Песочница недоступна (предупреждение)	Astra Linux без bubblewrap и Landlock (ядро < 5.13)	sudo apt install bubblewrap или обновить ядро
Обновление не приходит	Лента не настроена (.invalid) или недоступна	Проверить updater.log; убедиться, что ASTRACODE_PUBLISH_BASE_URL задан
БД повреждена	Сбой при записи; ручная очистка БД	Восстановить из резервной копии; не обнулять БД перенаправлением вывода
Приложение не запускается	ELECTRON_RUN_AS_NODE конфликт	Удалить переменную; запустить из терминала для диагностики

7.4. Сбор журналов

CLI:

```
tail -200 ~/.astracode/log/astracode-tui.log > /tmp/astracode-debug.txt
astracode --version >> /tmp/astracode-debug.txt
```

В TUI /debug-config покажет слои конфигурации. Отправляйте через /feedback.

Plugin: «AstraCode: Collect Diagnostics» или скрытая команда /give_me_logs.

Desktop: «Параметры» → «Код» → «Диагностика» — сборка, обновления, песочница, поток ошибок.

Не отправляйте secrets/, sessions/, журналы сессий — они могут содержать код и запросы.

7.5. Полный сброс

```
tar czf astracode-backup.tar.gz ~/.astracode/config.toml ~/.astracode/skills/   # опционально
rm -rf ~/.astracode
```

8. Дополнительные возможности

8.1. Обращение в службу поддержки

Основной канал — команда /feedback в интерфейсе клиента. Она отправляет категорию (Bug, Bad result, Good result, Safety check, Other), заметку и метаданные (версия, ОС, модель, провайдер, ID потока диалога) на HTTP-адрес коллектора. Журналы и запись сессии не отправляются. Адрес коллектора и данные базовой аутентификации зашиты в исполняемый файл. Команда отключается в config.toml.

8.2. Обратная связь (Desktop)

В Desktop /feedback открывает форму с пятью категориями и текстовым полем. Отправка идёт через встроенный сервер, не напрямую. При неудаче форма не закрывается — можно повторить. Отправляются категория, заметка и метаданные (версия, модель, провайдер, ОС, ID потока диалога); журналы не отправляются.

8.3. История изменений

Поверхности контроля

Поверхность	Команда	Что показывает
Список сессий	/resume	Сохранённые потоки диалога: дата, git-ветка, рабочий каталог
Текущая конфигурация	/status	Модель, провайдер, режим песочницы, расход токенов
Изменения рабочего дерева	/diff	git diff по отслеживаемым файлам
Code review	/review	Рецензия агента по незакоммиченным изменениям
Отмена правок	/undo	Откат файловых изменений агента
Фоновые процессы	/ps	Список фоновых терминалов
Ответвление диалога	/fork	Новый чат от выбранного сообщения
Журнал аудита	audit:: в журнале	Решения о подтверждениях и эскалациях

Хранение

- **Журналы сессий** — ~/.astracode/sessions/<год>/<месяц>/<день>/<thread_id>.jsonl.
- **БД состояния** state_5.sqlite — индекс потоков диалога, цели, агентные задачи, аудит подтверждений.
- **БД журналов работы** logs_2.sqlite — журнал событий работы ядра.
- **История ввода** history.jsonl — тексты, введённые пользователем.
- **Baseline-снимки для /undo** — ~/.astracode/projects/<sha256_cwd>/undo/<thread_id>.jsonl.

Ротация

```
find ~/.astracode/sessions -mtime +90 -delete
rm -rf ~/.astracode/log/*
```

БД журналов нельзя обнулять перенаправлением вывода — это повреждает WAL; обслуживание только при остановленном AstraCode.