



АСТРА ИИ

АСТРА ИИ [ХАБ]

Техническое описание

Москва
2026

СОДЕРЖАНИЕ

1. Введение.....	3
2. Класс решаемых задач.....	8
3. Архитектура.....	15
4. Основные свойства продукта.....	24
5. Типовые сценарии.....	31
6. Функционал программных модулей.....	38
7. Системные требования.....	51
8. Лицензирование.....	61

1. Введение

1.1. Назначение и область применения

Программное обеспечение АСТРА ИИ [ХАБ] (ASTRA AI Hub) — корпоративная платформа управления доступом к моделям искусственного интеллекта и их эксплуатацией. Платформа создаёт единый управляемый контур для использования искусственного интеллекта в организации: обеспечивает централизованный доступ к локальным и внешним моделям, применяет корпоративные политики безопасности и разграничения прав, управляет развёртыванием и жизненным циклом моделей, контролирует потребление ресурсов и формирует единую аналитику использования.

Базовый вариант платформы разворачивается в Kubernetes-кластере заказчика и предоставляет:

- единый OpenAI-совместимый программный интерфейс к локальным и внешним моделям ИИ;
- продуктовый уровень управления организациями, пользователями, ролями, подписками, ключами доступа, расписаниями работы, квотами потребления и аналитикой использования;
- платформенный уровень управления жизненным циклом моделей, вычислительными ресурсами и автомасштабированием.

Платформа обслуживает три категории пользователей и систем:

Пользователи и приложения

Корпоративные пользователи обращаются к моделям через единый программный интерфейс, совместимый с OpenAI API. Запросы выполняются от имени пользователя с применением персональных ключей доступа и политик организации. API-клиенты и приложения (консольные инструменты, плагины для интегрированных сред разработки, настольные приложения) используют тот же интерфейс, что обеспечивает единообразие интеграции.

Клиентские приложения семейства АСТРА ИИ [КОД] (далее – AstraCode) (консольный агент разработки, плагин для Visual Studio Code, настольное приложение) не входят в состав платформы АСТРА ИИ [ХАБ]. Они являются внешними потребителями моделей через программный интерфейс платформы.

Администраторы организации

Администраторы организации управляют пользователями, командами, ролями и правами в рамках своей организации. Они выдают и отзывают персональные и сервисные ключи доступа, настраивают расписания доступности, контролируют квоты потребления и просматривают аналитику использования моделей в пределах своей организации. Права администратора организации ограничены контуром этой организации и не распространяются на платформенные функции управления моделями и вычислительной инфраструктурой.

Администраторы платформы

Администраторы платформы управляют каталогом моделей, конфигурациями развёртывания, тарифами, подписками организаций и вычислительными ресурсами. Они создают и настраивают развёртывания локальных моделей, регистрируют внешние источники моделей, распределяют GPU между моделями, просматривают статусы и метрики платформы. Администратор платформы имеет доступ к аналитике по любой организации, но не обязан быть членом этих организаций.

1.2. Состав рассматриваемого продукта

АСТРА ИИ [ХАБ] — единый продукт, разрабатываемый двумя командами с чётким разделением ответственности по контурам:

Продуктовый контур АСТРА ИИ

Продуктовый контур охватывает всё, что касается пользовательского управления, политик доступа, учёта и аналитики:

- веб-консоль администрирования — интерфейс управления организациями, пользователями, командами, ролями, тарифами, подписками, ключами доступа, квотами, расписаниями, конфигурациями развёртывания моделей и аналитикой использования;
- серверная часть продукта — организации, пользователи, команды, роли, тарифы, подписки, ключи запросов к моделям, политики доступа, расписания, квоты, продуктовый журнал запросов, аналитика и учёт потребления;
- продуктовая точка входа запросов к моделям — проверка членства в организации, активности подписки, списка разрешённых моделей, расписания доступа и квот до вызова модели; маршрутизация разрешённых запросов в шлюз моделей;
- система идентификации и аутентификации — интеграция с корпоративной системой идентификации через OIDC (DEX на базе Keycloak), проверка пользовательских сессий и сервисных ключей;
- HTTP-адаптер — мост между REST API продукта и пользовательскими ресурсами Kubernetes, обеспечивает декларативное управление развёртываниями моделей, арендаторами платформы, клиентами API и вычислительными ресурсами без прямого обращения консоли к API кластера.

Платформенный контур исполнения моделей

Платформенный контур охватывает инфраструктуру, жизненный цикл моделей и вычислительные ресурсы:

- оператор Kubernetes — согласование состояния ресурсов моделей: управление кэшем весов, томами, подами исполнения, статусами; безопасное обновление развёртываний с учётом GPU;
- пользовательские ресурсы Kubernetes — декларативные описания моделей, профилей исполнения, классов исполнения, кэша весов, арендаторов платформы, клиентов API, цен и политик защиты содержимого;
- загрузчик весов моделей — загрузка весов из репозитория HuggingFace, зеркал и S3-совместимых хранилищ в кэш на постоянном томе;

- среда исполнения — поддержка движков vLLM, SGLang, Ollama, FasterWhisper для обслуживания языковых моделей, векторных представлений, переранжирования и распознавания речи; режим внешнего поставщика для проксирования запросов к OpenAI-совместимым внешним сервисам;
- автомасштабирование — переход $0 \leftrightarrow 1$ средствами оператора и масштабирование $1 \leftrightarrow N$ через KEDA на основе метрик нагрузки из Prometheus или VictoriaMetrics;
- наблюдаемость и метрики — экспорт метрик оператора и шлюза запросов, интеграция с Prometheus, VictoriaMetrics, Grafana, Loki, OpenTelemetry и DCGM для мониторинга GPU.

Совместные компоненты

Некоторые компоненты используются обеими командами с разделением ответственности по функциям:

- шлюз запросов к моделям — OpenAI-совместимый интерфейс с поддержкой маршрутов `/v1/chat/completions`, `/v1/completions`, `/v1/embeddings`, `/v1/audio/*`, `/v1/rerank`, `/v1/models`; функции категорийного учёта токенов, быстрых лимитов, расписаний доступности моделей, включаемых политик защиты содержимого (Guardrails) и исходящих уведомлений реализуются продуктовым контуром, инфраструктурная поставка и согласование состояния — платформенным контуром;
- PostgreSQL — разные схемы и роли: продуктовый журнал запросов, учёт потребления и аналитика ведутся продуктовым контуром; учёт платформы и хранение диалогов при включении соответствующей функции — платформенным контуром;
- ValKey — кэш ключей доступа для быстрой проверки на входном шлюзе APISIX, резервы квот, события учёта и кэш контекста инференса; используется обеими командами.

Дополнительные сервисы

В продуктовый контур платформы включён доступ к следующим сервисам:

- единый MCP Hub — централизованное управление серверами Model Context Protocol;
- RAG-сервис — сервис расширенной генерации с привлечением информации;
- маркетплейс Skills — каталог умений и навыков для агентских систем.

Детальный набор пользовательских функций, API, правила публикации контента и эксплуатационные параметры этих сервисов определяются отдельно в зависимости от состава конкретной поставки. В настоящем описании они обозначены как дополнительные сервисы экосистемы платформы без расширения заявленного перечня функций.

1.3. Варианты размещения

Базовая модель размещения использует Kubernetes версии не ниже 1.29 и Helm для установки компонентов платформы. Плоскость управления и плоскость исполнения запросов к моделям масштабируются независимо. Базовая поставка (оператор, шлюз запросов, пользовательские ресурсы для жизненного цикла моделей)

работоспособна без необязательных подсистем (учёт на стороне платформы, хранение диалогов, защита содержимого, KEDA, VictoriaMetrics, сетевые политики).

Из единой кодовой базы поддерживаются три варианта размещения, различающиеся конфигурацией:

Контур заказчика

Базовый сценарий: платформа и локальные модели разворачиваются в Kubernetes-кластере заказчика. Все данные запросов, учёта и аналитики остаются в инфраструктуре заказчика. При использовании только локальных моделей данные не покидают периметр заказчика.

Облачный вариант

Платформа может разворачиваться в облачной инфраструктуре с той же кодовой базой и набором компонентов. Детальная облачная топология определяется конфигурацией и не зафиксирована как отдельная архитектурная ветвь.

Изолированный стенд

Отдельная конфигурация для закрытого или автономного окружения, где требуется полная изоляция от внешних сетей и сервисов.

Интеграция с внешними системами

Внешний доступ к платформе организуется через входной шлюз APISIX с поддержкой OIDC для интерактивных пользователей и проверкой сервисных ключей для API-клиентов. Клиентские identity-заголовки заменяются доверенными на стороне шлюза, что исключает подмену личности на стороне клиента.

В целевой схеме внешнего доступа предусмотрена опциональная интеграция с Termidesk Connect. Termidesk Connect не входит в обязательный минимум базовой поставки. При включении этой опции Termidesk Connect используется как точка безопасного подключения внешних клиентов Astra Code к APISIX и как возможный канал к облачным и внешним контурам моделей. Без Termidesk Connect внешний доступ организуется иными согласованными средствами периметра, внутренние клиенты Astra Code подключаются к APISIX по VPN с JWT, а клиенты вне экосистемы Astra Code — к шлюзу моделей по VPN с сервисным ключом.

Ключевые инфраструктурные зависимости

Для работы платформы требуются следующие компоненты инфраструктуры:

Компонент	Роль	Обязательность
Kubernetes ≥ 1.29	Оркестрация, пользовательские ресурсы	Базовая
Helm	Установка платформы	Базовая
PostgreSQL	Продуктовый журнал запросов; учёт и диалоги платформы	Для учёта и аналитики

Компонент	Роль	Обязательность
ValKey	Кэш ключей, резервы квот, события учёта	Промышленный продуктовый режим
S3-совместимое хранилище	Веса моделей, холодное хранение диалогов	По функции
KEDA / Prometheus / VictoriaMetrics	Автомасштабирование 1↔N, метрики	По необходимости
DEX (Keycloak)	Поставщик корпоративной идентификации	Для консоли и OIDC-клиентов

Состав зависимостей определяется конфигурацией поставки: базовый оператор и шлюз запросов работают без KEDA, VictoriaMetrics и подсистемы хранения диалогов, если соответствующие функции не включены.

1.4. Статус и границы описания

Настоящий документ является техническим описанием продукта АСТРА ИИ [ХАБ] и опирается на актуальное состояние технических заданий, репозиториях и архитектурной документации по состоянию на сентябрь 2026 года.

Версия документа и версия продукта

Версия настоящего документа не равна версии продукта. Документ описывает архитектуру, функции и интерфейсы платформы на основании технических заданий продуктового контура, задач и состояния платформенного контура исполнения моделей, детальных постановок и репозиториях обеих команд.

Конкретная версия продукта, состав поставки и перечень принятых функций определяются отдельно в документах на поставку и не подменяются утверждением настоящего описания.

Статус описания функций

Функции, описанные в настоящем документе, разделяются на следующие категории:

- **документально описанные** — функции, для которых зафиксированы требования в технических заданиях и архитектурных схемах; их наличие означает, что функция спроектирована, но не гарантирует факта реализации и приёмки в конкретном релизе;
- **опциональные** — функции, которые включаются конфигурацией при установке (подсистема хранения диалогов, защита содержимого Guardrails, автомасштабирование через KEDA, стек наблюдаемости VictoriaMetrics/Grafana);
- **целевые** — функции, включённые в дорожную карту расширения платформы (мультимодельная маршрутизация, промышленный админ-портал, автоматическое добавление пользователей, каскадная маршрутизация, мультимодальные группы, реестр MCP); их наличие в конкретном релизе должно проверяться отдельно перед включением в коммерческое предложение или договор;

- подтвержденные испытаниями — функции, прошедшие приемку и зафиксированные в акте приемки конкретного релиза; только эти функции могут безоговорочно заявляться как реализованные.

Границы описания

Настоящее описание не фиксирует и не подменяет следующие сведения, определяемые отдельными контрактами и документами поставки:

- коммерческая модель лицензирования и цены поставки;
- соглашения об уровне обслуживания (SLA) и целевые показатели доступности и восстановления;
- детальная многокластерная топология и схемы отказоустойчивости;
- полные пользовательские контракты MCP Hub, RAG-сервиса и маркетплейса Skills;
- точные REST-операции всех сущностей, SQL-схемы таблиц, имена топигов Kafka и схемы потоков ValKey;
- перечень поддерживаемых моделей и inference-движков в конкретной поставке;
- требования к GPU, оперативной памяти, дисковому пространству и объемам объектного хранилища в зависимости от планируемой нагрузки;
- интеграция с конкретной корпоративной системой идентификации заказчика;
- условия сопровождения, обновления и технической поддержки.

Эти сведения должны фиксироваться в технико-коммерческом предложении, спецификации поставки и договоре с заказчиком на основании версии конкретного релиза продукта.

2. Класс решаемых задач

АСТРА ИИ [ХАБ] решает задачи заказчика по организации централизованного корпоративного доступа к моделям искусственного интеллекта, управлению эксплуатацией моделей и контролю использования. Платформа создает единый управляемый контур между корпоративными пользователями и приложениями, с одной стороны, и моделями ИИ, и вычислительной инфраструктурой — с другой.

При масштабировании искусственного интеллекта в организации возникает фрагментация: разные подразделения используют разные модели, ключи доступа, внешние программные интерфейсы и способы развертывания. Это усложняет управление доступом, эксплуатацию и контроль использования. Платформа сводит эти задачи в единый корпоративный контур с централизованными политиками, учетом и наблюдаемостью.

2.1. Централизованный доступ к моделям

Платформа предоставляет единый программный интерфейс, совместимый с OpenAI, для доступа к локальным и внешним моделям. Приложения работают с единым корпоративным контрактом вместо множества разных интерфейсов.

Единая точка входа

Все запросы к моделям проходят через продуктовую точку входа `/api/v1/inference/{path}`, которая проксирует разрешенные запросы в OpenAI-совместимый шлюз. Шлюз поддерживает маршруты для языковых моделей (`chat/completions`, `completions`), векторных представлений (`embeddings`), переранжирования (`rerank`), аудио (`audio/*`) и списка доступных моделей. Поддерживаются потоковые ответы, структурированный вывод и вызовы инструментов.

Служебные маршруты шлюза не публикуются наружу. До обращения к модели платформа проверяет членство пользователя, наличие активной подписки, разрешенный список моделей, расписание доступности и квоту. Разрешены только зафиксированные маршруты из белого списка. Решение об отказе принимается до потребления вычислительных ресурсов моделей.

Поддерживаемые типы моделей

Платформа обслуживает языковые модели через движки vLLM, SGLang и Ollama. Аудиомодели и распознавание речи работают через движок FasterWhisper. Внешние модели, совместимые с OpenAI, подключаются в режиме внешнего поставщика без локальных подов исполнения.

Каталог моделей формируется из локальных развертываний и внешних источников. Пользователи и приложения получают доступ только к моделям, разрешенным их подпиской и организационными политиками.

Разделение доступа

Доступ к моделям строится вокруг организаций, пользователей, команд и ролей. Организация является основной единицей продуктовой изоляции. Инфраструктурно организации соответствует `tenant` платформы.

Корпоративная идентификация. Пользователи проходят аутентификацию через OIDC с использованием корпоративного поставщика личности DEX (Keycloak). Интерактивный вход использует Authorization Code с PKCE. После входа личность пользователя передается только доверенными заголовками, установленными на стороне платформы. Клиент не может подставить заголовки личности или ключ запросов.

Организации и команды. Организация объединяет пользователей, роли, подписку и лимиты в изолированный контур. Команды группируют пользователей внутри организации и задают область применения отдельных политик. Состав организации синхронизируется с поставщиком личности.

Роли и права. Авторизацию операций управления выполняет серверная часть платформы. Роли организации задаются набором разрешений из фиксированного каталога. Встроенная защищенная роль администратора организации предоставляет полный доступ в контуре организации. Права платформенного администратора (управление развертываниями, конфигурациями моделей) отделены от ролей организации и доступны только администратору платформы без обязательного членства в организации.

Персональные ключи. Каждый активный пользователь организации получает персональный ключ запросов формата `sk-<8hex>-<32hex>`. Ключ генерирует и учитывает продуктовый контур. Полный секрет показывается не больше одного раза. Смена ключа делает старый недействительным. Отзыв ключа отключает доступ клиента к моделям. Секрет не попадает в журналы и ответы кластерного программного интерфейса.

Сервисные ключи. Машинный доступ от имени организации предоставляется через сервисные ключи без привязки к конкретному пользователю. В базе данных хранится хеш секрета. Число сервисных ключей ограничено тарифом организации. На запросы с сервисным ключом действуют только политики организации без проверки членства в команде.

Подписка. Подписка определяет доступ организации к каталогу моделей и квотам в продуктивном контуре. Без активной подписки запросы к моделям и каталог моделей организации недоступны. Права управления организацией при отсутствии подписки сохраняются. Оформление подписки регистрирует `tenant` платформы и обеспечивает пользовательские ключи запросов для активных членов организации.

2.2. Управление использованием ресурсов

Платформа позволяет ограничивать потребление ресурсов моделей и связывать доступ с ограничениями организации и клиента.

Тарифные ограничения и квоты

Тариф задает для организации лимиты числа пользователей и сервисных ключей, квоты потребления по моделям и категориям токенов, разрешенный список моделей и цену за период. Каталог тарифов ведет администратор платформы.

Квоты проверяются на уровне `tenant` (организации) и клиента программного интерфейса. Поддерживаются окна использования с первого запроса: 5 часов, 1 день, 1 неделя. При невозможности зарезервировать квоту запрос не передается в вычислительный контур моделей.

Лимиты запросов

Платформа контролирует следующие параметры использования:

- RPM — ограничение числа запросов в минуту;
- TRM — ограничение числа токенов в минуту;
- Параллельные запросы — потолок одновременных запросов для организации и клиента;
- Категории токенов — учет входных токенов, выходных токенов, чтения из кэша, записи в кэш и итога без двойного счета.

При превышении или исчерпании лимита платформа возвращает HTTP 429 с описанием превышенного ограничения. Продуктовый поток строится по схеме «резерв квоты — вызов шлюза — фиксация фактического потребления». События идут через `ValKey`, фоновый обработчик записывает данные в PostgreSQL.

Расписания доступности

Платформа поддерживает два связанных слоя расписаний. На уровне управления организацией задаются окна доступа организации, команды или пользователя к запросам до вызова модели. На уровне управления моделями устанавливается расписание модели для tenant или пользователя в ресурсах; вне окна освобождаются графические процессоры.

Учет и анализ потребления

Учет потребления. Учет ведется по категориям: вход, выход, чтение кэша, запись кэша и итог без двойного счета. Стоимость рассчитывается по категориям и модели на основании цен модели. Цены для кэшированных токенов задаются отдельно. Продуктовый журнал запросов фиксирует категории и квоты тарифа. При отсутствии стоимости от шлюза в аналитике отображается отсутствие данных.

Аналитика использования. Источником данных служит продуктовый журнал PostgreSQL. Участник организации видит свои запросы, администратор организации — данные организации, администратор платформы — любой организации. Аналитика включает сводку, разрезы по пользователю и модели, список и детали запросов, выгрузку CSV, фильтры периода, пользователя и модели.

Метрики. Метрики оператора и шлюза отдаются на отдельных точках для систем наблюдаемости. Панели мониторинга включают число запросов в секунду, задержки первого и последующих токенов, сквозную задержку, ошибки, использование токенов, состояние графических процессоров и очередь запросов.

Аудит. Структурированный журнал безопасности фиксирует действующее лицо, операцию, время и результат без записи секретных ключей, тел запросов и промптов. Сквозной идентификатор запроса обеспечивает корреляцию учета использования, аудита и разбора инцидентов.

2.3. Эксплуатация моделей

Платформа управляет жизненным циклом локальных моделей в Kubernetes. Администратор работает через продуктовый программный интерфейс и не обращается к Kubernetes напрямую из консоли.

Декларативное управление развертыванием

Администратор платформы ведет каталог конфигураций развертывания. Конфигурация создается, копируется, сравнивается с предпросмотром спецификации без создания ресурса в кластере. Некорректная спецификация отсекается до обращения к кластеру.

Создание развертывания регистрирует ресурс модели через HTTP-адаптер с именем формата `hubd-<name>-<org uid>`. Статусы, журналы, события, метрики, перечень графических процессоров и оценка размещения доступны через продуктовый программный интерфейс. Консоль не обращается к программному интерфейсу кластера напрямую.

Жизненный цикл модели

Платформенный контур исполнения моделей согласует фактическое состояние с желаемым через оператор Kubernetes. Оператор идемпотентно обеспечивает кэш и том весов, поды исполнения и статус ресурса модели. При смене шаблона пода действует безопасное для графических процессоров обновление: сначала создается и доводится до готовности новая реплика, затем удаляется старая.

Продуктовый контур АСТРА ИИ задает управление и предоставляет сведения о состоянии. Платформенный контур исполнения моделей согласует ресурсы и обслуживает модели в Kubernetes. Граница проходит по HTTP-адаптеру и пользовательским ресурсам.

Загрузка и кэш весов

Весы размещаются на постоянном томе атомарно через временный каталог с признаком готовности. Поддерживаются источники HuggingFace, зеркала и хранилища, совместимые с S3. Учетные данные передаются только через секрет Kubernetes.

Графические процессоры и автомасштабирование

Продуктовый программный интерфейс показывает перечень графических процессоров и предварительную оценку размещения. Переход между нулем и одной репликой выполняет оператор. Масштабирование от одной реплики к нескольким обеспечивает KEDA на основе метрик Prometheus или VictoriaMetrics.

Диагностика и наблюдаемость

Работоспособность компонентов проверяется отдельными проверками живости и готовности с учетом необходимых зависимостей режима. Серверная часть развертывается в одном из режимов: только управление, только исполнение запросов к моделям или оба сразу. Проверка живости отвечает на запусженность процесса без обращения к внешним сервисам. Проверка готовности смотрит доступность зависимостей нужного режима: база данных и ValKey для учета и кэша, шлюз моделей для исполнения, HTTP-адаптер для управления.

2.4. Контроль безопасности и данных

Платформа применяет принцип минимальных привилегий и контролирует доступ до вызова модели.

Контроль доступа

Доверенная личность. Пользователь аутентифицируется на входном шлюзе. Клиентские заголовки личности заменяются доверенными. Клиент не может подставить заголовки личности или управлять адресом сервиса, в который будет отправлен запрос к модели.

Белый список маршрутов. По умолчанию запрещено все, что не входит в белый список маршрутов шлюза. Неподдерживаемые и служебные маршруты не публикуются наружу.

Серверно заданный upstream. Клиент не управляет адресом сервиса, в который будет отправлен запрос к модели. Вышестоящий адрес задается только конфигурацией на стороне сервера.

Секреты. Секреты не попадают в спецификации, статусы ресурсов и журналы. Передача учетных данных выполняется через ссылки на секреты Kubernetes. Хеширование ключей в инфраструктурном контуре использует argon2id.

Идемпотентность. Повторные операции согласования и доставки не приводят к двойной обработке. Внешняя ошибка не дает тихого рассинхрона или двойной обработки.

Защита содержимого

Платформа поддерживает включаемый слой защиты содержимого (Guardrails) на шлюзе моделей. По умолчанию отключен и не изменяет трафик запросов к моделям, пока не включен в поставке.

Политики защиты. Политики описываются ресурсом GuardrailPolicy и привязываются к модели или клиенту. Оператор формирует эффективный набор и отражает его в статусе модели. Шлюз читает только этот набор без прямого доступа к политикам уровня кластера.

Фазы применения. Политики применяются в фазах `pre_call` (до вызова модели), `during_call` (перед проксированием), `post_call` (ответ модели, включая потоковые ответы) и `logging_only` (только аудит). Действия включают блокировку (`block`), маскирование (`mask`), вмешательство (`intervene`) и аудит (`audit`).

Встроенные детекторы. Платформа содержит детекторы персональных данных (`pii`), секретных ключей и токенов (`secret`), запрещенных формулировок (`denylist`) и внедрения промпта (`prompt_injection`). Поддерживается подключение внешнего сервиса проверки по контракту Generic Guardrail API. Адрес внешнего сервиса задается только конфигурацией платформы для защиты от подделки запросов на стороне сервера.

Обязательные политики. Политики с признаком `defaultOn` обязательны: клиент не может отключить их параметрами запроса. Попытка снять обязательную политику отклоняется. Фактическая блокировка контента нормализуется как отказ политики безопасности, а не как ошибка модели.

Защита от внедрения промпта. Детектор `prompt_injection` выполняет эвристический поиск сигнатур и шаблонов (`ignore/disregard previous instructions`, `override/reveal system prompt`, `developer/admin mode`, `jailbreak`, `DAN mode`) с НАСТРАИВАемым порогом срабатывания и собственным набором сигнатур заказчика. Типовое действие — блокировка запроса до вызова модели (`block`) или фиксация без блокировки (`audit`). Детектор рекомендуется включать в фазе `pre_call` в составе обязательной базовой политики безопасности организации. Он дополняет, а не заменяет, разграничение ключей, белый список маршрутов, серверный upstream и аудит.

Режим отказа. Режим отказа детектора по умолчанию закрытый (`fail closed`). Открытый режим (`fail-open`) допускается только при явной конфигурации и обязан оставлять наблюдаемый след.

События срабатывания защиты содержимого фиксируются в аудите и метриках без записи тел промптов и ответов.

События и уведомления

События запросов. После запроса к модели формируется событие и публикуется в Kafka. Технические метрики (учет, статус, задержка) собираются всегда. Содержимое записывается только по политике. Сбой Kafka не блокирует ответ клиенту. Повтор с тем же идентификатором запроса идемпотентен для получателя.

Исходящие уведомления. Шлюз отправляет события приближения к лимиту, исчерпания квоты, восстановления доступа и изменения состояния модели. Адрес, типы событий и включение задаются в ресурсах. Повторная доставка использует стабильный идентификатор события. Сбой получателя не прерывает запрос к модели.

2.5. Дополнительные и перспективные задачи

Ниже перечислены направления, заявленные в продуктивном контуре или обозначенные в дорожной карте. Их наличие и готовность должны проверяться по составу конкретного релиза.

Единый MCP Hub, RAG-сервис и маркетплейс Skills

Версия 3 технического описания включает в продуктивный контур доступ к единому MCP Hub, RAG-сервису и маркетплейсу Skills. Документ не определяет их полный набор пользовательских функций, программные интерфейсы, правила публикации контента или эксплуатационные параметры. Детальный состав поставки и пользовательские контракты фиксируются отдельно.

Мультимодальность

В ресурсе модели задаются входные и выходные типы данных. В учете учитываются токены, число изображений и длительность аудио или видео. Поддерживаются текстовый контур и распознавание речи с учетом аудиосекунд.

Задачи машинного обучения на центральных и графических процессорах

Пользовательский сценарий запускается на выделенных центральных или графических процессорах через пользовательский ресурс. Статусы включают ожидание, выполнение, завершение, ошибку и остановку. В статусе фиксируются машино-часы центральных и графических процессоров, тип графического процессора и длительность. После завершения или остановки ресурсы освобождаются согласованием состояния.

Развитие платформы

Дорожная карта расширяет ядро платформы следующими направлениями:

Шлюз. Мультимодельная маршрутизация, промышленный программный интерфейс, потоковая выдача.

Портал и единый вход. Административный портал, корпоративный вход и автоматическое добавление пользователей.

Изоляция. Изоляция tenant, ключи и реестр моделей.

Учет. Аналитика, квоты и повышение точности учета токенов с кэшем.

Наблюдаемость. Сквозной идентификатор запроса, панели нагрузки, задержек и графических процессоров.

Приватность. Режимы «только контур», «гибрид» и «обезличивание».

Расширения. Каскадная маршрутизация, мультимодальные группы и реестр МСР.

Мультитенантность. Иерархическая структура организаций.

Адаптация для ADP. Интеграция с платформой данных и приложений.

Гибкие лимиты. Расширенные механизмы управления потреблением.

Расписание доступности модели. Управление доступностью моделей по времени с освобождением графических процессоров.

Сбор данных. Политика сбора данных в ресурсах.

Состав функций дорожной карты проверяется по версии конкретного релиза перед включением в коммерческое предложение или договор.

3. Архитектура

АСТРА ИИ [ХАБ] построена как двухконтурная платформа с чётким разделением ответственности. Продуктовый контур решает задачи управления доступом, политиками и учётом использования. Платформенный контур исполнения моделей обеспечивает жизненный цикл моделей, вычислительные ресурсы и инфраструктуру обслуживания. Между контурами действуют интеграционные контракты и совместные компоненты с разделением функций.

3.1. Два контура и совместный слой

Архитектура платформы делит функции между двумя контурами разработки и эксплуатации.

Продуктовый контур

Продуктовый контур охватывает всё, что касается пользовательского управления, корпоративных политик и учёта:

Консоль управления — веб-интерфейс администрирования организаций, пользователей, команд, ролей, тарифов, подписок, ключей доступа, квот, расписаний, конфигураций развёртывания моделей и аналитики использования. Консоль работает только через продуктовый программный интерфейс и не обращается к программному интерфейсу кластера напрямую.

Входной шлюз APISIX — точка входа для всех запросов к платформе. Выполняет аутентификацию пользователей через OIDC с поставщиком личности DEX (Keycloak) и проверку сервисных ключей для машинного доступа. Клиентские заголовки личности заменяются доверенными на стороне шлюза, что исключает подмену личности клиентом. Для интерактивных пользователей применяется Authorization Code с PKCE. Для подключений WebSocket (консоль развёртываний) используется подпротокол bearer, <token>.

Система идентификации DEX — корпоративный поставщик личности на базе Keycloak, обеспечивающий интеграцию с внутренней системой идентификации заказчика по протоколу OIDC. Состав организации синхронизируется с поставщиком личности.

Серверная часть продукта — ядро продуктового контура, реализующее организации, пользователей, команды, роли, тарифы, подписки, ключи запросов к моделям, политики доступа, расписания, квоты, продуктовый журнал запросов, аналитику и учёт потребления. Серверная часть может работать в одном из режимов: только управление (консоль, организации, развёртывания моделей, политики), только исполнение запросов к моделям или оба режима одновременно.

Продуктовая точка входа запросов — маршрут `/api/v1/inference/{path}` проверяет членство пользователя в организации, активность подписки, список разрешённых моделей, расписание доступа и квоты до вызова модели. После успешных проверок разрешённые запросы маршрутизируются в шлюз запросов к моделям с установкой внутреннего служебного заголовка.

HTTP-адаптер — мост между REST-интерфейсом продукта и пользовательскими ресурсами Kubernetes. Адаптер обеспечивает декларативное управление развёртываниями моделей, арендаторами платформы, клиентами программного интерфейса и вычислительными ресурсами без прямого обращения консоли к программному интерфейсу кластера. Вызывается только серверной частью продукта.

Продуктовый контур также предоставляет доступ к дополнительным сервисам экосистемы: единому MCP Hub, RAG-сервису и маркетплейсу Skills. Детальный набор функций, программные интерфейсы и правила публикации контента определяются отдельно в зависимости от состава конкретной поставки.

Платформенный контур исполнения моделей

вычислительные ресурсы:

Оператор Kubernetes — согласует фактическое состояние ресурсов моделей с желаемым. Управляет кэшем весов, томами данных, подами исполнения и статусами. При изменении шаблона пода выполняет безопасное для графических процессоров обновление: сначала создаёт и доводит до готовности новую реплику, затем удаляет старую. Обеспечивает переход между нулём и одной репликой при поступлении запросов.

Пользовательские ресурсы Kubernetes — декларативные описания моделей: конфигурации моделей, профили исполнения, классы исполнения, кэш весов, арендаторы платформы, клиенты программного интерфейса, цены моделей и политики защиты содержимого. Обеспечивают согласование состояния без прямого управления подами исполнения.

Загрузчик весов моделей — загружает веса из репозитория HuggingFace, зеркал и хранилищ, совместимых с S3, в кэш на постоянном томе. Веса размещаются атомарно через временный каталог с признаком готовности. Учетные данные передаются только через секреты Kubernetes.

Среда исполнения — обеспечивает обслуживание моделей через движки vLLM, SGLang, Ollama для языковых моделей и FasterWhisper для распознавания речи. Поддерживает режим внешнего поставщика для проксирования запросов к OpenAI-совместимым внешним сервисам без локальных подов исполнения.

Автомасштабирование — переход от нуля к одной реплике выполняет оператор при поступлении первого запроса. Масштабирование от одной реплики к нескольким обеспечивает KEDA на основе метрик нагрузки из Prometheus или VictoriaMetrics. Без установленного KEDA модели работают с фиксированным числом реплик.

Средства наблюдаемости — интеграция с Prometheus, VictoriaMetrics, Grafana, Loki, OpenTelemetry и DCGM для мониторинга графических процессоров. Оператор и шлюз экспортируют метрики на отдельных точках для систем наблюдаемости.

Совместные компоненты

Некоторые компоненты используются обоими контурами с разделением ответственности по функциям:

Шлюз запросов к моделям — точка маршрутизации OpenAI-совместимых запросов без хранения состояния. Поддерживает маршруты /v1/chat/completions, /v1/completions, /v1/embeddings, /v1/audio/*, /v1/rerank, /v1/models. Функции категорийного учёта токенов, быстрых лимитов, расписаний доступности моделей, включаемых политик защиты содержимого и исходящих уведомлений реализуются на стороне продуктового контура. Инфраструктурная поставка и согласование состояния — задача платформенного контура.

PostgreSQL — используется обоими контурами с разделением по схемам и ролям. Продуктовый журнал запросов, учёт потребления и аналитика ведётся продуктовым контуром. Учёт платформы и хранение диалогов при включении соответствующей функции — платформенным контуром.

ValKey — кэш ключей доступа для быстрой проверки на входном шлюзе APISIX, резервы квот, события учёта и кэш контекста исполнения. После изменения ключей доступа обновляется кэш определителя ключей на APISIX из ValKey с запасным чтением PostgreSQL при недоступности кэша.

Kafka — поток событий запросов к моделям. После каждого запроса к модели формируется событие и публикуется в Kafka. Технические метрики (учёт, статус, задержка) собираются всегда; содержимое запроса и ответа записывается только по политике. Сбой Kafka не блокирует ответ клиенту. Повторная доставка использует стабильный идентификатор события для идемпотентной обработки получателем.

3.2. Управление и выполнение запросов

Платформа разделяет плоскость управления и плоскость исполнения запросов.

Управление моделями

Администратор платформы управляет моделями через продуктовый программный интерфейс без прямого обращения к Kubernetes:

1. **Консоль** отправляет запрос создания или изменения конфигурации развёртывания в продуктовый программный интерфейс.

2. **Серверная часть продукта** проверяет права администратора платформы, валидирует спецификацию и формирует запрос к HTTP-адаптеру. Некорректная спецификация отсекается до обращения к кластеру.
3. **HTTP-адаптер** преобразует REST-запрос в создание или изменение пользовательского ресурса модели в Kubernetes. Имя ресурса формируется по шаблону `hubd-<name>-<org uid>`.
4. **Оператор Kubernetes** обнаруживает изменение желаемого состояния и согласует фактическое состояние: обеспечивает кэш и том весов, создаёт или обновляет поды исполнения, устанавливает статус ресурса модели.
5. **Статусы, журналы, события и метрики** модели доступны через продуктовый программный интерфейс. Серверная часть читает статусы через HTTP-адаптер, консоль получает информацию через продуктовый программный интерфейс.

Этот путь обеспечивает декларативное управление развёртываниями без ручного создания подов и сервисов.

Путь запроса к модели

Запрос клиента к модели проходит через несколько слоёв проверок и маршрутизации:

1. **Входной шлюз APISIX** выполняет аутентификацию. Для интерактивного пользователя проверяется токен OIDC, для машинного доступа — сервисный ключ в заголовке X-API-Key. Клиентские заголовки личности заменяются доверенными на стороне шлюза.
2. **Продуктовая точка входа** `/api/v1/inference/{path}` принимает запрос с доверенными заголовками личности. Для запросов с персональным ключом проверяется членство пользователя в организации и политики команды. Для запросов с сервисным ключом применяются только политики организации без проверки членства пользователя.
3. **Проверки до вызова модели** выполняются на стороне серверной части продукта в следующем порядке:
 - членство пользователя в организации или действительность сервисного ключа;
 - наличие активной подписки;
 - присутствие запрашиваемой модели в списке разрешённых моделей тарифа;
 - соответствие расписанию доступности организации, команды или пользователя;
 - соблюдение квот и лимитов организации и клиента.
4. **Резерв квоты** выполняется перед вызовом модели. Резерв записывается в ValKey. При невозможности зарезервировать квоту запрос не передаётся в вычислительный контур моделей. Отказ резерва происходит до потребления вычислительных ресурсов.
5. **Вызов шлюза запросов** происходит только после успешного резерва. Запрос передаётся с внутренним служебным заголовком. Разрешены только

зафиксированные маршруты из белого списка. Служебные маршруты шлюза не публикуются наружу.

6. **Шлюз запросов к моделям** маршрутизирует запрос в поды исполнения локальных моделей или к внешним поставщикам. Балансировка выполняется по наименьшей нагрузке либо с привязкой по префиксу запроса для попадания в кэш контекста. При холодном старте запрос удерживается в пределах отведённого времени ожидания; за пределами этого срока возвращается код состояния 503.
7. **Под исполнения модели** обрабатывает запрос через движок vLLM, SGLang, Ollama или FasterWhisper. Если модель находится в состоянии холодного старта, оператор создаёт реплику и доводит её до готовности. Ответ модели содержит категории использования токенов: вход, выход, чтение кэша, запись кэша и итог без двойного счёта.
8. **Фиксация фактического потребления** выполняется после получения ответа от модели. Резерв квоты освобождается, в продуктовый журнал записывается реальное потребление по категориям из ответа шлюза. События резерва и фиксации проходят через ValKey; фоновый обработчик проецирует их в PostgreSQL.
9. **Ответ клиенту** возвращается с сохранением потокового режима, если запрошен. Потоковый ответ включает сведения об использовании токенов, в том числе в финальном фрагменте события отправки сервера. Поддерживаются структурированный вывод и вызовы инструментов.

При отказе любой проверки на этапах 3–4 запрос к модели не выполняется. Отказ политики безопасности не считается ошибкой модели. При превышении или исчерпании лимита платформа возвращает HTTP 429 с описанием превышенного ограничения.

Расписания доступности

Платформа поддерживает два связанных слоя расписаний:

Расписания организации задаются на уровне управления организацией. Определяют окна доступа организации, команды или пользователя к запросам до вызова модели. Вне окна доступа запрос отклоняется на этапе проверки политики доступа без обращения к вычислительному контуру.

Расписания моделей задаются на уровне управления моделями в пользовательских ресурсах платформы. Устанавливают расписание модели для арендатора или пользователя. Вне окна доступности платформа может освобождать графические процессоры путём масштабирования числа реплик до нуля.

3.3. Хранилища и события

Платформа использует несколько хранилищ и потоков событий с разделением ответственности.

Продуктовый журнал запросов

Продуктовый журнал в PostgreSQL фиксирует метаданные каждого запроса к модели: организацию, пользователя, модель, время запроса, статус выполнения, категории использования токенов и стоимость по тарифу. Журнал не содержит секретных ключей, тел запросов и промптов. Источником данных служит продуктовый журнал PostgreSQL. Участник организации видит свои запросы, администратор организации — данные организации, администратор платформы — любой организации.

Стоимость рассчитывается по категориям токенов и модели на основании цен модели. Цены для кэшированных токенов задаются отдельно. При отсутствии стоимости от шлюза в аналитике отображается отсутствие данных.

Учёт и диалоги платформы

Платформенный контур ведёт собственный учёт в PostgreSQL при включении соответствующей функции. Поддерживается опциональное хранение содержимого диалогов в PostgreSQL или хранилище, совместимом с S3. Включение хранения задаётся политикой в пользовательских ресурсах. При отсутствии политики хранения содержимое запросов и ответов не сохраняется; продуктовый журнал фиксирует только метаданные запроса и категории использования токенов.

Резервы и кэш ValKey

ValKey используется для нескольких целей с высокими требованиями к скорости:

Кэш ключей доступа на входном шлюзе APISIX обеспечивает быструю проверку действительности ключей без обращения к базе данных на каждом запросе. При недоступности ValKey выполняется запасное чтение из PostgreSQL.

Резервы квот фиксируются в ValKey перед вызовом модели. Резерв позволяет предотвратить превышение квоты без блокировок в базе данных. После получения ответа от модели резерв освобождается и записывается фактическое потребление.

События учёта проходят через ValKey и проецируются в PostgreSQL фоновым обработчиком. Это обеспечивает низкую задержку пути запроса и гарантированное сохранение данных учёта.

Кэш контекста исполнения хранит промежуточное состояние диалогов для ускорения повторных запросов с общим префиксом.

Поток событий Kafka

После каждого запроса к модели формируется событие и публикуется в Kafka. Технические метрики (учёт, статус, задержка) собираются всегда; содержимое запроса и ответа записывается только по политике сбора данных. Сбой Kafka не блокирует ответ клиенту. Повторная доставка использует стабильный идентификатор события для идемпотентной обработки получателем.

Поток событий используется для интеграции с внешними системами аналитики, дополнительного аудита и обработки данных без нагрузки на основной путь запросов.

Исходящие уведомления

Шлюз запросов отправляет исходящие уведомления о приближении к лимиту, исчерпании квоты, восстановлении доступа и изменении состояния модели. Адрес получателя, типы событий и включение уведомлений задаются в пользовательских ресурсах. Повторная доставка использует стабильный идентификатор события; сбой получателя не прерывает запрос к модели.

3.4. Внешние интерфейсы и границы доверия

Платформа определяет чёткие границы доверия и контракты внешних интерфейсов.

Продуктовый программный интерфейс

Маршруты `/api/v1/*` предоставляют операции управления для консоли и внешних клиентов:

- организации, пользователи, команды, роли и права;
- тарифы, подписки и ключи запросов к моделям;
- конфигурации развёртывания моделей и управление жизненным циклом;
- расписания доступности и квоты потребления;
- аналитика использования и журнал аудита;
- настройки клиентских приложений и каталог доступных моделей.

Авторизацию операций управления выполняет серверная часть продукта. APISIX аутентифицирует пользователя через OIDC или сервисный ключ, но не проверяет роли организации.

Точка входа запросов к моделям

Маршрут `/api/v1/inference/{path}` является единственной точкой входа для запросов к моделям со стороны клиентов. Разрешены только зафиксированные пути из белого списка: `/v1/chat/completions`, `/v1/completions`, `/v1/embeddings`, `/v1/audio/*`, `/v1/gerank`, `/v1/models`. Маршруты вне белого списка отклоняются. Служебные пути шлюза не публикуются наружу.

Клиент не управляет адресом сервиса, в который будет отправлен запрос к модели. Вышестоящий адрес задаётся только конфигурацией на стороне сервера. Это предотвращает подделку запросов на стороне сервера и обход политик доступа.

Внутренний контракт с шлюзом

После успешных проверок продуктовая точка входа передаёт запрос шлюзу с установкой внутреннего служебного заголовка. Этот заголовок не доступен клиентам и устанавливается только на стороне сервера после определения и проверки пользовательского или сервисного ключа.

Шлюз запросов к моделям принимает запросы только с корректным внутренним заголовком. Прямое обращение клиентов к шлюзу блокируется сетевыми политиками и отсутствием публичной маршрутизации.

Контракт HTTP-адаптера

HTTP-адаптер предоставляет REST-интерфейс для операций с пользовательскими ресурсами Kubernetes:

- создание, изменение, удаление и чтение ресурсов моделей;
- регистрация арендаторов платформы и клиентов программного интерфейса;
- управление внешними поставщиками моделей;
- получение статусов, журналов, событий и метрик;
- оценка доступности графических процессоров для размещения модели;
- подключение WebSocket для потоковых журналов и событий.

Адаптер вызывается только серверной частью продукта с токеном Bearer. Консоль и внешние клиенты не обращаются к адаптеру напрямую.

Доверенная личность

После прохождения APISIX личность пользователя передаётся только доверенными заголовками вида X-User-*. Входной шлюз перезаписывает любые клиентские заголовки личности на доверенные значения после успешной аутентификации. Клиент не может подставить заголовки личности или управлять адресом сервиса, в который будет отправлен запрос к модели.

Этот принцип обеспечивает невозможность подделки личности на стороне клиента и гарантирует, что все проверки политик доступа выполняются на стороне сервера с проверенной личностью.

3.5. Варианты размещения и целевая схема доступа

Из единой кодовой базы поддерживаются несколько вариантов размещения, различающихся конфигурацией.

Контур заказчика

Базовый сценарий размещения: платформа и локальные модели разворачиваются в Kubernetes-кластере заказчика. Все данные запросов, учёта и аналитики остаются в инфраструктуре заказчика. При использовании только локальных моделей данные не покидают периметр заказчика.

Базовая модель размещения использует Kubernetes версии не ниже 1.29 и Helm для установки компонентов платформы. Плоскость управления и плоскость исполнения запросов к моделям масштабируются независимо. Базовая поставка (оператор, шлюз запросов, пользовательские ресурсы для жизненного цикла моделей) работоспособна без необязательных подсистем (учёт на стороне платформы, хранение диалогов, защита содержимого, KEDA, VictoriaMetrics, сетевые политики).

Облачный вариант

Платформа может разворачиваться в облачной инфраструктуре с той же кодовой базой и набором компонентов. Детальная облачная топология определяется конфигурацией и не зафиксирована как отдельная архитектурная ветвь. Разделение плоскостей управления и исполнения позволяет масштабировать их независимо в зависимости от нагрузки.

Изолированный стенд

Отдельная конфигурация для закрытого или автономного окружения, где требуется полная изоляция от внешних сетей и сервисов. В этом варианте все зависимости (поставщик личности, хранилища, реестры контейнеров) размещаются внутри периметра заказчика.

Целевая схема внешнего доступа

Целевая схема доступа определяет три входных контура для подключения клиентов к платформе:

Внешние клиенты Astra Code подключаются через опциональный компонент Termidesk Connect к входному шлюзу APISIX, затем к продуктовой точке входа запросов к моделям. Передают служебный ключ формата `sk-<8hex>-<32hex>` для аутентификации. Termidesk Connect не входит в обязательный минимум базовой поставки. При включении этой опции он используется как точка безопасного подключения внешних клиентов к APISIX и как возможный канал к облачным и внешним контурам моделей.

Внутренние клиенты Astra Code подключаются к APISIX через корпоративную сеть или VPN с аутентификацией по токenu JWT. Маршруты `/v1/*` доступны после успешной проверки токена и установки доверенных заголовков личности. Запросы направляются к продуктовой точке входа, затем к шлюзу запросов к моделям.

Клиенты вне экосистемы Astra Code подключаются к шлюзу запросов к моделям по VPN с сервисным ключом организации. При таком подключении применяются только политики организации без проверки членства пользователя и команды.

От шлюза запросов запросы направляются к локальным вычислительным контурам (графические процессоры через балансировщики нагрузки и менеджер виртуальных машин) и, при необходимости, через опциональный Termidesk Connect к облачным и иным внешним контурам моделей.

Без Termidesk Connect внешний доступ организуется иными согласованными средствами периметра; внутренние клиенты Astra Code подключаются к APISIX по VPN с JWT, а клиенты вне экосистемы Astra Code — к шлюзу запросов по VPN с сервисным ключом.

Ключевые инфраструктурные зависимости

Для работы платформы требуются следующие компоненты инфраструктуры:

Компонент	Роль	Обязательность
Kubernetes ≥ 1.29	Оркестрация, пользовательские ресурсы	Базовая
Helm	Установка платформы	Базовая
PostgreSQL	Продуктовый журнал запросов; учёт и диалоги платформы	Для учёта и аналитики
ValKey	Кэш ключей, резервы	Промышленный

Компонент	Роль	Обязательность
	квот, события учёта	продуктовый режим
S3-совместимое хранилище	Веса моделей, холодное хранение диалогов	По функции
KEDA / Prometheus / VictoriaMetrics	Автомасштабирование $1 \leftrightarrow N$, метрики	По необходимости
DEX (Keycloak)	Поставщик корпоративной идентификации	Для консоли и OIDC-клиентов

Состав зависимостей определяется конфигурацией поставки: базовый оператор и шлюз запросов работают без KEDA, VictoriaMetrics и подсистемы хранения диалогов, если соответствующие функции не включены.

4. Основные свойства продукта

4.1. Единый доступ и совместимость

АСТРА ИИ [ХАБ] предоставляет единую точку входа для работы с моделями искусственного интеллекта через OpenAI-совместимый программный интерфейс. Продукт не привязан к единственной модели или поставщику — поддерживаются локальные развёртывания (vLLM, SGLang, Ollama, FasterWhisper) и внешние источники с совместимым API.

Поддерживаемые маршруты

Продуктовая точка входа `/api/v1/inference/{path}` направляет запросы в шлюз запросов к моделям, который поддерживает следующие маршруты:

- `/v1/chat/completions` — диалоговые запросы с потоковой выдачей;
- `/v1/completions` — одиночное завершение текста;
- `/v1/embeddings` — векторные представления;
- `/v1/audio/*` — распознавание и обработка речи;
- `/v1/rerank` — переупорядочение результатов;
- `/v1/models` — перечень доступных моделей.

Маршруты вне белого списка отклоняются. Служебные пути шлюза наружу не публикуются.

Потоковая выдача ответов

Потоковый ответ включает сведения об использовании токенов, в том числе в финальном фрагменте события отправки сервера (SSE). Поддерживаются структурированный вывод и вызовы инструментов. При отсутствии потоковой передачи у внешнего поставщика ядро ожидает полный ответ и передаёт его целиком.

Границы совместимости

Совместимость с расширениями OpenAI API зависит от возможностей конкретной модели и движка исполнения. Платформа не гарантирует поддержку всех параметров запроса всеми моделями. Балансировка выполняется по наименьшей

нагрузке либо с привязкой по префиксу запроса для попадания в кэш контекста. При холодном старте запрос удерживается в пределах отведённого времени ожидания; за пределами этого срока возвращается код состояния 503.

4.2. Изоляция организаций и управление правами

Организация — единица продуктовой изоляции: пользователи, роли, подписка, квоты и лимиты действуют в границах организации. Каждой организации соответствует tenant платформы с именем вида `hub-org-<uuid>`, создаваемый при оформлении подписки.

Идентификация и аутентификация

Продукт поддерживает два канала входа:

- Интерактивный вход пользователя — через протокол OIDC с поставщиком личности Keycloak, Authorization Code + PKCE. Проверка токена выполняется на APISIX; назначение токена — серверная часть продукта. Для WebSocket-подключений (консоль развёртываний) применяется подпротокол bearer,<token>.
- Машинный доступ сервисным ключом — заголовок X-API-Key на маршруте запросов к моделям без привязки к пользователю и команде.

После прохождения APISIX личность передаётся только доверенными заголовками вида X-User-*. Клиент не может подменить эти заголовки или напрямую передать ключ запросов к моделям; недоверенные заголовки перезаписываются.

Разграничение прав

Авторизацию операций управления выполняет серверная часть продукта. APISIX аутентифицирует пользователя или сервисный ключ, но не проверяет роли организации.

Права организации задаются ролями с набором разрешений из фиксированного каталога. Роли создаются в рамках организации. Существует встроенная защищённая роль администратора организации с полным доступом в контуре организации. Через роли организации нельзя выдать права платформы (управление развёртываниями моделей, конфигурации, связанные операции) — они доступны только администратору платформы без обязательного членства в организации.

Команда — группа пользователей внутри организации. Команда определяет область действия политик доступа; пользователь может входить в несколько команд одной организации.

Проверки до обращения к модели

Каждый запрос к модели проходит следующие проверки на стороне серверной части продукта до вызова модели:

1. Членство пользователя в организации или действительность сервисного ключа;
2. Наличие активной подписки;
3. Присутствие запрашиваемой модели в списке разрешённых моделей тарифа;
4. Соответствие расписанию доступности (раздел 4.2.1);
5. Соблюдение квот и лимитов (раздел 4.3).

При отказе любой проверки запрос к модели не выполняется. Отказ политики безопасности не считается ошибкой модели.

4.2.1. Расписания доступности

Продукт поддерживает два связанных, но различных уровня расписаний:

- Расписания организации — окна доступа организации, команды или пользователя к запросам до вызова модели. Вне окна доступа запрос отклоняется на этапе проверки политики доступа.
- Расписания моделей — расписание модели, tenant или пользователя в пользовательских ресурсах платформы. Вне окна платформа может освобождать вычислительные ресурсы GPU.

Ключи запросов к моделям

Ключи запросов генерируются и учитываются продуктовым контуром; платформе передаётся готовое значение.

Пользовательский ключ

- формат sk-<8hex>-<32hex>;
- имена: tenant hub-org-<uuid>, клиент API hub-user-<uuid>;
- привязка к tenant неизменяема;
- полный секрет предоставляется пользователю не больше одного раза при создании;
- смена ключа делает старый недействительным; отзыв отключает клиента API;
- после изменений обновляется кэш определителя ключей на APISIX из ValKey с запасным чтением PostgreSQL;
- секрет не попадает в журналы и ответы кластерного API.

Сервисный ключ организации

Машинный доступ от имени организации без привязки к пользователю и команде. В базе данных хранится хеш секрета. Число ключей ограничено тарифом; при параллельном создании применяются блокировки для предотвращения превышения лимита. На запросы сервисного ключа действуют только политики организации без проверки членства пользователя и команды.

4.3. Контроль потребления

Продукт учитывает потребление ресурсов моделей и применяет квоты без двойного списания.

Категории учёта токенов

Учёт ведётся по категориям:

- вход (prompt);
- выход (completion);
- чтение кэша;
- запись кэша;
- итог без двойного счёта.

Стоимость вычисляется по категориям и модели на основании цен модели. Цены для кэшированных токенов задаются отдельно. Продуктовый журнал запросов фиксирует категории и квоты тарифа; при отсутствии стоимости от шлюза в аналитике отображается «не получено».

Схема «резерв → вызов → фиксация»

Потребление учитывается по схеме:

1. Резерв квоты — перед вызовом модели продукт резервирует квоту организации; резерв записывается в ValKey.
2. Вызов шлюза — при успешном резерве выполняется вызов модели через внутренний заголовок.
3. Фиксация фактического потребления — после получения ответа от модели фиксируется фактическое потребление по категориям из ответа шлюза; резерв освобождается, в продуктовый журнал записывается реальное потребление.

Отказ резерва квоты происходит без вызова модели. События резерва и фиксации проходят через ValKey; фоновый обработчик проецирует их в PostgreSQL.

Лимиты потребления

Продукт проверяет лимиты клиента API и tenant до вызова модели. Поддерживаются окна с первого запроса:

- 5 часов;
- 1 день;
- 1 неделя.

Также проверяются лимиты:

- RPM (запросов в минуту);
- TPM (токенов в минуту);
- потолок одновременных запросов.

При превышении или исчерпании лимита возвращается код состояния HTTP 429 с описанием превышенного лимита.

Расширенный учёт

Учёт категорий кэшированных токенов и гибкие лимиты находятся в процессе развития. Их доступность в конкретной версии продукта определяется дорожной картой.

4.4. Безопасность и обращение с данными

Контроль доступа на уровне сети

По умолчанию запрещено всё, что не входит в белый список маршрутов шлюза. Вышестоящий адрес назначения модели или внешнего поставщика задаётся только конфигурацией на стороне сервера; клиентский запрос не может задать произвольный адрес.

Защита секретов

Секреты не попадают в спецификацию и статус пользовательских ресурсов Kubernetes и не записываются в журналы. Передача секретов выполняется через ссылки на секреты Kubernetes. На платформе применяется хеширование ключей (argon2id).

Журнал аудита

Продукт ведёт структурированный журнал аудита безопасности: кто, что, когда и результат операции. Журнал не содержит ключей, тел запросов и промптов. Сквозной идентификатор запроса обеспечивает корреляцию учёта использования, аудита и разбора инцидентов.

Защита содержимого (Guardrails)

Guardrails — отдельный включаемый слой проверки, маскирования и блокировки содержимого в шлюзе запросов к моделям. По умолчанию выключен (guardrails.enabled=false) и не изменяет трафик запросов к моделям, пока не включён в конфигурации поставки.

Политики защиты описываются пользовательским ресурсом GuardrailPolicy уровня кластера и привязываются к модели и/или клиенту API. Оператор платформы формирует эффективный набор политик и отражает его в статусе модели; шлюз читает только этот набор и не требует прямого доступа к ресурсам уровня кластера.

Фазы применения и действия

Политики применяются в одной из фаз:

- pre_call — до вызова модели (проверка входных данных);
- during_call — перед отправкой прокси-запроса;
- post_call — проверка ответа модели, в том числе в потоке SSE;
- logging_only — только аудит без блокировки.

Поддерживаемые действия: block (отклонение запроса), mask (маскирование фрагментов), intervene (изменение запроса или ответа), audit (фиксация события без изменения).

Встроенные типы детекторов

- pii — персональные данные (имена, адреса, номера документов);
- secret — ключи доступа, токены, учётные данные;
- denylist — запрещённые формулировки по списку заказчика;
- prompt_injection — эвристики подмены инструкций и попыток обхода ограничений модели (jailbreak);
- external — внешний сервис проверки по контракту Generic Guardrail API.

Адрес внешнего сервиса задаётся только конфигурацией платформы, не параметрами клиентского запроса, для защиты от подделки запросов на стороне сервера (SSRF).

Защита от внедрения промпта (prompt injection)

Внедрение промпта — класс атак, при котором во входных данных пользователя содержатся инструкции, направленные на переопределение системного промпта, обход ограничений модели («jailbreak»), раскрытие скрытых инструкций или перевод модели в привилегированный режим.

Встроенный детектор типа `prompt_injection` выполняет эвристический поиск сигнатур и шаблонов, включая:

- `ignore/disregard/forget previous instructions`;
- `override/reveal system prompt`;
- `developer/admin/system mode`;
- `jailbreak, DAN mode`.

Детектор поддерживает настраиваемый порог срабатывания и собственный набор сигнатур заказчика. Типовое действие — `block` (отклонение запроса до вызова модели) или `audit` (фиксация без блокировки). Маскирование текста для этого типа не применяется: цель — не пропустить вредоносную инструкцию в модель.

Детектор рекомендуется включать в фазе `pre_call` (и при необходимости `post_call` для ответов) в составе обязательной базовой политики безопасности организации.

Обязательные политики

Политики с признаком `defaultOn` обязательны: клиент не может отключить их параметрами запроса. Попытка снять обязательную политику отклоняется на уровне шлюза. Фактическая блокировка содержимого нормализуется как отказ политики безопасности, а не как ошибка модели.

Режим отказа и аудит

События срабатывания Guardrails фиксируются в аудите и метриках без записи тел промптов и ответов. Режим отказа детектора по умолчанию закрытый (`fail closed`): при недоступности детектора запрос блокируется. Открытый режим (`fail-open`) допускается только при явной конфигурации и обязан оставлять наблюдаемый след.

Приватность и изолированное размещение

Базовая модель продукта предполагает развёртывание в Kubernetes-кластере заказчика. При использовании только локальных моделей и без включения внешних поставщиков данные запросов не покидают периметр организации.

Условия хранения содержимого диалогов, обезличивания данных и внешних обращений определяются конфигурацией конкретной поставки. Режимы приватности (только контур / гибрид / обезличивание) находятся в процессе развития и не объявляются готовыми в базовой поставке.

4.5. Надёжность и диагностируемость

Идемпотентность операций

Согласование состояния оператором платформы выполняется идемпотентно: повторное применение спецификации пользовательского ресурса приводит к тому же фактическому состоянию без дублирования действий.

Схема «резерв → вызов → фиксация» предотвращает двойное списание токенов: резерв записывается перед вызовом, фиксация перезаписывает резерв фактическим потреблением, повторная фиксация с тем же идентификатором запроса игнорируется.

Компенсация частичных ошибок

При частичном создании ключей запросов или регистрации внешних моделей продукт выполняет компенсацию: откатывает созданные записи в продуктовой базе и удаляет пользовательские ресурсы платформы через HTTP-адаптер. Внешняя ошибка не приводит к тихому рассинхрону.

События и исходящие уведомления

После каждого запроса к модели формируется событие и публикуется в Kafka. Технические метрики (учёт, статус, задержка) собираются всегда; содержимое запроса и ответа — только по политике. Сбой Kafka не блокирует ответ клиенту; повторная доставка использует стабильный идентификатор события для идемпотентной обработки получателем.

Шлюз отправляет исходящие уведомления продукта о приближении к лимиту, исчерпании, восстановлении квоты и изменении состояния модели. Адрес получателя, типы событий и включение уведомлений задаются в пользовательских ресурсах. Повторная доставка использует стабильный идентификатор события; сбой получателя не прерывает запрос к модели.

Проверки живости и готовности

Серверную часть продукта можно развернуть в одном из режимов:

- только управление (консоль, организации, развёртывания моделей, политики);
- только исполнение запросов к моделям;
- оба режима одновременно.

Проверка `/health/live` отвечает, что процесс запущен, и не зависит от внешних сервисов.

Проверка `/health/ready` проверяет доступность зависимостей, необходимых для текущего режима:

- для учёта и кэша — база данных PostgreSQL и ValKey;
- для исполнения запросов — шлюз запросов к моделям;
- для управления — HTTP-адаптер развёртываний моделей.

Если зависимость недоступна, экземпляр не считается готовым к приёму нагрузки. Kubernetes не направляет на него трафик.

Корреляция событий

Каждый запрос к модели получает сквозной идентификатор, который фиксируется в продуктовом журнале запросов, журнале аудита, событиях Kafka и метриках шлюза. Идентификатор обеспечивает корреляцию событий при разборе инцидентов и анализе производительности.

4.6. Условия включения дополнительных возможностей

Базовая поставка продукта (оператор, шлюз запросов и пользовательские ресурсы для жизненного цикла моделей) работоспособна без необязательных подсистем. Следующие возможности включаются отдельно и не входят в обязательный состав:

Хранение диалогов

Платформенный контур поддерживает хранение содержимого диалогов в PostgreSQL или S3-совместимом хранилище. Включение хранения задаётся политикой в пользовательских ресурсах. При отсутствии политики хранения содержимое запросов и ответов не сохраняется; продуктовый журнал фиксирует только метаданные запроса и категории использования токенов.

Автомасштабирование

Переход реплик модели от 0 к 1 выполняется оператором платформы при поступлении первого запроса. Масштабирование от 1 к N выполняется KEDA на основе метрик Prometheus или VictoriaMetrics. Без установленного KEDA и стека наблюдаемости модели работают с фиксированным числом реплик.

Средства наблюдаемости

Оператор и шлюз экспортируют метрики на отдельных точках. Стек наблюдаемости (Prometheus или VictoriaMetrics, Grafana, Loki, OpenTelemetry, DCGM для GPU) не обязателен для работы платформы; при его отсутствии автомасштабирование и панели мониторинга недоступны.

Termidesk Connect

Termidesk Connect — опциональный компонент поставки продукта, а не обязательный элемент базовой конфигурации. При включении этой опции Termidesk Connect используется как точка безопасного подключения внешних клиентов Astra Code к APISIX и как возможный канал к облачным и иным внешним контурам моделей.

Без Termidesk внешний доступ организуется иными согласованными средствами периметра; внутренние клиенты Astra Code по-прежнему подключаются к APISIX по VPN с JWT, а клиенты вне Astra Code — к шлюзу запросов по VPN с ключом.

5. Типовые сценарии

В этой главе описаны ключевые сценарии использования АСТРА ИИ [ХАБ], которые связывают функции платформы с ролями пользователей, предпосылками и ожидаемыми результатами. Сценарии сформулированы на уровне задач и бизнес-логики; они не являются пошаговыми инструкциями пользовательского интерфейса.

5.1. Предоставление корпоративного доступа

Роль: администратор организации.

Назначение: управление участниками, командами и правами доступа к моделям в пределах организации.

Предпосылки

- Организация зарегистрирована в платформе и связана с поставщиком идентификации (DEX / Keycloak).
- Оформлена активная подписка, определяющая каталог разрешённых моделей, квоты и лимиты.
- Администратор организации обладает правами на управление пользователями и ключами.

Описание сценария

Администратор назначает роли участникам организации, управляет составом команд и выдаёт персональные или сервисные ключи запросов к моделям. Платформа проверяет членство пользователя в организации, наличие активной подписки и соответствие лимитам тарифа (число пользователей, число сервисных ключей). При успешном создании пользовательского ключа продукт генерирует секрет формата sk-<8hex>-<32hex>, который передаётся в платформенный контур для регистрации клиента API с именем вида hub-user-<uuid> и привязкой к tenant организации hub-org-<uuid>. Полный секрет отображается однократно; при смене ключа старый секрет становится недействительным.

Администратор также может создать сервисный ключ организации для машинного доступа без привязки к конкретному пользователю. Число таких ключей ограничено тарифом. На запросы от имени сервисного ключа действуют только политики организации, без применения командных или пользовательских ограничений.

Результат

- Участники организации получают права в соответствии с назначенными ролями.
- Каждый активный пользователь имеет персональный ключ для запросов к моделям; администратор может отозвать или сменить ключ.
- Сервисные ключи организации позволяют приложениям работать от имени организации без пользовательского контекста.
- Каталог доступных моделей определяется активной подпиской; без подписки запросы к моделям недоступны.

5.2. Подключение приложения сервисным ключом

Роль: разработчик приложения, системный интегратор.

Назначение: машинный доступ к моделям от имени организации через OpenAI-совместимый интерфейс.

Предпосылки

- Организация имеет активную подписку.
- Администратор организации создал сервисный ключ и передал секрет приложению защищённым способом.

Описание сценария

Приложение направляет запрос к продуктовой точке входа `/api/v1/inference/{path}`, передавая сервисный ключ в заголовке `X-API-Key`. Платформа определяет ключ, проверяет его действительность (не отозван, не истёк срок), извлекает привязку к организации и `tenant`. До обращения к модели выполняются проверки:

- членство и активность организации;
- наличие активной подписки;
- доступность запрашиваемой модели в каталоге организации;
- соответствие расписанию доступности (если задано);
- наличие доступной квоты (окна 5 часов, 1 день, 1 неделя; RPM, TPM, параллельные запросы).

Если хотя бы одна проверка не пройдена, платформа возвращает ранний отказ (HTTP 403 или 429) без вызова модели. При успехе запрос передаётся через внутренний заголовок в OpenAI-совместимый шлюз, который маршрутизирует его в локальную или внешнюю модель. После получения ответа платформа фиксирует фактическое потребление токенов (вход, выход, чтение кэша, запись кэша) в продуктивном журнале и списывает зарезервированную квоту.

Результат успешного запроса

- Приложение получает ответ модели в формате OpenAI API.
- Потребление токенов учитывается в аналитике организации.
- Квота уменьшается на фактический объём без двойного списания.

Результат отклонённого запроса

- Приложение получает HTTP 403 (нарушение политики) или 429 (исчерпание квоты) с описанием причины отказа.
- Модель не вызывается; токены не списываются; inference-ресурсы не потребляются.

5.3. Размещение локальной модели

Роль: администратор платформы.

Назначение: создание развёртывания локальной модели с выделением вычислительных ресурсов и подключением к каталогу.

Предпосылки

- Администратор платформы имеет права на создание конфигураций моделей и развёртываний.
- Определены параметры модели: источник весов (HuggingFace, S3-совместимое хранилище), движок исполнения (vLLM, SGLang, Ollama, FasterWhisper), требуемое количество GPU, параметры запуска.
- Доступен HTTP-адаптер для взаимодействия с пользовательскими ресурсами Kubernetes.

Описание сценария

Администратор создаёт конфигурацию развёртывания через продуктовый API, указывая имя модели, репозиторий весов, движок, профиль GPU и параметры. Платформа проверяет корректность спецификации до обращения к кластеру. При необходимости администратор запрашивает предварительную оценку размещения (fit check), которая показывает доступность GPU на узлах кластера, свободную память и осуществимость развёртывания.

После подтверждения администратор создаёт развёртывание с именем вида `hubd-
<name>-<org-uuid>`. Продуктовый контур регистрирует ресурс модели через HTTP-адаптер, передавая спецификацию в платформенный контур. Оператор платформы согласует состояние: инициирует загрузку весов в кэш на постоянном томе, создаёт поды исполнения модели, выделяет GPU и публикует статус ресурса. Консоль получает статус, логи, события и метрики развёртывания через продуктовый API; прямого обращения к API кластера не требуется.

При изменении конфигурации (например, обновлении образа или параметров) оператор выполняет безопасный для GPU rollout: сначала создаётся новая реплика и доводится до состояния Ready, затем удаляется старая, что снижает риск простоя.

Ответственность платформенного контура

- Загрузка весов из указанного источника (с учётными данными из секрета Kubernetes).
- Согласование пода исполнения, выделение GPU, подключение кэша весов.
- Автоматическое масштабирование: переход $0 \leftrightarrow 1$ управляется оператором, масштабирование $1 \leftrightarrow N$ — через KEDA на основе метрик.
- Обновление статуса ресурса модели (готовность, ошибки, размещение, перечень GPU).

Результат

- Локальная модель зарегистрирована в каталоге платформы и доступна организациям в соответствии с подписками.
- Оператор поддерживает жизненный цикл модели: развёртывание, масштабирование, обновление, удаление.
- Статус, логи и метрики доступны администратору через консоль.
- Модель доступна для запросов через OpenAI-совместимый шлюз.

5.4. Подключение внешней модели

Роль: администратор платформы.

Назначение: регистрация внешнего OpenAI-совместимого источника в каталоге платформы без локального развёртывания.

Предпосылки

- Администратор платформы имеет права на создание внешних моделей.
- Известен адрес внешнего API, совместимого с OpenAI, и учётные данные доступа.

- Определены условия передачи данных и требования безопасности для обращения к внешнему источнику.

Описание сценария

Администратор регистрирует внешнюю модель через продуктовый API, указывая имя, адрес upstream-сервиса и ссылку на секрет Kubernetes с учётными данными. Платформа создаёт ресурс модели в режиме external без локальных подов обслуживания. Шлюз проверяет доступность внешнего источника и отражает состояние в статусе ресурса. Адрес вышестоящего сервиса задаётся только на стороне платформы; клиентский запрос не может изменить upstream (защита от SSRF).

Внешняя модель включается в каталог организации в соответствии с подпиской. Запросы к ней проходят те же проверки политик, расписаний и квот, что и локальные модели. Шлюз маршрутизирует запрос к внешнему API, получает ответ и возвращает его клиенту. Учёт токенов ведётся по данным внешнего источника; если стоимость не получена, в аналитике отображается «не получено».

Результат

- Внешняя модель доступна в каталоге организаций наравне с локальными.
- Клиентский интерфейс остаётся единым; приложения не различают локальные и внешние источники.
- Условия передачи данных к внешнему источнику определяются политикой организации и контрактом поставки.
- Статус доступности внешней модели отражается в консоли.

5.5. Управление потреблением и анализ использования

Роль: администратор организации, администратор платформы, пользователь.

Назначение: настройка политик потребления, просмотр аналитики использования и диагностика ограничений доступа.

Предпосылки

- Организация имеет активную подписку с определёнными квотами и лимитами.
- Ведётся продуктовый журнал запросов с учётом токенов по категориям (вход, выход, чтение кэша, запись кэша).

Описание сценария

Администратор организации наСТРАивает допустимые политики потребления: квоты на окна использования (5 часов, 1 день, 1 неделя), ограничения RPM и TPM, потолок параллельных запросов. Политики применяются на уровне организации, команды или пользователя. Платформа проверяет лимиты до вызова модели; при исчерпании квоты возвращается HTTP 429 с описанием превышенного ограничения.

Пользователь видит свои запросы: модель, время, токены, стоимость, статус. Администратор организации просматривает использование всей организации с разрезами по пользователям, моделям, периодам. Администратор платформы имеет доступ к аналитике любой организации. Доступны сводка, детализация, фильтры и выгрузка в CSV.

При отказе в доступе пользователь получает ответ с указанием причины: отсутствие подписки, модель не в каталоге, исчерпание квоты, нарушение расписания. События приближения к лимиту, исчерпания и восстановления могут направляться на настроенные адреса исходящих уведомлений.

Результат

- Администратор контролирует интенсивность и объём использования моделей.
- Пользователи получают прозрачную информацию о своих запросах и потреблении.
- Аналитика позволяет выявлять аномалии, оптимизировать квоты и планировать ресурсы.
- Причины ограничения доступа фиксируются в журнале и доступны для анализа.

5.6. Применение защиты содержимого

Роль: администратор платформы, администратор организации.

Назначение: включение и настройка политик Guardrails для проверки, маскирования и блокировки содержимого запросов и ответов моделей.

Предпосылки

- Функция Guardrails включена в конфигурации поставки платформы (`guardrails.enabled=true`).
- Определены политики GuardrailPolicy на уровне кластера, привязанные к моделям и/или клиентам.

Описание сценария

Администратор платформы создаёт политики защиты содержимого, указывая типы детекторов (PII, секреты, denylist, prompt injection, внешний сервис), фазы применения (`pre_call`, `during_call`, `post_call`, `logging_only`) и действия (`block`, `mask`, `intervene`, `audit`). Политики с признаком `defaultOn` являются обязательными; клиент не может отключить их параметрами запроса.

При обработке запроса шлюз применяет эффективный набор политик, сформированный оператором платформы и отражённый в статусе модели. Детектор `prompt_injection` выполняет эвристический поиск сигнатур внедрения инструкций (`ignore previous instructions`, `override system prompt`, `jailbreak`, `DAN mode` и др.) с НАСТРАИваемым порогом срабатывания. При обнаружении нарушения в фазе `pre_call` запрос блокируется до вызова модели; клиент получает отказ с указанием причины безопасности.

Детекторы PII и секретов маскируют персональные данные и токены в содержимом. Детектор `denylist` проверяет наличие запрещённых формулировок. Внешний детектор обращается к заданному сервису проверки по контракту Generic Guardrail API; адрес задаётся только конфигурацией платформы (защита от SSRF).

События срабатывания Guardrails фиксируются в журнале аудита и метриках без записи тел промптов и ответов. Режим отказа детектора по умолчанию закрытый (`fail closed`); открытый `fail-open` допускается только при явной конфигурации и обязан оставлять наблюдаемый след.

Результат проверки

- Запросы и ответы, нарушающие политики безопасности, блокируются или маскируются до передачи клиенту.
- Обязательные политики нельзя снять клиентским запросом; это предотвращает обход защиты.
- Аудит содержит записи срабатывания без секретов и содержимого промптов.
- Администратор получает метрики блокировок и маскирований для анализа угроз.

Условия применения

Сценарий актуален только при включённой функции Guardrails в конфигурации поставки. Если функция выключена (`guardrails.enabled=false`), содержимое не проверяется и не изменяется; защита содержимого обеспечивается организационными мерами: разграничением ключей, белым списком маршрутов, серверным upstream и аудитом.

5.7. Дополнительные сервисы: MCP Hub, RAG и Skills

Роль: пользователи и администраторы организаций, разработчики приложений.

Назначение: использование единого MCP Hub, RAG-сервиса и маркетплейса Skills в составе продуктового контура.

Статус описания

Версия 3 технического описания включает MCP Hub, RAG-сервис и маркетплейс Skills в продуктовый контур АСТРА ИИ [ХАБ]. При этом документ не определяет их полный набор пользовательских функций, API, правила публикации контента, эксплуатационные параметры и типовые последовательности действий.

Поэтому в данном разделе не приводятся конкретные сценарии использования этих сервисов. Детальные пользовательские контракты, интеграционные интерфейсы и сценарии работы определяются отдельными документами поставки и могут быть добавлены в техническое описание после их согласования.

Общие принципы интеграции

- Доступ к дополнительным сервисам регулируется теми же механизмами идентификации, ролей и подписок, что и доступ к моделям.
- Единая точка входа через APISIX с доверенной идентификацией и заменой клиентских заголовков.
- Учёт использования и аудит действий фиксируются в продуктивном журнале.
- Политики безопасности и квоты потребления применяются в соответствии с настройками организации.

После получения детальных контрактов данный раздел может быть дополнен типовыми сценариями публикации Skills, интеграции с MCP Hub и использования RAG-сервиса для контекстного поиска.

6. Функционал программных модулей

Настоящий раздел распределяет описанные функции АСТРА ИИ [ХАБ] по подсистемам и зонам ответственности. Платформа построена как двухконтурный продукт: продуктовый контур решает задачи управления доступом, политиками и учётом; платформенный контур исполнения моделей обеспечивает инфраструктуру, жизненный цикл моделей и вычислительные ресурсы. Совместные компоненты разделены по ответственности за функции.

6.1. Консоль управления

Консоль управления — веб-интерфейс администрирования платформы, реализованный как одностраничное приложение на React. Консоль работает исключительно через продуктовый программный интерфейс и не обращается к программному интерфейсу кластера Kubernetes напрямую.

Управление организациями и доступом

Консоль предоставляет администраторам организации управление пользователями, командами, ролями и правами. Состав организации синхронизируется с корпоративным поставщиком личности через систему идентификации DEX (Keycloak). Администратор может создавать команды внутри организации, назначать пользователей в команды и управлять ролями. Встроенная роль администратора организации предоставляет полный доступ в контуре организации.

Консоль позволяет генерировать персональные ключи запросов формата `sk-<8hex>-<32hex>` для активных пользователей. Полный секрет ключа показывается только один раз при создании. Смена ключа делает старый недействительным. Для машинного доступа администратор может создавать сервисные ключи без привязки к конкретному пользователю. Число сервисных ключей ограничено тарифом организации.

Тарифы и подписки

Платформенный администратор управляет тарифными планами через консоль. Тарифный план определяет:

- число пользователей и сервисных ключей;
- список разрешённых моделей (allowlist);
- квоты потребления по моделям и категориям токенов (входные, выходные, кэш);
- цену за период в условных единицах.

Администратор организации оформляет подписку на выбранный тарифный план. Подписка активирует доступ организации к запросам моделей и каталогу моделей. Без активной подписки запросы к моделям недоступны, но права управления организацией сохраняются.

Каталог конфигураций и развёртывания

Платформенный администратор управляет каталогом конфигураций моделей через консоль без прямого обращения к Kubernetes. Каталог содержит описания доступных моделей с указанием источника весов (HuggingFace, зеркало, хранилище S3), профиля исполнения, класса исполнения, требований к ресурсам и движка обслуживания (vLLM, SGLang, Ollama, FasterWhisper).

Для внешних моделей, совместимых с OpenAI, каталог содержит режим внешнего поставщика с параметрами подключения и списком доступных моделей удалённого сервиса. Такие модели не требуют локальных подов исполнения и графических процессоров.

Консоль позволяет создавать и изменять развёртывания моделей. Администратор выбирает конфигурацию из каталога, задаёт число реплик и параметры размещения. Серверная часть проверяет права платформенного администратора, валидирует спецификацию и передаёт запрос в HTTP-адаптер. HTTP-адаптер создаёт или изменяет пользовательский ресурс модели в Kubernetes. Некорректная спецификация отсекается до обращения к кластеру.

Консоль показывает статус развёртываний в реальном времени. Подключение к журналу подов исполнения выполняется через WebSocket с подпротоколом `bearer,<token>`. Администратор видит события жизненного цикла модели, загрузку весов, готовность реплик и ошибки конфигурации.

Политики, квоты и расписания

Консоль предоставляет администратору организации управление политиками доступа к моделям. Политики задаются на уровне организации и команды и определяют:

- список разрешённых моделей для организации (сужение каталога тарифа);
- расписания доступности по времени и дням недели;
- быстрые лимиты запросов в минуту для защиты от перегрузки;
- квоты потребления токенов по моделям и категориям.

Квоты задаются в токенах отдельно для входных, выходных и кэшированных токенов. Политика применяется до вызова модели: серверная часть проверяет членство пользователя в организации, активность подписки, список разрешённых моделей, расписание доступа и квоту. Решение об отказе принимается до потребления вычислительных ресурсов моделей. После успешного выполнения запроса серверная часть фиксирует фактическое потребление в продуктивном журнале.

Консоль отображает текущее состояние квот организации и команд с разбивкой по моделям. Администратор видит использованные и оставшиеся квоты в периоде, темп потребления и прогноз исчерпания.

Аналитика и диагностика

Консоль предоставляет аналитику использования моделей с разрезами по организациям, командам, пользователям, моделям и периодам. Источником данных служит продуктивный журнал запросов в PostgreSQL. Аналитика включает:

- объём потреблённых токенов по категориям (входные, выходные, кэш);
- число запросов и распределение по моделям;
- задержки выполнения запросов (среднее, медиана, перцентили);
- ошибки доступа и отказы по политикам.

Консоль позволяет фильтровать журнал запросов по идентификатору запроса, пользователю, модели, статусу ответа и временному интервалу. Детальная запись запроса содержит метаданные (личность, модель, время, задержка, токены) без содержимого промта и ответа, если иное не задано политикой записи.

Диагностика развёртываний показывает метрики здоровья моделей из платформенного контура: готовность реплик, нагрузку на графические процессоры, память, задержки обработки, длину очереди запросов. Эти метрики собираются средствами наблюдаемости платформенного контура (Prometheus, VictoriaMetrics) и доступны через продуктовый программный интерфейс.

6.2. Идентификация и входной шлюз

Входной шлюз APISIX является единой точкой входа для всех запросов к платформе и выполняет аутентификацию пользователей. Система идентификации DEX (Keycloak) обеспечивает интеграцию с корпоративным поставщиком личности заказчика по протоколу OIDC.

Аутентификация пользователей

Для интерактивных пользователей APISIX выполняет аутентификацию через Authorization Code Flow с PKCE. Пользователь перенаправляется на страницу входа DEX, проходит аутентификацию через корпоративного поставщика личности и получает токен идентификации. APISIX проверяет подпись токена, извлекает личность пользователя и устанавливает доверенные заголовки для передачи в серверную часть продукта.

Для машинного доступа приложения используют сервисные ключи организации. APISIX проверяет ключ запросов из заголовка `Authorization: Bearer sk-...` по кэшу в ValKey с запасным чтением из PostgreSQL при недоступности кэша. После проверки APISIX устанавливает заголовки личности сервисного аккаунта.

Клиентские заголовки личности и ключей запросов заменяются доверенными на стороне шлюза. Это исключает подмену личности клиентом: серверная часть доверяет только заголовкам, установленным APISIX, и не читает заголовки, пришедшие от клиента напрямую.

После изменения ключей доступа в серверной части продукта кэш определителя ключей в ValKey обновляется автоматически. APISIX читает обновлённый кэш при следующем запросе. Задержка актуализации кэша не превышает нескольких секунд.

Разрешение доступа к маршрутам

APISIX публикует только разрешённые маршруты продуктового контура. Служебные маршруты внутренних компонентов, программный интерфейс кластера Kubernetes и административные точки входа не публикуются наружу.

Разрешённые маршруты включают:

- консоль управления (статические ресурсы и SPA);
- продуктовый программный интерфейс /api/*;
- продуктовую точку входа запросов к моделям /api/v1/inference/*;
- WebSocket подключения для журналов развёртываний.

Авторизацию операций управления (права на изменение организации, создание развёртываний, настройку политик) выполняет серверная часть продукта, а не входной шлюз. APISIX только устанавливает доверенную личность.

6.3. Серверная часть продукта

Серверная часть продукта — ядро продуктового контура, реализующее организацию, пользователей, команды, роли, тарифы, подписки, ключи запросов, политики доступа, расписания, квоты, продуктовый журнал запросов, аналитику и учёт потребления. Серверная часть может работать в одном из режимов: только управление (консоль, организации, развёртывания моделей, политики), только исполнение запросов к моделям или оба режима одновременно.

Организации, команды, пользователи и права

Организация является основной единицей продуктовой изоляции. Инфраструктурно организации соответствует арендатор платформы (Tenant). Организация объединяет пользователей, роли, подписку и лимиты в изолированный контур.

Команды группируют пользователей внутри организации и задают область применения отдельных политик. Пользователь может состоять в нескольких командах. Политики команды уточняют политики организации: сужают список разрешённых моделей, задают квоты команды внутри квоты организации.

Роли организации задаются набором разрешений из фиксированного каталога. Встроенная защищённая роль администратора организации предоставляет полный доступ в контуре организации. Права платформенного администратора (управление развёртываниями, конфигурациями моделей, тарифами) отделены от ролей организации и доступны только администратору платформы без обязательного членства в организации.

Серверная часть проверяет права на операции управления по ролям и разрешениям пользователя. Состав организации синхронизируется с поставщиком личности DEX. При изменении состава организации в поставщике личности синхронизация обновляет членство в продуктивном контуре.

Инфраструктурные сущности — арендатор платформы (Tenant) и клиент программного интерфейса (APIClient) — создаются автоматически при оформлении подписки организации. Арендатор платформы является единицей изоляции на уровне кластера. Клиент программного интерфейса получает права на вызов моделей в пространстве имён арендатора.

Тарифы, подписки, ключи

Тарифный план определяет параметры доступа организации к каталогу моделей и ресурсам платформы. Тариф задаёт:

- число пользователей и сервисных ключей;
- список разрешённых моделей (allowlist);
- квоты потребления по моделям и категориям токенов;
- цену за период в условных единицах.

Подписка связывает организацию с тарифным планом. Без активной подписки запросы к моделям и каталог моделей организации недоступны. Права управления организацией при отсутствии подписки сохраняются. Оформление подписки регистрирует арендатора платформы и обеспечивает персональные ключи запросов для активных членов организации.

Персональные ключи запросов генерируются серверной частью продукта в формате `sk-<8hex>-<32hex>`. Полный секрет показывается один раз при создании. В базе данных хранится хеш секрета. Смена ключа делает старый недействительным. Отзыв ключа отключает доступ клиента к моделям. Секрет не попадает в журналы и ответы программного интерфейса.

Сервисные ключи предоставляют машинный доступ от имени организации без привязки к конкретному пользователю. Число сервисных ключей ограничено тарифом организации. На запросы с сервисным ключом действуют только политики организации без проверки членства в команде.

Политики, квоты и расписания

Политики доступа к моделям задаются на уровне организации и команды. Серверная часть проверяет политики до вызова модели через продуктовую точку входа `/api/v1/inference/{path}`:

1. Членство пользователя в организации.
2. Активность подписки организации.
3. Наличие модели в списке разрешённых (allowlist тарифа и организации).
4. Расписание доступности организации и команды.
5. Быстрые лимиты запросов в минуту (rate limit).
6. Квоты потребления токенов.

Расписания доступности задаются по времени и дням недели на уровне организации и команды. Расписание организации ограничивает доступ ко всем моделям в контуре организации. Расписание команды уточняет расписание организации.

Следует различать расписание доступности организации и расписание модели. Расписание организации определяет, когда пользователи организации могут обращаться к моделям. Расписание модели определяет, когда платформа поддерживает реплики модели в работающем состоянии (управление автомасштабированием). Эти расписания независимы и проверяются на разных уровнях: организационное — серверной частью продукта, модели — оператором платформенного контура.

Квоты задаются в токенах отдельно для входных, выходных и кэшированных токенов по моделям. При поступлении запроса серверная часть резервирует оценочную квоту в ValKey. После выполнения запроса фактическое потребление

фиксируется в продуктовом журнале и вычитается из квоты организации. Если запрос не завершился (ошибка, отмена клиентом), резерв освобождается.

Решение об отказе по политикам принимается до потребления вычислительных ресурсов моделей. Это защищает платформу от неавторизованного использования и перегрузки.

Каталог и развёртывания

Каталог конфигураций моделей хранится в PostgreSQL и управляется через продуктовый программный интерфейс. Конфигурация модели описывает:

- идентификатор модели и отображаемое имя;
- источник весов (HuggingFace, зеркало, хранилище S3);
- профиль исполнения (параметры движка vLLM, SGLang, Ollama);
- класс исполнения (требования к ресурсам, граничные условия);
- движок обслуживания (vLLM, SGLang, Ollama, FasterWhisper);
- режим внешнего поставщика для OpenAI-совместимых внешних моделей.

Серверная часть проверяет спецификацию конфигурации перед созданием развёртывания. Проверка включает корректность параметров движка, доступность источника весов, соответствие требований к ресурсам классу исполнения. Некорректная спецификация отсекается до обращения к кластеру.

Для создания или изменения развёртывания серверная часть формирует запрос к HTTP-адаптеру. HTTP-адаптер преобразует REST-запрос в создание или изменение пользовательского ресурса модели в Kubernetes. Имя ресурса формируется по шаблону `hubd-<name>-<org uuid>`.

Оператор платформенного контура согласует фактическое состояние ресурса с желаемым. Серверная часть опрашивает статус развёртывания через HTTP-адаптер и предоставляет его консоли в реальном времени. Администратор видит события жизненного цикла модели, загрузку весов, готовность реплик и ошибки конфигурации через WebSocket подключение к журналу подов.

Внешние модели в режиме внешнего поставщика не требуют развёртывания в кластере. Серверная часть регистрирует внешнюю модель в каталоге с параметрами подключения (базовый URL, ключ доступа, список доступных моделей) и прокси-режимом в шлюзе запросов.

Продуктовая точка входа запросов

Продуктовая точка входа `/api/v1/inference/{path}` является единым маршрутом для запросов к моделям через продуктовый контур. После установки доверенной личности входным шлюзом APISIX серверная часть выполняет проверки политик:

1. Извлечение личности из доверенных заголовков.
2. Проверка членства пользователя в организации.
3. Проверка активности подписки организации.
4. Извлечение модели из тела запроса (поле `model`).
5. Проверка наличия модели в `allowlist` тарифа и организации.
6. Проверка расписания доступности организации и команды.
7. Проверка быстрых лимитов запросов (`rate limit`).

8. Резервирование оценочной квоты в ValKey.

При успешных проверках запрос маршрутизируется в шлюз запросов к моделям с установкой внутреннего служебного заголовка. Шлюз запросов не публикуется наружу и доступен только через продуктовую точку входа. Прямое обращение к шлюзу без прохождения проверок продуктового контура невозможно.

После получения ответа от модели серверная часть фиксирует фактическое потребление в продуктивном журнале запросов и вычитает токены из квоты организации. Событие запроса публикуется в Kafka для аналитики и исходящих уведомлений. Сбой Kafka не блокирует ответ клиенту.

Аналитика и аудит

Продуктовый журнал запросов ведётся в PostgreSQL в отдельной схеме продуктового контура. Каждая запись содержит:

- идентификатор запроса (UUID);
- личность пользователя (пользователь или сервисный аккаунт);
- организация, команда, роль;
- модель и маршрут программного интерфейса;
- время начала и завершения запроса;
- задержка выполнения (миллисекунды);
- токены по категориям (входные, выходные, кэш);
- статус ответа (успех, ошибка, отказ по политике);
- причина отказа (если применимо).

Содержимое промта и ответа записывается только по политике записи. По умолчанию продуктовый журнал содержит только метаданные. Это защищает чувствительные данные и снижает объём хранилища.

Серверная часть предоставляет программный интерфейс аналитики с разрезами по организациям, командам, пользователям, моделям и периодам. Консоль строит визуализации на основе данных продуктового журнала.

Аудит изменений конфигурации ведётся в отдельных таблицах продуктового контура. Каждое изменение организации, тарифа, подписки, политики, развёртывания фиксируется с личностью инициатора, временем и содержанием изменения.

6.4. Шлюз запросов к моделям

Шлюз запросов к моделям — компонент без хранения состояния, маршрутизирующий OpenAI-совместимые запросы к подам исполнения моделей в кластере Kubernetes. Шлюз является совместным компонентом: инфраструктурная поставка и согласование состояния — задача платформенного контура; функции категорийного учёта токенов, быстрых лимитов, расписаний доступности моделей, включаемых политик защиты содержимого и исходящих уведомлений реализуются на стороне продуктового контура.

Разрешённые маршруты и маршрутизация

Шлюз поддерживает следующие маршруты OpenAI-совместимого программного интерфейса:

- /v1/chat/completions — генерация ответов языковых моделей в диалоговом формате;
- /v1/completions — генерация продолжения текста;
- /v1/embeddings — получение векторных представлений текста;
- /v1/audio/* — распознавание и синтез речи;
- /v1/rerank — переранжирование документов по релевантности;
- /v1/models — список доступных моделей.

Поддерживаются потоковые ответы (Server-Sent Events), структурированный вывод (JSON Schema) и вызовы инструментов (tools/function calling).

Маршрутизация запросов выполняется по полю model в теле запроса. Шлюз определяет целевой сервис модели в кластере по имени модели и пространству имён арендатора. Если модель не готова (нет реплик в состоянии Ready), шлюз передаёт сигнал оператору для масштабирования от нуля к одной реплике и ожидает готовности.

Для внешних моделей в режиме внешнего поставщика шлюз проксирует запрос к удалённому сервису без локальных подов исполнения. Учётные данные внешнего поставщика хранятся в секретах Kubernetes и передаются шлюзом в заголовках запроса к удалённому сервису.

Продуктовые функции шлюза

Следующие функции шлюза реализуются на стороне продуктового контура и вызываются через внутренний контракт интеграции:

Категорийный учёт токенов. Шлюз учитывает токены отдельно для входных, выходных и кэшированных токенов. Учёт выполняется по данным, возвращённым моделью в поле usage ответа. Если модель не вернула учёт, шлюз оценивает токены по длине текста.

Быстрые лимиты. Защита от перегрузки на уровне шлюза проверяет число запросов в минуту по идентификатору клиента (пользователь или сервисный аккаунт). Счётчики лимитов хранятся в ValKey с временем жизни одна минута. При превышении лимита запрос отклоняется до вызова модели.

Расписания доступности моделей. Расписание модели определяет, когда платформа поддерживает реплики модели в работающем состоянии. Вне расписания оператор масштабирует модель до нуля реплик. Запросы вне расписания отклоняются шлюзом с указанием времени следующей доступности.

Следует различать расписание доступности модели и расписание доступности организации. Расписание модели управляет автомасштабированием на уровне платформенного контура. Расписание организации управляет правами доступа на уровне продуктового контура. Проверка расписания организации выполняется серверной частью продукта до передачи запроса в шлюз.

Защита содержимого

Политики защиты содержимого (Guardrails) — включаемая функция, проверяющая запросы пользователей и ответы моделей на наличие чувствительной информации и нарушений политики. Политики включают:

- обнаружение персональных данных (PII);
- обнаружение секретов (ключи доступа, токены, пароли);
- список запрещённых шаблонов (denylist);
- обнаружение атак внедрения промтов (prompt injection).

Политика защиты содержимого задаётся как пользовательский ресурс в Kubernetes и привязывается к модели. Шлюз применяет эффективные политики при обработке запроса и ответа. При обнаружении нарушения запрос отклоняется или ответ фильтруется в зависимости от настройки политики.

Решение о включении функций защиты содержимого и границы ответственности за их реализацию требуют согласования между продуктовым и платформенным контурами. В текущей поставке функции защиты содержимого находятся в стадии разработки.

События и уведомления

После каждого запроса к модели шлюз формирует событие и публикует его в Kafka. Событие содержит:

- идентификатор запроса (UUID);
- метаданные личности и организации;
- модель и маршрут;
- время и задержка выполнения;
- токены по категориям;
- статус ответа.

Содержимое запроса и ответа записывается в событие только по политике. Технические метрики (учёт, статус, задержка) собираются всегда.

Сбой Kafka не блокирует ответ клиенту. Событие публикуется асинхронно после возврата ответа. Повторная доставка использует стабильный идентификатор события для идемпотентной обработки получателем.

Исходящие уведомления (webhooks) отправляются фоновыми обработчиками продуктового контура при чтении потока событий из Kafka. Администратор организации может зарегистрировать webhook для получения уведомлений о запросах к моделям. Контракт уведомлений и проверка готовности функции определяются отдельно по дорожной карте.

6.5. HTTP-адаптер и платформенные компоненты

HTTP-адаптер — мост между REST-интерфейсом продуктового контура и пользовательскими ресурсами Kubernetes платформенного контура. Адаптер обеспечивает декларативное управление развёртываниями моделей, арендаторами, клиентами программного интерфейса и вычислительными ресурсами без прямого обращения консоли к программному интерфейсу кластера.

HTTP-адаптер

HTTP-адаптер вызывается только серверной частью продукта. Консоль и пользователи не имеют прямого доступа к адаптеру. Адаптер предоставляет REST-интерфейс для управления пользовательскими ресурсами:

- конфигурации моделей (ModelConfig);
- профили исполнения (RuntimeProfile);
- классы исполнения (RuntimeClass);
- кэш весов (WeightsCache);
- арендаторы платформы (Tenant);
- клиенты программного интерфейса (APIClient);
- цены моделей (ModelPrice);
- политики защиты содержимого (ContentGuardrail).

При получении REST-запроса адаптер преобразует его в создание, изменение или удаление соответствующего пользовательского ресурса в Kubernetes. Оператор платформенного контура следит за изменениями ресурсов и согласует фактическое состояние с желаемым.

HTTP-адаптер также предоставляет чтение статусов ресурсов и журналов подов исполнения. Серверная часть опрашивает адаптер для получения актуального состояния развёртываний и передачи его консоли в реальном времени через WebSocket.

Оператор Kubernetes

Оператор платформенного контура реализует согласование состояния пользовательских ресурсов моделей. Оператор отслеживает изменения ресурсов ModelConfig, RuntimeProfile, RuntimeClass, WeightsCache и управляет жизненным циклом подов исполнения моделей.

При создании или изменении конфигурации модели оператор:

1. Проверяет готовность кэша весов. Если веса не загружены, запускает загрузчик весов.
2. Создаёт тома данных для кэша весов и разделяемой памяти.
3. Формирует шаблон пода исполнения на основе профиля исполнения и класса исполнения.
4. Создаёт поды исполнения с требованиями к ресурсам (CPU, память, графические процессоры).
5. Ожидает готовности подов и обновляет статус ресурса модели.

При изменении шаблона пода (обновление профиля или класса исполнения) оператор выполняет безопасное для графических процессоров обновление: сначала создаёт и доводит до готовности новую реплику, затем удаляет старую. Это предотвращает потерю доступности модели и зависание графических процессоров.

Оператор обеспечивает переход между нулём и одной репликой при поступлении запросов. Если модель масштабирована до нуля реплик и поступает запрос через шлюз, оператор получает сигнал и создаёт одну реплику. Шлюз ожидает готовности реплики и маршрутизирует запрос.

Оператор экспортирует метрики состояния моделей на отдельной точке для систем наблюдаемости (Prometheus, VictoriaMetrics).

Загрузчик весов

Загрузчик весов — компонент платформенного контура, загружающий веса моделей из репозитория HuggingFace, зеркал и хранилищ, совместимых с S3, в кэш на постоянном томе. Загрузчик вызывается оператором при создании или обновлении кэша весов.

Веса размещаются атомарно через временный каталог с признаком готовности. После завершения загрузки загрузчик создаёт файл `.ready` в каталоге весов. Оператор проверяет наличие признака готовности перед созданием подов исполнения. Это исключает запуск модели с неполными весами.

Учётные данные для доступа к репозиториям и хранилищам передаются только через секреты Kubernetes. Загрузчик не читает учётные данные из конфигурационных файлов или переменных окружения.

Кэш весов является постоянным томом, разделяемым между репликами модели. При масштабировании модели до нескольких реплик каждая реплика монтирует один и тот же том с весами в режиме чтения.

Среды исполнения и автомасштабирование

Платформа поддерживает следующие движки обслуживания моделей:

- vLLM — высокопроизводительный движок для больших языковых моделей с оптимизацией памяти (PagedAttention);
- SGLang — движок с поддержкой структурированного вывода и ограничений грамматики;
- Ollama — движок для локального обслуживания моделей малого и среднего размера;
- FasterWhisper — движок для распознавания речи на основе Whisper с ускорением CTranslate2.

Движок задаётся в профиле исполнения модели. Под исполнения создаётся с соответствующим образом контейнера и параметрами запуска.

Поддерживается режим внешнего поставщика для проксирования запросов к OpenAI-совместимым внешним сервисам без локальных подов исполнения. В этом режиме оператор не создаёт поды, а шлюз маршрутизирует запросы к удалённому сервису.

Автомасштабирование моделей выполняется в два этапа:

1. **Переход 0→1.** Оператор платформенного контура создаёт одну реплику при поступлении первого запроса через шлюз. Шлюз ожидает готовности реплики и маршрутизирует запрос.
2. **Масштабирование 1→N.** KEDA (Kubernetes Event-Driven Autoscaling) масштабирует число реплик на основе метрик нагрузки из Prometheus или VictoriaMetrics. KEDA следит за длиной очереди запросов, задержками обработки и использованием графических процессоров.

Без установленного KEDA модели работают с фиксированным числом реплик, заданным в конфигурации развёртывания.

Расписание доступности модели управляет автомасштабированием: вне расписания оператор масштабирует модель до нуля реплик, в пределах расписания — поддерживает минимальное число реплик или разрешает масштабирование от нуля.

Наблюдаемость

Платформенный контур интегрируется с системами наблюдаемости:

- Prometheus / VictoriaMetrics — сбор метрик состояния моделей, подов, запросов, задержек, использования графических процессоров;
- Grafana — визуализация метрик и дашборды;
- Loki — сбор и поиск журналов подов исполнения;
- OpenTelemetry — трассировка распределённых запросов;
- DCGM — мониторинг графических процессоров NVIDIA (температура, использование памяти, утилизация).

Оператор и шлюз экспортируют метрики на отдельных точках /metrics для систем наблюдаемости. Консоль продуктового контура получает метрики здоровья развёртываний через продуктовый программный интерфейс без прямого обращения к Prometheus.

Опционально платформенный контур может хранить диалоги пользователей с моделями в PostgreSQL. Эта функция отделена от обязательного продуктового журнала запросов. Хранение диалогов включается отдельной настройкой и ведётся в схеме платформенного контура с отдельной ролью базы данных.

6.6. Хранилища и фоновые обработчики

Платформа использует несколько типов хранилищ с разделением по функциям и ответственности.

PostgreSQL

PostgreSQL используется обоими контурами с разделением по схемам и ролям базы данных:

Продуктовый контур: - организации, пользователи, команды, роли, разрешения; - тарифы, подписки, ключи запросов; - политики доступа, квоты, расписания; - каталог конфигураций моделей; - продуктовый журнал запросов с метаданными и учётом потребления; - аудит изменений конфигурации.

Платформенный контур: - учёт платформы (метрики подов, ресурсов, задержек); - опциональное хранение диалогов пользователей с моделями.

Разделение по схемам и ролям обеспечивает изоляцию данных контуров. Продуктовый контур не имеет доступа к схеме платформенного контура и наоборот.

Резервное копирование базы данных выполняется штатными средствами PostgreSQL с учётом требований заказчика к восстановлению.

ValKey

ValKey (форк Redis) используется для хранения быстродействующих структур данных:

- кэш ключей доступа для быстрой проверки на входном шлюзе APISIX;

- резервы квот пользователей и организаций;
- счётчики быстрых лимитов запросов (rate limit);
- события учёта перед фиксацией в PostgreSQL;
- кэш контекста исполнения запросов.

ValKey работает в режиме кэша с временем жизни записей. При недоступности ValKey входной шлюз APISIX читает ключи доступа из PostgreSQL с увеличенной задержкой. Проверки политик и резервирование квот при недоступности ValKey временно блокируются до восстановления доступности.

Kafka

Kafka используется как поток событий запросов к моделям. После каждого запроса шлюз публикует событие в Kafka с метаданными запроса, ответа и учёта. Фоновые обработчики продуктового контура читают поток событий для:

- фиксации потребления в продуктивном журнале;
- агрегации аналитики;
- отправки исходящих уведомлений (webhooks).

Сбой Kafka не блокирует ответ клиенту. Событие публикуется асинхронно после возврата ответа. Повторная доставка использует стабильный идентификатор события для идемпотентной обработки получателем.

Объектное хранилище

Объектное хранилище, совместимое с S3, используется для:

- хранения весов моделей, загруженных из репозитория;
- резервных копий кэша весов;
- опционального хранения содержимого запросов и ответов при включении соответствующей политики.

Доступ к объектному хранилищу выполняется через учётные данные, переданные в секретах Kubernetes. Загрузчик весов и компоненты платформенного контура читают и пишут объекты без доступа к файловым системам подов.

6.7. Дополнительные сервисы экосистемы

Продуктовый контур предоставляет доступ к дополнительным сервисам экосистемы, состав и функции которых определяются конкретной поставкой.

MCP Hub

MCP Hub — единый корпоративный центр для управления серверами протокола взаимодействия с моделями (Model Context Protocol). MCP Hub позволяет организациям публиковать, регистрировать и предоставлять доступ к корпоративным серверам MCP для использования приложениями и агентами.

Детальный набор функций, программные интерфейсы и правила публикации серверов MCP определяются отдельно. На момент подготовки технического описания полная спецификация MCP Hub отсутствует.

RAG-сервис

RAG-сервис (Retrieval-Augmented Generation) обеспечивает доступ к единому корпоративному сервису работы с документами и знаниями организации. RAG-сервис предоставляет индексирование документов, семантический поиск и интеграцию с моделями для ответов на вопросы по корпоративным документам.

Детальный набор функций, программные интерфейсы и интеграция с каталогом моделей определяются отдельно.

Маркетплейс навыков Skills

Маркетплейс Skills — торговая площадка для публикации, поиска и использования навыков (skills) — повторно используемых шаблонов запросов, цепочек вызовов моделей и интеграций с инструментами. Организации могут публиковать собственные навыки и использовать навыки других организаций.

Детальный набор функций, программные интерфейсы и правила публикации навыков определяются отдельно.

6.8. Функции развития

Следующие функции находятся в стадии разработки или планирования и не включены в текущую поставку:

Мультимодальность

Поддержка мультимодальных моделей (текст, изображения, аудио, видео) с обработкой вложений и потоковой передачей медиа. Требуется расширения контракта шлюза запросов и интеграции с движками, поддерживающими мультимодальность.

Задачи CPU и GPU

Разделение моделей по требованиям к ресурсам: модели CPU без графических процессоров (малые языковые модели, векторные представления) и модели GPU (большие языковые модели, распознавание речи). Требуется отдельных классов исполнения и политик размещения.

Иерархии организаций

Поддержка вложенных организаций (родительская организация, дочерние организации) с наследованием политик и квот. Позволяет управлять доступом к моделям в структурах холдингов и крупных корпораций.

Прямая политика доступа (ADP)

Attribute-Based Access Control с проверкой атрибутов пользователя, ресурса и контекста запроса. Расширяет политики доступа за пределы членства в организации и команде.

Каскадная маршрутизация

Автоматический выбор модели из списка кандидатов по критериям (стоимость, задержка, качество, доступность). Позволяет строить отказоустойчивые конфигурации с резервными моделями.

Готовность этих функций, контракты интеграции и сроки поставки определяются дорожной картой продукта и требуют отдельного согласования.

7. Системные требования

7.1. Варианты размещения и состав конфигурации

АСТРА ИИ [ХАБ] предназначен для размещения в одном из следующих вариантов:

- Кластер заказчика — развёртывание в собственной инфраструктуре с полным управлением ресурсами и данными.
- Облачная инфраструктура — размещение в поддерживаемых облачных платформах с изоляцией контуров по организациям.
- Изолированный стенд — автономная конфигурация без доступа к внешним сетевым ресурсам; требует отдельной схемы доставки зависимостей и весов моделей.

Все варианты поставляются из единой кодовой базы и различаются только конфигурацией; ответвления кода не создаются.

Минимальный платформенный состав

Минимальная конфигурация включает:

оркестратор контейнеров (Kubernetes);

средство установки (Helm);

оператор управления моделями;

шлюз запросов к моделям;

пользовательские ресурсы для жизненного цикла моделей.

Эта конфигурация позволяет запускать модели и обрабатывать запросы, но не включает продуктовую плоскость управления (консоль, организации, подписки, квоты, аналитику).

Промышленный продуктовый режим

Полный продуктовый состав дополнительно требует:

базу данных для хранения журнала запросов, учёта потребления и аналитики;

хранилище данных в памяти для кэширования ключей, резервирования квот и событий учёта;

систему идентификации и аутентификации пользователей;

единый шлюз доступа с проверкой подлинности и маршрутизацией;

серверную часть продукта (плоскости управления и исполнения);

веб-консоль администрирования.

Промышленный режим обеспечивает полный цикл управления доступом, учёта, квотирования и мониторинга использования моделей.

7.2. Программные зависимости

7.2.1. Базовая платформа

Компонент	Назначение	Условие обязательности	Проверенная версия	Требует подтверждения
Kubernetes	Оркестрация контейнеров, пользовательские ресурсы, жизненный цикл подов	Базовая зависимость	≥ 1.29	Верхняя граница и ОС узлов
Helm	Установка и управление конфигурацией продукта	Базовая зависимость	3.x	Минимальная версия

Примечания:

- Kubernetes версии ниже 1.29 не поддерживаются.
- Поддерживаемые архитектуры процессоров (x86-64, ARM64) и операционные системы узлов требуют уточнения в документации к релизу.
- Средства контейнеризации (containerd, CRI-O) должны соответствовать требованиям выбранной версии Kubernetes.

7.2.2. Продуктовый режим

Компонент	Назначение	Условие обязательности	Проверенная версия	Требует подтверждения
PostgreSQL	Журнал запросов, учёт потребления, состояние организаций и подписок, аналитика	Обязателен для продуктового режима	14.x–16.x	Точный диапазон версий
ValKey (или Redis)	Кэш ключей запросов, резервирование квот, поток событий учёта с проекцией в базу данных	Обязателен в промышленном режиме	ValKey 7.2+ или Redis 7.0+	Совместимость версий
Dex или Keycloak	Система идентификации (IdP), единый вход (OIDC), синхронизация организаций	Базовый для консоли и клиентов с OIDC	Dex 2.37+, Keycloak 22+	Поддерживаемые версии
APISIX	Шлюз входа: проверка	Базовый для промышленного	3.x	Точная версия

Компонент	Назначение	Условие обязательности	Проверенная версия	Требует подтверждения
	OIDC-токенов, определение ключей запросов, перезапись недоверенных заголовков, маршрутизация	о режима		

Примечания:

- PostgreSQL используется для хранения журнала запросов продуктовой части; при включении учёта и диалогов на платформенной стороне создаются отдельные схемы или роли.
- ValKey предпочтителен в новых установках; Redis поддерживается для совместимости с существующими инфраструктурами.
- При отсутствии ValKey/Redis продуктовый режим не может обеспечить резервирование квот и кэширование ключей запросов; система переходит в режим только управления моделями.
- Выбор между Dex и Keycloak определяется политикой интеграции с корпоративными системами идентификации заказчика.

7.2.3. Зависимости по функции

Компонент	Назначение	Условие обязательности	Проверенная версия	Требует подтверждения
S3-совместимое хранилище	Хранение весов моделей, холодное хранение диалогов, резервные копии	По функции (загрузка весов из S3, архивирование диалогов)	MinIO, AWS S3, совместимые	Требования к производительности
KEDA	Автомасштабирование подов исполнения моделей от 1 до N реплик	По необходимости (автомасштабирование)	2.x	Совместимость с метриками
Prometheus или VictoriaMetrics	Сбор метрик для автомасштабирования и мониторинга	По необходимости (наблюдаемость, автомасштабирование)	Prometheus 2.x, VictoriaMetrics 1.x	Требования к хранилищу метрик
Kafka	Поток событий запросов к моделям для	По функции (интеграция с внешними)	3.x	Производительность и размер очереди

Компонент	Назначение	Условие обязательности	Проверенная версия	Требует подтверждения
	внешних подписчиков	системами аналитики)		
Grafana	Панели мониторинга и визуализация метрик	Опционально	9.x–10.x	—
Loki	Агрегация журналов	Опционально	2.x	—
OpenTelemetry Collector	Трассировка распределённых запросов	Опционально	0.x	—
DCGM Exporter	Метрики GPU (utilization, memory, temperature)	По функции (наличие GPU)	3.x	—

Примечания:

- Без S3-совместимого хранилища поддерживается только загрузка весов моделей из локальных томов или зеркал HuggingFace.
- Переход масштаба $0 \leftrightarrow 1$ реплики модели выполняет оператор управления моделями; KEDA управляет только диапазоном $1 \leftrightarrow N$.
- Платформа работоспособна без средств наблюдаемости (Prometheus, Grafana, Loki), если соответствующая функция отключена в конфигурации установки.
- При отсутствии Kafka события запросов к моделям не публикуются во внешние системы, но внутренний учёт продолжает работать через ValKey и PostgreSQL.

7.2.4. Движки исполнения моделей

Движок	Назначение	Поддерживаемые типы моделей	Требует подтверждения
vLLM	Обслуживание больших языковых моделей с оптимизацией по памяти и пропускной способности	Языковые модели (LLM)	Версия, требования к GPU
SGLang	Обслуживание языковых моделей с поддержкой структурированной генерации	Языковые модели (LLM)	Версия, требования к GPU
Ollama	Локальные модели в упрощённом контуре без сложной конфигурации	Языковые модели малого и среднего размера	Версия
FasterWh	Распознавание речи из	Аудио → текст	Версия, требования к

Движок	Назначение	Поддерживаемые типы моделей	Требует подтверждения
isper	аудио		CPU/GPU
external	Прокси-режим к внешнему OpenAI-совместимому API без локальных подов исполнения	Любые (делегирование внешнему поставщику)	Контракт совместимости

Примечания:

- Движки vLLM и SGLang требуют GPU для производительной работы с большими моделями; возможность работы на CPU с малыми моделями ограничена и требует отдельной проверки.
- Режим external используется для интеграции с облачными поставщиками моделей (OpenAI, Azure OpenAI, cloud.ru и др.) без размещения локальных вычислительных ресурсов.

7.3. Вычислительные ресурсы и хранение

Раздел определяет рамку требований к размещению компонентов продукта. Конкретные значения зависят от выбранного состава, нагрузки и профиля использования; минимальные требования, не подтверждённые измерениями, не приводятся.

7.3.1. Серверные компоненты

Компонент	Минимальные ресурсы (CPU / RAM)	Рекомендуемые ресурсы	Примечания
Оператор управления моделями	500m / 512 MiB	1 CPU / 1 GiB	Согласование состояния; нагрузка зависит от числа моделей и частоты изменений
Шлюз запросов (data-plane)	1 CPU / 1 GiB	2 CPU / 2 GiB	Маршрутизация без хранения состояния; масштабируется горизонтально; требования растут с числом одновременных запросов
Серверная часть продукта (control-plane)	500m / 512 MiB	1 CPU / 1 GiB	Плоскость управления (консоль, организации, деплой, политики); низкая нагрузка

Компонент	Минимальные ресурсы (CPU / RAM)	Рекомендуемые ресурсы	Примечания
Серверная часть продукта (data-plane)	1 CPU / 1 GiB	2 CPU / 2 GiB	Плоскость исполнения (проверка подписок, квот, расписаний до вызова модели); масштабируется по числу запросов
Обработчик учёта (usage-worker)	500m / 512 MiB	1 CPU / 1 GiB	Проекция событий учёта из ValKey в PostgreSQL
Веб-консоль (frontend)	100m / 128 MiB	500m / 256 MiB	Статическая раздача через nginx или CDN
HTTP-адаптер к Kubernetes	200m / 256 MiB	500m / 512 MiB	Мост REST → пользовательские ресурсы; вызывается только серверной частью

Примечания:

- Значения приведены для оценки; точные требования определяются профилем нагрузки (число запросов в секунду, размер организаций, частота изменений конфигурации).
- Серверная часть может развёртываться в режиме «только управление», «только исполнение» или совмещённом; проверки готовности учитывают выбранный режим.

7.3.2. Размещение моделей

Требования к ресурсам для исполнения моделей не могут быть указаны без выбора конкретных моделей и движков. Каждая модель требует:

- CPU: достаточно для загрузки весов и обработки токенов; в отсутствие GPU вся нагрузка ложится на процессор.
- Память (RAM): должна вмещать веса модели и активации; для больших моделей (≥ 30 млрд параметров) требуется от 60 ГБ и выше.
- GPU: тип, число устройств и видеопамять зависят от размера модели и требований к пропускной способности; типовые значения:
 - Малые модели (до 7 млрд параметров): $1 \times$ GPU с ≥ 16 ГБ видеопамяти (например, NVIDIA Tesla T4, RTX 4090).
 - Средние модели (7–13 млрд параметров): $1\text{--}2 \times$ GPU с ≥ 24 ГБ (например, NVIDIA A10, RTX 4090).

- Большие модели (30–70 млрд параметров): 2–8× GPU с ≥ 40 ГБ каждый (например, NVIDIA A100 40 ГБ, H100).
- Очень большие модели (более 70 млрд параметров): множество GPU с распределением модели (tensor parallelism, pipeline parallelism).
- Диск (хранилище весов): размер определяется числом моделей и их размерами; типовая модель 7B занимает 15–30 ГБ в зависимости от квантизации; кэш весов снижает время холодного старта, но требует быстрого хранилища (NVMe SSD).

Внешний режим (external): при использовании внешнего поставщика моделей локальные поды исполнения не создаются; требования к GPU и памяти отсутствуют.

7.3.3. Зависимые службы

Служба	Минимальные ресурсы	Объём хранилища	Примечания
PostgreSQL	1 CPU / 2 GiB	50 ГБ (начальный)	Объём растёт с числом запросов, глубиной хранения журнала и числом диалогов
ValKey/ Redis	500m / 1 GiB	В памяти; персистентность опционально	Кэш ключей и резервы квот требуют малых объёмов (< 1 ГБ для тысяч ключей); поток событий учёта — промежуточный буфер до проекции в PostgreSQL
Dex	100m / 128 MiB	Без хранилища (состояние в Kubernetes)	—
Keycloak	500m / 1 GiB	10 ГБ (база данных)	Требования растут с числом пользователей и организаций
APISIX	500m / 512 MiB	Без хранилища (конфигурация в etcd или ConfigMap)	Масштабируется горизонтально

Примечания:

- Объём хранилища PostgreSQL напрямую зависит от политики хранения журнала запросов и диалогов; при архивировании холодных данных в S3 требования к диску снижаются.
- ValKey/Redis используется как промежуточный буфер; основное хранилище учёта — PostgreSQL.

7.3.4. Хранение и потоки данных

- Журнал запросов к моделям: каждый запрос фиксируется с метаданными (организация, пользователь, модель, потребление токенов, задержки, статус);

типовой размер записи — 1–5 КБ. При нагрузке 100 запросов/с журнал накапливает ~25–125 ГБ/месяц.

- Диалоги (опционально): при включении хранения диалогов сохраняются промпты и ответы; объём зависит от средней длины запроса и политики хранения. Для 1 млн запросов со средней длиной 2 КБ потребуется ~2 ГБ.
- События в Kafka (опционально): буфер событий для внешних подписчиков; время хранения и размер очереди НАСТРАиваются в конфигурации Kafka.
- Кэш весов моделей: для быстрого старта подов исполнения рекомендуется локальное хранилище (NVMe SSD) объёмом, достаточным для всех активных моделей; типовое значение — 200–500 ГБ для нескольких моделей среднего размера.

7.4. Сеть и интеграции

7.4.1. Сетевая связность компонентов

- Внешний доступ: клиенты обращаются к единой точке входа (APISIX) по HTTPS; внутренние компоненты взаимодействуют внутри кластера по HTTP.
- Интеграция с корпоративной IdP: APISIX проверяет OIDC-токены через Dex или Keycloak; необходима сетевая доступность поставщика идентификации с узлов APISIX.
- Загрузка весов моделей: при использовании внешних источников (HuggingFace, зеркала, S3) узлы с подами загрузчика должны иметь доступ к соответствующим конечным точкам; в изолированном стенде требуется локальное зеркало или ручная доставка весов.
- Внешние поставщики моделей: при использовании режима external шлюз запросов должен иметь доступ к конечным точкам API внешних поставщиков (OpenAI, Azure OpenAI, cloud.ru и др.).
- Kafka (опционально): при публикации событий в Kafka серверная часть должна иметь сетевой доступ к брокерам Kafka.

7.4.2. Сертификаты и шифрование

- TLS на входе: рекомендуется завершение TLS на APISIX с действительным сертификатом для публичного имени хоста.
- Внутрикластерное взаимодействие: по умолчанию HTTP; TLS между компонентами (mutual TLS) поддерживается при включении сетевых политик и настройке соответствующих секретов.
- Webhook проверки ресурсов: проверяющий webhook оператора требует TLS; сертификат создаётся автоматически при установке.

7.4.3. Требования к изолированному стенду

В конфигурации без доступа к внешним сетям:

- Все образы контейнеров должны быть доставлены в локальный реестр образов (Docker Registry, Harbor).
- Веса моделей загружаются из локального S3-совместимого хранилища или предварительно размещаются на постоянных томах.
- Зеркала зависимостей (Helm-репозитории, пакеты ОС) размещаются во внутренней инфраструктуре.

- Обновления и исправления доставляются через согласованный канал без прямого доступа в интернет.

7.5. Клиентская среда и совместимость

7.5.1. Веб-консоль

Поддерживаемые браузеры (требует подтверждения в документации к релизу):

- Google Chrome / Chromium ≥ 100
- Mozilla Firefox ≥ 100
- Apple Safari ≥ 15
- Microsoft Edge ≥ 100

Консоль использует современные веб-технологии (ES2020, WebSocket); устаревшие браузеры не поддерживаются.

7.5.2. Клиенты API

Запросы к моделям через OpenAI-совместимый API не требуют специфичного клиентского программного обеспечения; достаточно поддержки HTTP/1.1, JSON и (опционально) потоковой передачи (SSE — Server-Sent Events).

Официальные библиотеки OpenAI (Python, Node.js, Go и др.) совместимы при указании базового URL продукта.

Примечание: поддержка настольных операционных систем и средств разработки для клиента Astra Code определяется отдельным продуктом и не дублируется в данном описании.

7.6. Требования к установке и сопровождению

7.6.1. Права и доступ для установки

Установка через Helm требует:

- Права на создание пространств имён (Namespace) в кластере Kubernetes.
- Права на создание пользовательских ресурсов (CustomResourceDefinition, CRD).
- Права на развёртывание контроллеров с доступом к API кластера (ClusterRole, ClusterRoleBinding).
- Доступ к реестру образов контейнеров (публичному или локальному).

Установку выполняет администратор кластера или учётная запись с достаточными правами; подробные инструкции приведены в руководстве администратора.

7.6.2. Проверка готовности

После установки необходимо проверить:

- Готовность подов всех компонентов (оператор, шлюз, серверная часть, веб-консоль).
- Доступность точек проверки живости и готовности (/health/live, /health/ready) серверной части.
- Создание начальных пользовательских ресурсов (тестовый Tenant, APIClient, AIModel).

- Доступность веб-консоли через точку входа (APISIX).
- Успешную аутентификацию пользователя через OIDC.

Команды и сценарии проверки приведены в руководстве администратора.

7.6.3. Обновления и совместимость

- Обновление продукта выполняется через `helm upgrade` с новыми значениями конфигурации и версией образов.
- Обновление не должно изменять имя хоста точки входа и разрывать существующие подключения клиентов.
- При обновлении схемы базы данных выполняются миграции; отказоустойчивость миграций (откат при ошибке) обеспечивается настройками задания миграции (`backoffLimit`, `activeDeadlineSeconds`).
- Обновление пользовательских ресурсов (CRD) может требовать пересоздания ресурсов; процедура описывается в примечаниях к релизу.

Примечание: совместимость обновлений между версиями, процедуры отката и резервного копирования состояния описаны в руководстве администратора; данная глава определяет только общую рамку требований.