Искра Мессенджер – корпоративный мессенджер с ИИ версия 1.0

Руководство администратора и DevOps

Содержание

Документ предназначен для DevOps-инженеров, системных администраторов и инженеров эксплуатации заказчика, которые отвечают за установку, настройку, обновление и промышленную эксплуатацию Искры Мессенджер в локальном корпоративном контуре. Документ описывает текущую серверную реализацию Искры, ее эксплуатационные зависимости, порядок промышленного развертывания, проверки готовности, резервное копирование, восстановление, обновления и диагностику.

Документ не заменяет архитектурную спецификацию. Архитектурная спецификация определяет целевую топологию и принципы масштабирования; это руководство переводит эти требования в действия DevOps-команды.

1. Область применения

Искра Мессенджер устанавливается как набор серверных компонентов вокруг центрального сервиса Gateway. Gateway обслуживает HTTP API, WebSocket, авторизацию, сессии, пользователей, пространства, диалоги, сообщения, очереди, вложения, звонки, OTA-обновления и интеграционные обратные вызовы.

Промышленное внедрение должно иметь заполненный **профиль внедрения Искры**. Профиль внедрения является рабочим документом проекта установки и фиксирует:

- включенные модули Искры;
- публичные домены и внутренние адреса сервисов;
- сетевые зоны и правила доступа;
- параметры PostgreSQL, Redis, NATS, объектного хранилища и LiveKit;
- способ хранения секретов и сертификатов;
- RPO/RTO, резервное копирование и порядок восстановления;
- правила масштабирования;
- требования к журналам, метрикам, оповещениям и централизованному сбору журналов;
- ответственных владельцев со стороны заказчика.

До заполнения профиля внедрения промышленный запуск не считается подготовленным.

2. Состав поставки и роли компонентов

Gateway. Центральный серверный сервис Искры: API, WebSocket, авторизация, сессии, сообщения, пространства, очереди, вложения, звонки, OTA и интеграционные адреса обратных вызовов. Не хранит постоянное локальное состояние.

Web. Web-клиент Искры, собранный как Vite/React SPA. Отдает пользовательский интерфейс и проксирует запросы к серверному контуру.

Webchat service. Публичный webchat-контур, события Webchat и обратные вызовы доставки. Состояние хранится в PostgreSQL.

Assistant. ИИ-агенты, сводки, интеграции календарей, Bitrix, HeadHunter и web-поиска. Не хранит постоянное локальное состояние.

Event-organizer. Сценарии событий и обработка ответов участников. Состояние хранится в PostgreSQL.

Transcription worker и Sticker normalizer. Фоновые обработчики расшифровки и нормализации медиа-артефактов. Не хранят постоянное локальное состояние.

Telegram bridge и MAX bridge. Bridge-сервисы внешних каналов. Состояние внешних событий ведется через Gateway и PostgreSQL.

PostgreSQL. Источник истины для пользователей, сессий, сообщений, диалогов, очередей, агентов и конфигураций.

Redis. Краткоживущее состояние среды выполнения, служебные кэши и присутствие.

NATS. Распространение событий реального времени и очереди.

Объектное хранилище S3/MinIO. Вложения, записи, OTA-файлы, загрузки и артефакты.

LiveKit и LiveKit egress. Звонки, конференции, медиа-сеансы, запись и выгрузка медиа в объектное хранилище.

LiveKit SIP / Kamailio. SIP-сценарии и телефония при включении модуля. Конкретная схема зависит от PBX и сетевой модели заказчика.

Prometheus / Grafana. Метрики и панели мониторинга при включении эксплуатационного контура.

GlitchTip. Отчетность об ошибках при включении. Использует отдельную БД и хранилище загрузок.

Искра не должна хранить бизнес-состояние на локальном диске прикладных контейнеров или виртуальных машин. Данные, которые должны пережить рестарт или масштабирование, размещаются в PostgreSQL, объектном хранилище или другом согласованном сервисе состояния.

3. Поддерживаемая эксплуатационная модель

Промышленная установка Искры строится как набор контейнеризованных сервисов без постоянного локального состояния у прикладного слоя. Поддерживаемая логика эксплуатации:

- несколько экземпляров Gateway, Web, Webchat service, Assistant, bridge-сервисов и фоновых обработчиков могут работать параллельно;
- все экземпляры используют общий PostgreSQL, Redis, NATS и объектное хранилище;
- балансировщик направляет HTTPS-трафик на Web/Gateway/Webchat;
- WebSocket-соединения могут обслуживаться любым экземпляром Gateway при общем контуре событий;
- LiveKit масштабируется отдельно от контура сообщений;
- миграции PostgreSQL выполняются контролируемо до запуска новой версии прикладного слоя;
- секреты и сертификаты принадлежат инфраструктурному контуру заказчика.

Искра может быть развернута на виртуальных машинах с Docker Compose или в Kubernetes при соблюдении той же модели состояния: прикладные контейнеры без локального постоянного состояния, общие сервисы с состоянием, общее объектное хранилище, единая схема секретов, корректные проверки готовности и жизнеспособности, контролируемый запуск миграций. Наличие конкретных Helm-чартов или шаблонов оркестрации не является предпосылкой этой спецификации.

4. Обязательная инфраструктура заказчика

Перед установкой заказчик предоставляет или утверждает следующие ресурсы.

4.1 Вычислительный контур

- узлы или пространство имен оркестратора для Gateway, Web, Webchat service, Assistant, Event-organizer, bridge-сервисов и фоновых обработчиков;
- отдельную емкость для LiveKit и LiveKit egress;
- лимиты CPU/RAM для каждого контура по результатам расчета емкости;
- возможность горизонтального добавления экземпляров прикладных сервисов;
- регламент обновления образов и перезапуска сервисов.

4.2 Сервисы с состоянием

- PostgreSQL 17 или совместимая корпоративная управляемая версия;
- Redis 7 или совместимая корпоративная версия;
- NATS 2.10 с JetStream при включенных сценариях реального времени и очередей;
- S3-совместимое объектное хранилище или MinIO;
- отдельная БД для GlitchTip, если включена отчетность об ошибках.

4.3 Публичные точки доступа

Профиль внедрения фиксирует публичные URL:

- Web-клиент Искры;
- Gateway API и WebSocket;
- Webchat, если включен;
- LiveKit signaling URL;
- SIP/PBX точка доступа для телефонии, если включен SIP-модуль;
- манифест OTA и файлы OTA, если включены мобильные OTA;
- публичные URL обратных вызовов для Telegram, MAX, event organizer, billing и OAuth-интеграций Assistant.

Публичный пользовательский трафик должен идти по HTTPS. WebSocket публикуется поверх TLS. Медиа LiveKit требует отдельной проверки UDP-доступности и NAT-модели. SIP/PBX подключение публикуется только при включенной телефонии; профиль внедрения фиксирует

SIP-домен или IP, транспорт UDP/TCP/TLS, публичный контактный адрес, NAT-модель и правила доступа от PBX или SIP-провайдера.

5. Сетевая матрица

Пользовательский входящий трафик. Клиенты Web, Mobile и Desktop обращаются к Web и Gateway по HTTPS и WSS. Для звонков клиенты обращаются к LiveKit по HTTPS/WSS и используют UDP-медиапотоки. Внешние каналы обращаются к Gateway или bridge-сервисам по HTTPS для веб-перехватчиков и обратных вызовов.

Gateway к сервисам состояния. Gateway обращается к PostgreSQL по TCP 5432 для пользователей, сессий, сообщений, диалогов, очередей и агентов. Redis используется по TCP 6379 для состояния среды выполнения и кэшей. NATS используется по TCP 4222 для событий реального времени. Объектное хранилище используется по HTTPS или внутренней S3-точке доступа для вложений, OTA и записей.

Gateway к прикладным сервисам. Gateway обращается к LiveKit по HTTP/WSS для выдачи и проверки медиа-токенов, к Assistant по HTTP для ИИ-агентов, сводок и инструментов, к Event-organizer по HTTP для сценариев событий.

Assistant к Gateway и внешним поставщикам. Assistant обращается к Gateway по HTTP/WSS для аренды среды выполнения ботов и потоковых событий. При включении ИИ-функций Assistant обращается к LLM-поставщику по HTTPS. При включении интеграций Assistant обращается к OAuth/API поставщиков Bitrix, HeadHunter, Google, Yandex и Apple по HTTPS.

Медиа и эксплуатация. LiveKit egress обращается к объектному хранилищу по S3 API для записей и выгрузок. Prometheus обращается к Gateway, Assistant, Event-organizer и экспортерам по HTTP для сбора метрик. Эксплуатационный контур заказчика обращается к сервисам через SSH или API платформы для обновлений, диагностики и сбора журналов.

SIP/PBX подключение. При включенной телефонии PBX или SIP-провайдер заказчика подключается к SIP/Kamailio-контуру по согласованному транспорту UDP/TCP/TLS, обычно на порты 5060/5061/5062 согласно профилю внедрения. SIP/Kamailio взаимодействует с LiveKit SIP и телефоническим bridge-сервисом внутри медиа-контура; Gateway остается управляющим сервисом для прав, пользователей, звонков и событий, но не передает через себя RTP-медиапоток.

Внутренний трафик может защищаться сетевыми политиками частной сети, mTLS или сервисной сеткой. Выбранный подход фиксируется в профиле внедрения.

6. DNS, TLS и публикация

Промышленная установка должна иметь:

- DNS-записи для публичных точек доступа Искры;
- TLS-сертификаты для Web/Gateway/Webchat/LiveKit signaling/OAuth callback;
- балансировщик или ingress, поддерживающий WebSocket;
- корректные лимиты размера тела запроса для вложений;
- отдельные правила публикации для `/v1/updates/manifest`, `/v1/updates/files/...` и `/downloads/...`, если включены OTA и файлы загрузки;

- закрытый прямой доступ к PostgreSQL, Redis, NATS, MinIO/S3 console, Prometheus и Grafana из публичной сети.

Gateway и Webchat не должны публиковать секретные административные интерфейсы. MinIO console, Grafana, Prometheus и GlitchTip admin UI публикуются только по политике заказчика и с отдельной защитой доступа.

7. Профиль конфигурации

Все параметры промышленной установки фиксируются в утвержденном профиле внедрения. Ниже перечислены группы конфигурации, которые должны быть заполнены до запуска.

7.1 Базовые параметры

- `APP_ENV=production` - режим запуска промышленного контура Искры. Рекомендуемое значение для промышленной установки: `production`.
- `PUBLIC_APP_URL` - внешний HTTPS-адрес пользовательского Web-клиента и публичных Gateway-точек. Рекомендуемое значение: основной публичный URL Искры в домене заказчика, например `https://iskra.company.example`.
- `DATABASE_URL` - строка подключения Gateway, Assistant и служебных процессов к основной БД PostgreSQL.
- `DATABASE_MAX_OPEN_CONNS` - максимальное число открытых соединений PostgreSQL на один экземпляр сервиса. Рекомендуемое начальное значение: `20-50` для Gateway, `5-20` для фоновых сервисов; суммарный лимит всех экземпляров должен быть ниже лимита PostgreSQL с резервом для администрирования.
- `DATABASE_MAX_IDLE_CONNS` - максимальное число простаивающих соединений PostgreSQL на один экземпляр сервиса. Рекомендуемое начальное значение: `5-10` для Gateway и `2-5` для фоновых сервисов.
- `DATABASE_CONN_MAX_LIFETIME` - максимальное время жизни соединения PostgreSQL в пуле. Рекомендуемое начальное значение: `30m`.
- `DATABASE_CONN_MAX_IDLE_TIME` - максимальное время простоя соединения PostgreSQL в пуле. Рекомендуемое начальное значение: `5m`.
- `REDIS_URL` - адрес Redis для краткоживущего состояния, кэшей и служебной синхронизации.
- `NATS_URL` - адрес NATS для событий реального времени и фоновой доставки.
- `JWT_SECRET` - секрет подписи пользовательских access/refresh-токенов Искры. Рекомендуемое значение: случайная строка не менее 32 байт, сгенерированная и хранимая в хранилище секретов заказчика.
- `REALTIME_INSTANCE_ID` - явное имя экземпляра Gateway для диагностики WebSocket-соединений и присутствия. Рекомендуемое значение: уникальное имя экземпляра, например имя pod/контейнера/виртуальной машины.

7.2 Аутентификация пользователей

- `OTP_PROVIDER` - выбранный промышленный поставщик одноразовых кодов или звонков подтверждения. Рекомендуемое значение: промышленный поставщик из договора

заказчика; для произвольного HTTP/SMS-поставщика используется `http`, для звонка подтверждения Ростелеком - профиль Ростелеком.

- `OTP_MODE` - режим проверки OTP; в промышленном контуре используется режим внешнего поставщика, не `dev`. Рекомендуемое значение: режим промышленного поставщика, зафиксированный в профиле внедрения.
- `OTP_TTL` - срок действия одноразового кода. Рекомендуемое значение: `5m`.
- `OTP_RESEND_COOLDOWN` - минимальный интервал между повторными отправками кода одному пользователю. Рекомендуемое значение: `60s`.
- `OTP_MAX_ACTIVE_REQUESTS` - максимальное число активных OTP-запросов для одного пользователя или телефона. Рекомендуемое значение: `3`.
- `OTP_MAX_ATTEMPTS` - максимальное число попыток ввода кода до блокировки сценария. Рекомендуемое значение: `5`.
- `AUTHENTICATOR_SECRET_ENCRYPTION_KEY` - ключ шифрования секретов TOTP/аутентификатора. Рекомендуемое значение: случайный ключ не менее 32 байт из хранилища секретов.
- `USER_PHONE_ENCRYPTION_KEY` - ключ шифрования телефонных номеров пользователей в БД. Рекомендуемое значение: случайный ключ не менее 32 байт из хранилища секретов.
- `INVITE_ONLY_SIGNUP_ENABLED` - включает регистрацию только по приглашениям. Рекомендуемое значение для закрытого корпоративного контура: `true`.
- `WAITLIST_ENABLED` - включает лист ожидания для пользователей без приглашения. Рекомендуемое значение для закрытого корпоративного контура: `false`.

Промышленный `OTP_MODE=dev` запрещен. Поставщик OTP/SMS должен быть выбран и проверен до допуска пользователей.

7.3 SMS/OTP поставщики

Используемые переменные зависят от выбранного поставщика:

- `SMS_PROVIDER` - тип SMS-поставщика для `OTP_PROVIDER=http` или SMS-режима. Рекомендуемое значение: конкретный поставщик из договора заказчика; для универсальной HTTP-интеграции - `http`.
- `SMS_API_URL` - HTTPS-адрес API произвольного SMS-поставщика. Рекомендуемое значение: промышленный HTTPS endpoint поставщика, не тестовый URL.
- `SMS_API_METHOD` - HTTP-метод отправки запроса к произвольному SMS-поставщику. Рекомендуемое значение: `POST`, если поставщик не требует другое.
- `SMS_API_CONTENT_TYPE` - значение `Content-Type` для запроса к произвольному SMS-поставщику. Рекомендуемое значение: `application/json`, если поставщик не требует другое.
- `SMS_API_AUTH_HEADER` - имя заголовка авторизации для произвольного SMS-поставщика. Рекомендуемое значение: заголовок из документации поставщика, обычно `Authorization`.

- `SMS_API_AUTH_VALUE` - секретное значение заголовка авторизации для произвольного SMS-поставщика. Рекомендуемое значение: токен промышленного контура из хранилища секретов; тестовый токен запрещен.
- `SMS_SENDER` - имя отправителя, отображаемое пользователю в SMS, если поставщик поддерживает sender ID. Рекомендуемое значение: зарегистрированное имя отправителя заказчика, согласованное с SMS-поставщиком.
- `SMS_REQUEST_TEMPLATE` - шаблон тела запроса к произвольному SMS-поставщику. Рекомендуемое значение: минимальный JSON-шаблон поставщика с полями номера, текста и идентификатора отправителя.
- `SIGMASMS_API_URL` - адрес API SigmaSMS. Рекомендуемое значение: промышленный HTTPS endpoint SigmaSMS из договора.
- `SIGMASMS_API_KEY` - ключ доступа SigmaSMS. Рекомендуемое значение: промышленный ключ из хранилища секретов.
- `SIGMASMS_SENDER` - имя отправителя SigmaSMS. Рекомендуемое значение: зарегистрированное имя отправителя заказчика.
- `BEELINE_SMS_API_URL` - адрес API Beeline SMS. Рекомендуемое значение: промышленный HTTPS endpoint Beeline из договора.
- `BEELINE_SMS_USERNAME` - имя учетной записи Beeline SMS. Рекомендуемое значение: промышленная учетная запись заказчика.
- `BEELINE_SMS_PASSWORD` - пароль учетной записи Beeline SMS. Рекомендуемое значение: пароль из хранилища секретов.
- `BEELINE_SMS_SENDER` - имя отправителя Beeline SMS. Рекомендуемое значение: зарегистрированное имя отправителя заказчика.
- `P1SMS_API_URL` - адрес API P1SMS. Рекомендуемое значение: промышленный HTTPS endpoint P1SMS из договора.
- `P1SMS_API_KEY` - ключ доступа P1SMS. Рекомендуемое значение: промышленный ключ из хранилища секретов.
- `P1SMS_SENDER` - имя отправителя P1SMS. Рекомендуемое значение: зарегистрированное имя отправителя заказчика.
- `P1SMS_PROJECT` - идентификатор проекта P1SMS, если используется в договоре поставщика. Рекомендуемое значение: промышленный project ID из личного кабинета P1SMS.
- `ROSTELECOM_AUTH_NUMBER_API_URL` - адрес API Ростелеком для подтверждения номера звонком. Рекомендуемое значение: промышленный HTTPS endpoint Ростелеком из договора.
- `ROSTELECOM_CLIENT_ID` - идентификатор клиента Ростелеком. Рекомендуемое значение: промышленный client ID заказчика.
- `ROSTELECOM_SIGN_KEY` - ключ подписи запросов Ростелеком. Рекомендуемое значение: ключ подписи из хранилища секретов.

- `ROSTELECOM_FROM_NUMBER` - номер, с которого выполняется звонок подтверждения. Рекомендуемое значение: выделенный номер заказчика, разрешенный договором Ростелеком.

7.4 Объектное хранилище и вложения

- `OBJECT_STORAGE_ENDPOINT` - S3-совместимая точка доступа Gateway к объектному хранилищу. Рекомендуемое значение: внутренний HTTPS endpoint объектного хранилища из прикладной зоны.
- `EGRESS_OBJECT_STORAGE_ENDPOINT` - S3-совместимая точка доступа LiveKit egress к объектному хранилищу. Рекомендуемое значение: внутренний endpoint, доступный из медиа-зоны; допускается то же значение, что `OBJECT_STORAGE_ENDPOINT`, если маршрутизация одинакова.
- `OBJECT_STORAGE_ACCESS_KEY` - ключ доступа к объектному хранилищу для Gateway и фоновых обработчиков. Рекомендуемое значение: отдельная сервисная учетная запись с доступом только к корзине Искры.
- `OBJECT_STORAGE_SECRET_KEY` - секретный ключ объектного хранилища. Рекомендуемое значение: секрет сервисной учетной записи из хранилища секретов.
- `OBJECT_STORAGE_BUCKET` - корзина для вложений, записей, OTA-файлов и артефактов Искры. Рекомендуемое значение: отдельная корзина для конкретного контура, например `iskra-prod`.
- `OBJECT_STORAGE_USE_SSL` - включает TLS при обращении к S3-совместимой точке доступа. Рекомендуемое значение для промышленного контура: `true`.
- `MESSAGE_ATTACHMENT_MAX_BYTES` - максимальный размер одного вложения в сообщении. Рекомендуемое начальное значение: `104857600` (100 МБ), если политика заказчика не задает меньший лимит.
- `MESSAGE_MULTIPART_MAX_BYTES` - максимальный размер multipart-запроса загрузки. Рекомендуемое начальное значение: на 10-20% выше `MESSAGE_ATTACHMENT_MAX_BYTES`.
- `OBJECT_STAGING_CLEANUP_INTERVAL` - период запуска очистки промежуточных объектов. Рекомендуемое значение: `15m`.
- `OBJECT_STAGING_CLEANUP_STALE_AFTER` - возраст промежуточного объекта, после которого он считается устаревшим. Рекомендуемое значение: `24h`.
- `OBJECT_STAGING_CLEANUP_BATCH_SIZE` - максимальное число промежуточных объектов, удаляемых за один проход. Рекомендуемое значение: `100-500` в зависимости от размера хранилища и допустимой нагрузки.

7.5 LiveKit, звонки и телефония

- `LIVEKIT_URL` - внутренний адрес LiveKit для Gateway. Рекомендуемое значение: внутренний `http://` или `ws://` endpoint LiveKit в медиа-зоне, доступный только из прикладного контура.
- `LIVEKIT_PUBLIC_URL` - публичный WSS-адрес LiveKit для клиентов. Рекомендуемое значение: `wss://` URL в публичном домене заказчика.

- `LIVEKIT_API_KEY` - ключ API LiveKit, используемый Gateway для выпуска медиа-токенов. Рекомендуемое значение: отдельный API key LiveKit для Искры.
- `LIVEKIT_API_SECRET` - секрет API LiveKit, используемый Gateway для подписи медиа-токенов. Рекомендуемое значение: секрет LiveKit из хранилища секретов, не короче 32 байт.
- `LIVEKIT_NODE_IP` - маршрутизируемый IP узла LiveKit для WebRTC-кандидатов. Рекомендуемое значение: IP или адрес, достижимый клиентами согласно NAT-модели заказчика.
- `LIVEKIT_RTC_UDP_START` - начало UDP-диапазона WebRTC-медиа. Рекомендуемое значение: начало выделенного диапазона, например `50000`.
- `LIVEKIT_RTC_UDP_END` - конец UDP-диапазона WebRTC-медиа. Рекомендуемое значение: конец выделенного диапазона, например `60000`; диапазон должен соответствовать расчету одновременных медиа-сессий.
- `CALL_MAX_PARTICIPANTS` - максимальное число участников одного звонка. Рекомендуемое начальное значение: `25` для обычных корпоративных звонков; большее значение требует отдельного расчета LiveKit.
- `CALL_MAX_PARALLEL_PER_USER` - максимальное число параллельных звонков на пользователя. Рекомендуемое значение: `1`.
- `CALL_RINGING_TIMEOUT` - время ожидания ответа на входящий звонок. Рекомендуемое значение: `30s`.
- `TELEPHONY_BRIDGE_SECRET` - секрет взаимодействия Gateway с телефоническим bridge-сервисом. Рекомендуемое значение: случайная строка не менее 32 байт из хранилища секретов.
- `EXTERNAL_CONNECTOR_SECRET` - секрет проверки внешних коннекторов и bridge-сервисов. Рекомендуемое значение: случайная строка не менее 32 байт из хранилища секретов.

Для SIP-сценариев профиль внедрения отдельно фиксирует PBX, публичный контактный адрес, транспорт UDP/TCP, NAT-модель и владельца PBX со стороны заказчика.

7.6 Assistant и ИИ-агенты

- `ASSISTANT_SERVICE_URL` - внутренний адрес Assistant для Gateway. Рекомендуемое значение: внутренний HTTP endpoint сервиса Assistant в прикладной зоне.
- `ASSISTANT_PUBLIC_URL` - публичный адрес Искры, используемый Assistant в ссылках и OAuth-сценариях. Рекомендуемое значение: тот же внешний HTTPS URL, что `PUBLIC_APP_URL`, если профиль внедрения не выделяет отдельный домен.
- `ASSISTANT_SERVICE_TIMEOUT` - максимальное время ожидания ответа Assistant со стороны Gateway. Рекомендуемое начальное значение: `30s`.
- `BOT_RUNTIME_BASE_URL` - внутренний адрес Gateway, по которому Assistant арендует обработку ИИ-агентов. Рекомендуемое значение: внутренний HTTP endpoint Gateway, доступный из контейнера Assistant.
- `ASSISTANT_SUMMARY_OPENAI_*` - параметры LLM-поставщика для сводок диалогов. Рекомендуемое значение: отдельный профиль модели для кратких сводок с лимитами поставщика, согласованными с нагрузкой.

- `ASSISTANT_AGENT_OPENAI_*` - базовые параметры LLM-поставщика для ИИ-агентов. Рекомендуемое значение: основной профиль модели для типовых агентских задач.
- `ASSISTANT_AGENT_MEDIUM_OPENAI_*` - параметры LLM-поставщика для задач средней сложности. Рекомендуемое значение: отдельный профиль модели с большим контекстом или лимитом ответа, если такие задачи включены.
- `ASSISTANT_AGENT_LARGE_OPENAI_*` - параметры LLM-поставщика для длительных или сложных задач. Рекомендуемое значение: отдельный профиль модели для длительных задач; включается только при наличии бюджета и лимитов поставщика.
- `ASSISTANT_AGENT_WEB_SEARCH_URL` - адрес поискового инструмента, доступного ИИ-агентам. Рекомендуемое значение: внутренний HTTPS/HTTP endpoint SearXNG или корпоративного поискового шлюза, доступный только Assistant.
- `ASSISTANT_AGENT_GOOGLE_OAUTH_*` - OAuth-параметры Google-интеграций Assistant. Рекомендуемое значение: промышленный OAuth client заказчика с redirect URL из профиля внедрения.
- `ASSISTANT_AGENT_YANDEX_*` - OAuth, календарные и поисковые параметры Yandex-интеграций Assistant. Рекомендуемое значение: промышленное приложение Yandex заказчика с redirect URL из профиля внедрения.
- `ASSISTANT_AGENT_BITRIX_*` - OAuth-параметры Bitrix-интеграции Assistant. Рекомендуемое значение: промышленное приложение Bitrix заказчика или портала, подключаемого к Искре.
- `ASSISTANT_AGENT_HEADHUNTER_*` - OAuth-параметры HeadHunter-интеграции Assistant. Рекомендуемое значение: промышленное приложение HeadHunter заказчика.
- `ASSISTANT_AGENT_IDA_KB_ENDPOINT` - точка доступа базы знаний Ида, если она подключена к агентам. Рекомендуемое значение: внутренний HTTPS endpoint базы знаний; поле не заполняется, если база знаний Ида не используется.

В контейнерной установке `BOT_RUNTIME_BASE_URL` должен указывать на внутренний адрес Gateway, доступный из Assistant. Значения `127.0.0.1` и `localhost` допустимы только для локальной разработки, потому что внутри контейнера они указывают на сам контейнер Assistant.

7.7 Push и мобильные обновления

- `FCM_PROJECT_ID` - идентификатор Firebase-проекта для Android push-уведомлений. Рекомендуемое значение: production Firebase project ID мобильного приложения заказчика.
- `FCM_SERVICE_ACCOUNT_PATH` - путь к файлу служебной учетной записи FCM внутри контейнера. Рекомендуемое значение: путь к смонтированному секрету, если используется файловая передача.
- `FCM_SERVICE_ACCOUNT_JSON_BASE64` - защищенный вариант передачи служебной учетной записи FCM без файла. Рекомендуемое значение: использовать вместо `FCM_SERVICE_ACCOUNT_PATH`, если платформа секретов безопасно передает переменные среды.
- `APNS_TEAM_ID` - идентификатор Apple Developer Team для iOS push-уведомлений. Рекомендуемое значение: Team ID production Apple Developer аккаунта заказчика.

- `APNS_KEY_ID` - идентификатор ключа APNS. Рекомендуемое значение: Key ID production APNS ключа.
- `APNS_AUTH_KEY_PATH` - путь к файлу закрытого ключа APNS внутри контейнера. Рекомендуемое значение: путь к смонтированному секрету, если используется файловая передача.
- `APNS_AUTH_KEY_BASE64` - защищенный вариант передачи закрытого ключа APNS без файла. Рекомендуемое значение: использовать вместо `APNS_AUTH_KEY_PATH`, если платформа секретов безопасно передает переменные среды.
- `APNS_BUNDLE_ID` - bundle identifier iOS-приложения Искры. Рекомендуемое значение: production bundle ID, совпадающий с выпущенной iOS-сборкой.
- `APNS_USE_SANDBOX` - переключает APNS sandbox для отладочных iOS-сборок. Рекомендуемое значение для промышленного контура: `false`.
- `OTA_UPDATES_ROOT` - каталог OTA-файлов, доступный Gateway. Рекомендуемое значение: смонтированный read-only каталог или объектное хранилище, отделенное от пользовательских загрузок.
- `OTA_PUBLIC_BASE_URL` - публичный HTTPS-адрес, через который мобильные клиенты получают OTA. Рекомендуемое значение: HTTPS URL в публичном домене Искры, совпадающий с опубликованными правилами `/v1/updates/...`.

Для TestFlight/App Store сборок используется production APNS. Sandbox APNS применяется только для debug/development-signed iOS установок.

7.8 Event-organizer, billing и внешние обратные вызовы

- `EVENT_ORGANIZER_SERVICE_URL` - внутренний адрес Event-organizer для Gateway. Рекомендуемое значение: внутренний HTTP endpoint Event-organizer в прикладной зоне.
- `EVENT_ORGANIZER_SERVICE_TIMEOUT` - максимальное время ожидания ответа Event-organizer. Рекомендуемое начальное значение: `10s`.
- `EVENT_ORGANIZER_CONTROL_TOKEN` - внутренний токен управления сценариями Event-organizer. Рекомендуемое значение: случайная строка не менее 32 байт из хранилища секретов.
- `EVENT_ORGANIZER_WEBHOOK_SECRET` - секрет проверки обратных вызовов Event-organizer. Рекомендуемое значение: случайная строка не менее 32 байт из хранилища секретов.
- `ROBOKASSA_MERCHANT_LOGIN` - идентификатор магазина Robokassa. Рекомендуемое значение: production merchant login из личного кабинета Robokassa.
- `ROBOKASSA_PASSWORD_1` - первый пароль Robokassa для формирования платежных подписей. Рекомендуемое значение: production пароль из хранилища секретов.
- `ROBOKASSA_PASSWORD_2` - второй пароль Robokassa для проверки уведомлений о платеже. Рекомендуемое значение: production пароль из хранилища секретов, отличный от `ROBOKASSA_PASSWORD_1`.
- `ROBOKASSA_HASH_ALGO` - алгоритм подписи Robokassa, согласованный с настройками магазина. Рекомендуемое значение: алгоритм из production-настроек Robokassa; обычно `SHA256`, если магазин настроен на него.

- `ROBOKASSA_TEST_MODE` - включает тестовый режим Robokassa. Рекомендуемое значение для промышленного приема платежей: `false`; тестовый контур использует `true`.
- `ROBOKASSA_BASE_URL` - адрес платежной страницы Robokassa. Рекомендуемое значение: production URL Robokassa для приема платежей.
- `BILLING_RENEWAL_INTERVAL` - периодичность обработки продлений подписок и платежных состояний. Рекомендуемое начальное значение: `1h`.
- `BILLING_ADMIN_INTERNAL_TOKEN` - внутренний токен административных операций billing-модуля. Рекомендуемое значение: случайная строка не менее 32 байт из хранилища секретов.

8. Секреты и сертификаты

Секреты Искры должны храниться в утвержденном хранилище секретов заказчика или в защищенном механизме развертывания. Запрещено хранить секреты в репозитории, пользовательских журналах, клиентских сборках Web/Mobile/Desktop и диагностических пакетах без маскирования.

Критичные классы секретов:

- `JWT_SECRET`;
- учетные данные PostgreSQL, Redis, NATS и объектного хранилища;
- `LIVEKIT_API_KEY` и `LIVEKIT_API_SECRET`;
- `AUTHENTICATOR_SECRET_ENCRYPTION_KEY`;
- `USER_PHONE_ENCRYPTION_KEY`;
- SMS/OTP учетные данные;
- служебная учетная запись FCM;
- ключ авторизации APNS;
- ключи API LLM-поставщиков;
- клиентские секреты OAuth для календарей, Bitrix и HeadHunter;
- `EXTERNAL_CONNECTOR_SECRET`, `TELEPHONY_BRIDGE_SECRET`;
- секреты billing-модуля;
- закрытые TLS-ключи.

Рекомендуемые значения и правила для секретов:

- симметричные секреты Искры (`JWT_SECRET`, `EXTERNAL_CONNECTOR_SECRET`, `TELEPHONY_BRIDGE_SECRET`, `EVENT_ORGANIZER_CONTROL_TOKEN`, `EVENT_ORGANIZER_WEBHOOK_SECRET`, `BILLING_ADMIN_INTERNAL_TOKEN`) создаются как случайные значения не короче 32 байт;
- ключи шифрования пользовательских телефонов и секретов аутентификатора создаются как случайные значения не короче 32 байт и не меняются без отдельного плана миграции зашифрованных данных;

- пароли, API-ключи и OAuth-секреты поставщиков берутся только из промышленного контура поставщика, не из тестовых кабинетов;
- TLS-ключи выпускаются для утвержденных публичных доменов Искры и хранятся отдельно от прикладных образов;
- плановая ротация интеграционных секретов выполняется не реже одного раза в 12 месяцев или по политике заказчика; внеплановая ротация выполняется немедленно при подозрении на раскрытие;
- после ротации выполняется smoke тестирование входа, сообщений, вложений, звонков, ИИ-агентов и включенных внешних интеграций.

Ротация секретов выполняется через профиль изменения. Для каждого секрета указывается владелец, место хранения, зависимые сервисы, порядок перезапуска и smoke тестирование после ротации. Ротация `JWT_SECRET`, ключей шифрования телефонов и секретов аутентификатора требует отдельного плана, потому что влияет на активные сессии и зашифрованные данные.

9. Порядок промышленной установки

9.1 Подготовка

1. Утвердить профиль внедрения.
2. Подготовить DNS и TLS-сертификаты.
3. Создать БД PostgreSQL и учетные данные приложения.
4. Создать Redis, NATS и объектное хранилище.
5. Создать bucket для вложений, записей, OTA и артефактов.
6. Подготовить LiveKit и media-порты.
7. Разместить секреты в утвержденном хранилище.
8. Подготовить конфигурацию Gateway, Web, Assistant, Webchat, Event-organizer и bridge-сервисов.
9. Подготовить систему мониторинга, сбор журналов и оповещения.

9.2 Первый запуск

1. Запустить PostgreSQL, Redis, NATS и объектное хранилище.
2. Запустить миграции PostgreSQL через сервис `migrate` или утвержденный job миграций.
3. Запустить Gateway.
4. Запустить Web.
5. Запустить Webchat service, если включен Webchat.
6. Запустить Assistant и фоновые обработчики, если включены ИИ-функции.
7. Запустить Event-organizer, если включены сценарии событий.
8. Запустить Telegram/MAX bridge-сервисы, если включены внешние каналы.
9. Запустить LiveKit, egress и SIP-компоненты, если включены звонки и телефония.

10. Запустить Prometheus/Grafana/GlitchTip, если они входят в контур эксплуатации.

Миграции выполняются до запуска новой версии Gateway. Одновременный запуск нескольких миграционных процессов запрещен.

9.3 Приемочные проверки

После первого запуска DevOps-команда выполняет smoke тестирование:

- Gateway отвечает на `/healthz`;
- Gateway отдает `/metrics`;
- `/healthz/livekit` успешен при включенных звонках;
- Web-клиент открывается по публичному HTTPS URL;
- вход по телефону работает через промышленного OTP/SMS-поставщика;
- refresh-сессии работают после обновления access-токена;
- WebSocket `/v1/realtime` подключается и получает события;
- создается диалог;
- отправляется текстовое сообщение;
- загружается и открывается вложение;
- создается звонок, media-соединение устанавливается;
- Assistant отвечает на `/healthz`, если включены ИИ-функции;
- ИИ-агент получает аренду обработки и отвечает в тестовом диалоге;
- точка доступа Webchat и обратный вызов доставки работают, если включен Webchat;
- веб-перехватчик Telegram/MAX проходит проверку секрета, если включены каналы;
- манифест OTA отвечает, если включены мобильные OTA;
- Prometheus собирает метрики;
- прикладные журналы поступают в систему централизованного сбора журналов заказчика, если такая интеграция включена.

10. Проверки здоровья и метрики

Сервис	Точка доступа	Назначение
Gateway	`GET /healthz`	Процесс Gateway отвечает
Gateway	`GET /healthz/livekit`	Gateway может проверить LiveKit по настроенному URL
Gateway	`GET /metrics`	Prometheus-метрики Gateway
Assistant	`GET /healthz`	Процесс Assistant отвечает
Assistant	`GET /metrics`	Метрики Assistant

Event-organizer	`GET /healthz`	Процесс Event-organizer отвечает
Event-organizer	`GET /metrics`	Метрики Event-organizer
Webchat service	`GET /healthz`	Webchat service отвечает
Transcription worker	`GET /healthz`	Обработчик расшифровки отвечает
Sticker normalizer	`GET /healthz`	Обработчик нормализации отвечает
Telegram/MAX bridge	`GET /healthz`	Bridge-сервис отвечает
Web	`GET /healthz`	Web-контейнер отвечает

`/healthz` подтверждает доступность процесса, но не заменяет сквозную проверку входа, сообщений, вложений и звонков. Для промышленной приемки используются оба типа проверки.

Минимальные метрики для мониторинга:

- HTTP RPS, p95/p99 задержка и доля 5xx по Gateway;
- активные WebSocket-соединения и переподключения;
- соединения PostgreSQL, медленные запросы, блокировки и рост таблиц;
- память Redis, вытеснения и задержка;
- задержка потребителей NATS и ожидающие сообщения;
- задержка очереди ИИ, длительность LLM-запросов, ошибочные потоки, доля передач оператору;
- фоновые задания, повторы и dead-letter события;
- комнаты LiveKit, участники, потери пакетов, jitter, bitrate, задания выгрузки;
- ошибки загрузки и выгрузки S3;
- ошибки веб-перехватчиков и доля дублей событий.

11. Эксплуатационные журналы

Прикладные журналы Gateway, Assistant, Webchat service, Event-organizer, bridge-сервисов, LiveKit, фоновых обработчиков и инфраструктуры заказчика используются для диагностики и расследования инцидентов.

В текущей реализации отдельные журналы событий подсистем доступны в PostgreSQL:

- `webchat_audit_events` для событий Webchat;
- `event_audit_log` для действий Event-organizer.

Эти таблицы не являются единым платформенным журналом аудита. Экспорт прикладных журналов в систему централизованного сбора журналов заказчика настраивается как эксплуатационная интеграция. Единый экспорт юридически значимых событий не относится к

базовой текущей реализации и должен быть явно указан в профиле внедрения, если входит в конкретную поставку.

Журналы не должны содержать сырые токены, секреты ботов, ключи LLM, SMS-секреты, закрытые ключи, полные учетные данные поставщика и персональные данные сверх необходимого минимума.

12. Резервное копирование

Резервное копирование должно быть включено до допуска пользователей. Профиль внедрения фиксирует RPO, расписание, срок хранения, место хранения, шифрование резервных копий, ответственного владельца и дату последней проверки восстановления.

Обязательные объекты резервного копирования:

- глобальные объекты PostgreSQL;
- схема и данные основной БД Искры;
- отдельная БД GlitchTip, если включена отчетность об ошибках;
- объектное хранилище: вложения, записи, OTA-файлы, файлы загрузки и артефакты;
- конфигурация развертывания без секретных значений;
- секреты в утвержденном хранилище секретов заказчика;
- правила DNS/TLS/ingress, если они управляются как код;
- артефакты релиза, необходимые для восстановления версии.

Резервное копирование без проверки восстановления не считается подтвержденным механизмом защиты данных. Проверка восстановления проводится в изолированном контуре по расписанию заказчика и фиксируется протоколом.

13. Восстановление

Типовой порядок восстановления:

1. Подготовить чистый контур инфраструктуры.
2. Восстановить глобальные объекты PostgreSQL.
3. Восстановить основную БД Искры.
4. Восстановить БД GlitchTip, если она используется.
5. Восстановить объектное хранилище.
6. Вернуть секреты и конфигурацию.
7. Развернуть тот же релиз Искры, для которого восстановлены данные.
8. Выполнить миграции только если это предусмотрено планом восстановления.
9. Запустить Gateway и зависимые сервисы.
10. Проверить `/healthz`, `/metrics`, вход, сообщения, вложения, звонки, Webchat, ИИ-агента и OTA.

Восстановление БД на более старую версию схемы без совместимого релиза приложения запрещено. Если откат требует отката данных, сначала останавливаются прикладные сервисы, затем восстанавливается БД и только после этого запускается совместимый релиз.

14. Обновление версии

Обновление промышленного контура выполняется как контролируемое изменение.

Перед обновлением:

- зафиксировать версию релиза и список образов;
- проверить совместимость миграций;
- создать резервную копию PostgreSQL;
- убедиться, что резервное копирование завершилось успешно;
- проверить доступность объектного хранилища;
- проверить наличие секретов новой версии;
- согласовать окно изменения;
- подготовить план отката.

Порядок обновления:

1. Загрузить новые образы в реестр или на целевые узлы.
2. Остановить фоновые задачи, если релиз требует паузы.
3. Выполнить миграции через один контролируемый процесс.
4. Обновить Gateway.
5. Обновить Web.
6. Обновить Assistant, Webchat service, Event-organizer и bridge-сервисы.
7. Обновить фоновые обработчики.
8. Проверить точки проверки работоспособности.
9. Выполнить smoke тестирование.
10. Включить пользовательский трафик или снять режим обслуживания.

После обновления:

- проверить вход по OTP;
- проверить refresh-сессии;
- проверить WebSocket `/v1/realtime`;
- проверить диалоги, сообщения и вложения;
- проверить LiveKit call config и создание звонка;
- проверить включенные внешние интеграции;
- проверить ИИ-агента и аренду обработки;

- проверить OTA-точки доступа, если они используются;
- проверить метрики и журналы.

15. Откат версии

Откат приложения без отката данных возможен только если новая версия не выполнила несовместимые миграции и старая версия совместима с текущей схемой БД.

Откат с откатом данных выполняется только по утвержденному плану:

1. Остановить пользовательский трафик.
2. Остановить Gateway, Assistant, Webchat, Event-organizer, bridge-сервисы и фоновые обработчики.
3. Восстановить PostgreSQL из резервной копии.
4. Восстановить объектное хранилище, если релиз изменял формат или размещение объектов.
5. Развернуть совместимый релиз приложения.
6. Запустить сервисы.
7. Выполнить smoke тестирование.
8. Зафиксировать факт восстановления и потерянное окно данных относительно RPO.

Нельзя откатывать только контейнеры, если миграции уже изменили схему или данные несовместимо с предыдущей версией.

16. Горизонтальное масштабирование

Gateway, Web, Webchat service, Assistant, Event-organizer, bridge-сервисы и фоновые обработчики масштабируются добавлением экземпляров при общем PostgreSQL, Redis, NATS и объектном хранилище.

Gateway/API. Метрики масштабирования: RPS, p95/p99 задержка, доля 5xx, активные WebSocket-соединения. Увеличиваются экземпляры Gateway, пул соединений PostgreSQL и ресурсы NATS/Redis.

Web. Метрики масштабирования: HTTP RPS и задержка. Увеличиваются экземпляры Web или емкость CDN/ingress.

Webchat. Метрики масштабирования: внешние сессии, сообщения и задержка обратных вызовов. Увеличиваются экземпляры Webchat service и Gateway.

Assistant. Метрики масштабирования: очередь агентов, задержка LLM и ошибочные потоки. Увеличиваются экземпляры Assistant и лимиты LLM-поставщика.

Bridge-сервисы. Метрики масштабирования: частота веб-перехватчиков, повторы и дубли. Увеличиваются экземпляры bridge-сервисов и лимиты поставщиков.

PostgreSQL. Метрики масштабирования: соединения, блокировки, медленные запросы и рост таблиц. Увеличиваются пул соединений, индексы, ресурсы БД и реплики чтения для безопасных read-сценариев.

NATS. Метрики масштабирования: задержка потребителей и ожидающие сообщения. Увеличиваются ресурсы NATS и настройки JetStream.

Redis. Метрики масштабирования: память, вытеснения и задержка. Увеличиваются ресурсы Redis и корректируются TTL-политики.

LiveKit. Метрики масштабирования: комнаты, участники, битрейт и потери пакетов. Увеличиваются узлы LiveKit и пропускная способность сети.

Объектное хранилище. Метрики масштабирования: пропускная способность, ошибки и дневной прирост. Увеличиваются емкость, политики жизненного цикла и пропускная способность.

При масштабировании Gateway контролируется суммарное число соединений PostgreSQL. Увеличение количества экземпляров без настройки `DATABASE_MAX_OPEN_CONNS` и пула БД может ухудшить стабильность.

17. Мобильные OTA и артефакты настольных приложений

Мобильные OTA-артефакты публикуются через собственный контур Expo Updates и раздаются Gateway:

- `OTA_UPDATES_ROOT` указывает на локальный или общий каталог OTA-файлов;
- `OTA_PUBLIC_BASE_URL` определяет публичную базу URL;
- `GET /v1/updates/manifest` отдает манифест;
- `GET /v1/updates/files/...` отдает файлы OTA.

OTA применяется только к установленной сборке, в которой уже присутствует `expo-updates`. Изменения нативного кода требуют новой APK/IPA сборки.

Артефакты настольных приложений публикуются как установщики и ленты обновлений. Профиль внедрения фиксирует, какие пользовательские псевдонимы загрузки публикуются и кто отвечает за обновление лент настольных приложений.

18. Примеры команд и файлов развертывания

Этот раздел содержит справочные команды и ссылки на эталонные заготовки файлов. Они не заменяют профиль внедрения заказчика и не являются единственным поддерживаемым способом установки. Их назначение - показать проверяемую последовательность промышленной выкладки: один процесс миграций, управляемый запуск сервисов, проверка точек работоспособности, резервное копирование и контролируемый откат.

Примеры поставляются в каталоге `docs/deployment-examples`:

- `env.production.example` - каркас промышленного профиля переменных;
- `docker-compose.production.example.yml` - эталонная заготовка запуска прикладного слоя через Docker Compose;
- `kubernetes.example.yml` - эталонная заготовка Kubernetes-манифестов прикладного слоя Искры;
- `backup-restore.sh` - пример команд резервного копирования и восстановления;
- `rollout.sh` - пример последовательности выкладки через Docker Compose.

Перед применением примеров инженер заказчика заменяет все значения вида `<...>`, подключает утвержденное хранилище секретов, задает реальные имена образов и адаптирует публикацию HTTPS/WSS под сетевой контур заказчика.

18.1 Подготовка профиля переменных

Базовый профиль создается из примера:

```
cp docs/deployment-examples/env.production.example .env.production
```

В `.env.production` задаются адреса PostgreSQL, Redis, NATS, объектного хранилища, LiveKit, публичные URL, OTP/SMS-поставщик и переменные ИИ-агентов. Секретные значения должны подставляться из хранилища секретов заказчика или из защищенного механизма оркестратора.

Минимальная проверка профиля перед запуском:

```
test -n "$DATABASE_URL"
test -n "$REDIS_URL"
test -n "$NATS_URL"
test -n "$JWT_SECRET"
test -n "$LIVEKIT_API_KEY"
test -n "$LIVEKIT_API_SECRET"
```

18.2 Миграции PostgreSQL

Миграции выполняются одним процессом `migrate` до запуска новой версии Gateway:

```
docker compose --env-file .env.production \
  -f docs/deployment-examples/docker-compose.production.example.yml \
  run --rm migrate
```

В Kubernetes миграции запускаются отдельным `Job`:

```
kubectl -n iskra apply -f docs/deployment-examples/kubernetes.example.yml
kubectl -n iskra wait --for=condition=complete job/iskra-migrate --timeout=300s
kubectl -n iskra logs job/iskra-migrate
```

Если миграция завершилась ошибкой, запуск новой версии Gateway не выполняется. Сначала анализируются журналы `migrate`, состояние схемы PostgreSQL и доступность `DATABASE_URL`.

18.3 Запуск через Docker Compose

Примерный порядок запуска прикладного слоя:

```
export ISKRA_GATEWAY_IMAGE=<registry>/iskra-gateway:<release-tag>
export ISKRA_WEB_IMAGE=<registry>/iskra-web:<release-tag>
export ISKRA_TRANSCRIPTION_WORKER_IMAGE=<registry>/iskra-transcription-
worker:<release-tag>
export ISKRA_STICKER_NORMALIZER_IMAGE=<registry>/iskra-sticker-normalizer:<release-
tag>


docker compose --env-file .env.production \
  -f docs/deployment-examples/docker-compose.production.example.yml \
  pull
```

```
docker compose --env-file .env.production \
  -f docs/deployment-examples/docker-compose.production.example.yml \
  up -d gateway web assistant event-organizer webchat transcription-worker sticker-normalizer
```

После запуска проверяется состояние контейнеров:

```
docker compose --env-file .env.production \
  -f docs/deployment-examples/docker-compose.production.example.yml \
  ps
```

18.4 Запуск в Kubernetes

Пример `docs/deployment-examples/kubernetes.example.yml` показывает базовую форму для прикладного слоя Искры: `migrate`, Gateway, Web, Assistant, Event-organizer, Webchat, transcription worker и sticker normalizer. Перед применением заказчик задает реальные образы, политику получения образов, секреты, ingress, сетевые политики, ресурсы, правила хранения OTA и дополнительные bridge-сервисы, если они входят в профиль внедрения.

Применение примера:

```
kubectl apply -f docs/deployment-examples/kubernetes.example.yml
kubectl -n iskra rollout status deployment/gateway --timeout=300s
kubectl -n iskra rollout status deployment/assistant --timeout=300s
kubectl -n iskra rollout status deployment/web --timeout=300s
kubectl -n iskra rollout status deployment/event-organizer --timeout=300s
kubectl -n iskra rollout status deployment/webchat --timeout=300s
```

Проверка экземпляров:

```
kubectl -n iskra get pods
kubectl -n iskra get svc
kubectl -n iskra logs deployment/gateway --tail=100
```

Масштабирование Gateway выполняется изменением числа экземпляров или политикой HPA:

```
kubectl -n iskra scale deployment/gateway --replicas=4
kubectl -n iskra get hpa
```

При увеличении экземпляров Gateway контролируется суммарный пул соединений PostgreSQL: `replicas * DATABASE_MAX_OPEN_CONNS` не должен превышать лимит, согласованный для БД.

18.5 Проверки после запуска

Базовые проверки HTTP-точек:

```
curl -fsS https://<im.example.ru>/healthz
curl -fsS https://<im.example.ru>/metrics
curl -fsS https://<im.example.ru>/v1/updates/manifest
```

Проверка Gateway изнутри Kubernetes:

```
kubectl -n iskra port-forward svc/gateway 18080:18080
curl -fsS http://127.0.0.1:18080/healthz
curl -fsS http://127.0.0.1:18080/healthz/livekit
```

Проверка WebSocket выполняется инструментом заказчика, поддерживающим WSS. Точка доступа Искры: `wss://<im.example.ru>/v1/realtime`.

Сквозная приемка после технических проверок включает вход по OTP, создание диалога, отправку сообщения, загрузку вложения, звонок, ответ ИИ-агента, Webchat и внешние каналы, если эти модули включены.

18.6 Резервное копирование и восстановление

Пример команд находится в `docs/deployment-examples/backup-restore.sh`. Перед использованием задаются переменные:

```
export DATABASE_URL=postgres://<user>:<password>@<postgres-host>:5432/<database>?sslmode=require
export BACKUP_DIR=/var/backups/iskra
export OBJECT_STORAGE_BUCKET=<bucket-name>
export S3_BACKUP_URI=s3://<backup-bucket>/iskra/prod

docs/deployment-examples/backup-restore.sh backup
```

Восстановление PostgreSQL выполняется только в остановленном или изолированном контуре:

```
export RESTORE_DUMP=/var/backups/iskra/iskra-<timestamp>.dump
docs/deployment-examples/backup-restore.sh restore-postgres
```

Восстановление объектного хранилища:

```
export RESTORE_OBJECTS_URI=s3://<backup-bucket>/iskra/prod/objects/<timestamp>/
docs/deployment-examples/backup-restore.sh restore-objects
```

18.7 Контролируемая выкладка и откат

Пример последовательности выкладки для Docker Compose:

```
export PUBLIC_APP_URL=https://<im.example.ru>
export ISKRA_GATEWAY_IMAGE=<registry>/iskra-gateway:<release-tag>
export ISKRA_WEB_IMAGE=<registry>/iskra-web:<release-tag>
export ISKRA_TRANSCRIPTION_WORKER_IMAGE=<registry>/iskra-transcription-worker:<release-tag>
export ISKRA_STICKER_NORMALIZER_IMAGE=<registry>/iskra-sticker-normalizer:<release-tag>

docs/deployment-examples/rollout.sh
```

Откат без восстановления данных допускается только при совместимости старого релиза с текущей схемой PostgreSQL:

```
export ISKRA_GATEWAY_IMAGE=<registry>/iskra-gateway:<previous-release-tag>
export ISKRA_WEB_IMAGE=<registry>/iskra-web:<previous-release-tag>

docker compose --env-file .env.production \
  -f docs/deployment-examples/docker-compose.production.example.yml \
  up -d gateway web assistant event-organizer webchat transcription-worker sticker-normalizer
```

Если новая версия выполнила несовместимые миграции, откат выполняется по разделу 15 с восстановлением PostgreSQL из резервной копии.

19. Типовые неисправности

Gateway не стартует

Проверить:

- доступность `DATABASE_URL`;
- наличие `JWT_SECRET`;
- применены ли миграции;
- доступность Redis/NATS/S3/LiveKit по внутренним адресам;
- права чтения секретов;
- свободен ли порт сервиса или целевой порт в оркестраторе.

Пользователь не получает OTP

Проверить:

- `OTP_PROVIDER`, `OTP_MODE`, `SMS_PROVIDER`;
- учетные данные SMS/OTP поставщика;
- `OTP_TTL`, `OTP_RESEND_COOLDOWN`, `OTP_MAX_ATTEMPTS`;
- ограничения частоты запросов и лимиты поставщика;
- прикладные журналы Gateway по идентификатору запроса;
- сетевой исходящий доступ к поставщику.

Сообщения сохраняются, но события реального времени не обновляются

Проверить:

- WebSocket `/v1/realtime`;
- `NATS_URL`;
- работоспособность Redis/NATS;
- токен доступа клиента;
- ошибки распространения событий в журналах Gateway;
- балансировщик и поддержку переключения соединения на WebSocket.

Не загружаются вложения

Проверить:

- `OBJECT_STORAGE_ENDPOINT`;
- корзину объектного хранилища и права ключа доступа;
- `MESSAGE_ATTACHMENT_MAX_BYTES`;
- `MESSAGE_MULTIPART_MAX_BYTES`;

- доступность S3 из Gateway;
- задания очистки промежуточных загрузок;
- лимиты входящего прокси/nginx на размер тела запроса.

Не работают звонки

Проверить:

- `LIVEKIT_URL`, `LIVEKIT_PUBLIC_URL`, `LIVEKIT_NODE_IP`;
- `/healthz/livekit`;
- `LIVEKIT_API_KEY`, `LIVEKIT_API_SECRET`;
- UDP-диапазон LiveKit;
- межсетевой экран и NAT;
- права микрофона/камеры на клиенте;
- потери пакетов и джиттер в метриках LiveKit.

ИИ-агент не отвечает

Проверить:

- `ASSISTANT_SERVICE_URL`;
- `BOT_RUNTIME_BASE_URL`;
- `/healthz` Assistant;
- `ASSISTANT_AGENT_OPENAI_*`;
- наличие учетных данных среды выполнения ботов;
- точки доступа аренды обработки `/v1/bot-runtime/leases`;
- WebSocket `/v1/bot-runtime/ws`;
- ошибки запросов инструментов и лимиты LLM-поставщика.

20. Чек-лист допуска в эксплуатацию

- Профиль внедрения заполнен и утвержден.
- DNS и TLS настроены.
- PostgreSQL, Redis, NATS, объектное хранилище и LiveKit доступны из прикладной зоны.
- Секреты размещены в утвержденном хранилище.
- Миграции PostgreSQL выполнены одним контролируемым процессом.
- Gateway, Web, Webchat, Assistant, Event-organizer и bridge-сервисы запущены согласно включенным модулям.
- `/healthz` и `/metrics` отвечают для включенных сервисов.
- Вход по OTP работает через промышленного поставщика.

- события реального времени через WebSocket работают.
- Сообщения, вложения и звонки проходят сквозную проверку.
- ИИ-агент отвечает в тестовом диалоге, если модуль включен.
- Webchat и внешние каналы проходят smoke тестирование, если включены.
- Резервное копирование PostgreSQL и объектного хранилища настроено.
- Восстановление проверено в изолированном контуре или назначена дата первой проверки до промышленного допуска.
- Метрики собираются.
- Журналы поступают в эксплуатационный контур заказчика, если интеграция включена.
- Оповещения настроены для критичных отказов.
- Секреты не попали в репозиторий, клиентские сборки и пользовательские журналы.

Приложение A. Справочно: локальная среда разработки

Локальная разработка не является промышленной моделью установки. Она используется разработчиками и инженерами поддержки для воспроизведения сценариев.

Базовый запуск:

```
make dev-up
make run-gateway
make run-assistant
make run-transcription-worker
```

Проверка локальных контейнеров:

```
make dev-check
```

Остановка локальной инфраструктуры:

```
make dev-down
```

Локальный compose поднимает PostgreSQL, Redis, NATS, SearXNG, MinIO, LiveKit, LiveKit egress, LiveKit SIP, Kamailio, telephony-edge-dev, transcription-worker, Assistant, Event-organizer, Gateway и Webchat. Если Gateway запускается на хосте через `make run-gateway`, он использует локальные адреса инфраструктуры. Если Gateway запускается контейнером, используются внутренние docker-имена.

Приложение B. Справочно: предпромышленный стенд поставщика

Предпромышленный стенд поставщика в репозитории описан в `docker-compose.preprod.yml` и внутренних документах выкладки. Он используется как справочный материал по составу сервисов и smoke тестированию. Промышленная установка заказчика выполняется по профилю внедрения, разделам 9-18 и утвержденным файлам развертывания заказчика.

Текущий предпромышленный compose включает:

- PostgreSQL, Redis, NATS, MinIO;
- LiveKit и egress;

- Gateway, Web, Webchat service;
- Assistant, Event-organizer, transcription-worker, sticker-normalizer;
- Telegram bridge и MAX bridge;
- GlitchTip;
- Prometheus, Grafana, экспортеры;
- задание миграции.

Внутренний регламент выкладки предпромышленного стенда создает резервную копию PostgreSQL перед миграциями, запускает миграции, обновляет сервисы, выполняет проверки работоспособности и проверяет опубликованные артефакты. Для промышленного заказчика эти шаги переводятся в регламент обновления из разделов 14 и 15.

21. Ссылки на связанные документы

- Архитектурная спецификация Искры: `docs/iskra-architecture-manual.ru.md`;
- руководство разработчика API для ИИ-агентов: `docs/iskra-developer-api-manual.ru.md`;
- примеры промышленного развертывания: `docs/deployment-examples/`;
- регламент мобильных OTA: `docs/ota-preprod.md`;
- регламент обновлений настольных приложений: `docs/desktop-update-runbook.md`;
- промышленный регламент Webchat: `docs/webchat-production-runbook.md`.