SPEAKNODE

Руководство по установке

Версия 1.0

Настоящая документация может быть использована только для поддержки работоспособности продуктов, поставленных платформой Speaknode.

Все примеры конфигураций, реквизитов и других данных в документе вымышлены и не относятся к реальным сущностям. Любое совпадение случайно.

Все встречающиеся в тексте товарные знаки и зарегистрированные товарные знаки являются собственностью их владельцев и используются исключительно для идентификации программного обеспечения или компаний.

Данная документация может не отражать некоторых модификаций программного обеспечения. Если вы заметили ошибки или опечатки, сообщите об этом по контактным данным ниже.

-

help@mail.speaknode.com · speaknode.com

Содержание

Руководство по установке

Развертывание

Как установить, настроить и запустить Speaknode.

- Быстрый старт — `docker compose up` из публичного репозитория
- Требования — CPU, RAM, диск по сервисам и калькулятор ресурсов
- Установка — способы поставки, варианты развёртывания, как получить токен registry
- Конфигурация — переменные окружения и профили

Конфигурация

Вся конфигурация в одном `.env` файле рядом с `docker-compose.yml`. Копируем пример и заполняем поля под своё развёртывание:

```
cp .env.example .env
```

Эталонный `.env.example` опубликован в публичном репо quickstart рядом с compose-файлом.

Секции

Файл сгруппирован по сервисам. Каждый сервис читает **только** свой префикс — например, всё для asp-net бэкенда это `BACKEND_*`, для python-воркера это `WORKER_*` и т.п. Общие значения (хост Postgres, брокеры Kafka, пароль Redis) определены один раз в **базовых** секциях в начале файла и подставляются через `${VAR}` в секциях сервисов ниже.

Секция	Префикс	Что настраивает
Host ports	`*_EXTERNAL_PORT`	Хост-порты, на которых docker выставляет сервисы. Меняй если конфликтует.
Service URLs	`*_INTERNAL_URL`, `*_PUBLIC_URL`	Внутренние DNS-имена vs URL, видимые из браузера.
Postgres	`POSTGRES_*`	Креды БД. Один инстанс Postgres хостит несколько БД.
Redis	`REDIS_*`	Общий пароль. Сервисы используют разные logical DB.
Kafka	`KAFKA_*`	SASL-креды + bootstrap servers.
MinIO	`MINIO_*`	S3-совместимое хранилище для записей и счетов.
Keycloak	`KEYCLOAK_*`	Admin-пользователь, SMTP, OAuth, подтверждение email.
LiveKit	`LIVEKIT_*`	API-ключи, SIP-транк, WebRTC NAT-конфиг.
Backend	`BACKEND_*`	Настройки .NET API, ключи LLM, коды биллинговых планов.
Worker	`WORKER_*`	Python voice agent — ключи LLM/STT/TTS провайдеров.
Frontend	`FRONTEND_CLIENT_*`, `FRONTEND_WIDGET_*`	Runtime-конфиг для Angular-бандлов.
Lago	`GETLAGO_*`	Биллинговый движок: БД, Redis, креды платёжного провайдера, webhook URL.
Webhooks	`WEBHOOKS_*`	Приёмник входящих вебхуков (Lago, LiveKit).
Logging (opt-in)	`LOGGING_*`	Креды Fluent Bit / Benthos / Elasticsearch.

Критичные настройки

Часть значений **обязательно** заполнить перед первым запуском — дефолты не подойдут:

Переменная	Зачем
API-ключи для LLM / STT / TTS-провайдеров, которые вы включаете	Доступ к моделям для агентов (конкретные имена переменных зависят от вендора — смотрите `.env.example`)
`GETLAGO_MIGRATE_STRIPE_SECRET_KEY`	Биллинг (только если подключён платёжный провайдер)
`KEYCLOAK_SMTP_*`	Доставка email при регистрации / сбросе пароля
`LIVEKIT_API_KEY` / `LIVEKIT_API_SECRET`	Подпись WebRTC — сгенерируйте случайные 32-char секреты
`GETLAGO_RSA_PRIVATE_KEY`	Base64-кодированный RSA для подписи Lago-вебхуков
Encryption keys (`*_ENCRYPTION_*`, `SECRET_KEY_BASE`)	В проде заменить дефолты на случайные строки

В `.env.example` лежат рабочие placeholder'ы для локальной разработки, но **каждый** секрет должен быть заменён для прода.

Профили

Часть сервисов — opt-in и не стартует без профиля:

```
# Стек логирования (Elasticsearch + Kibana + Fluent Bit + Benthos)
docker compose --profile logging up -d

# Аналитика (ClickHouse)
docker compose --profile analytics up -d
```

Масштабирование отдельных сервисов

Чтобы запустить более одного инстанса session-bound сервиса (worker, livekit, bridge) — используйте реплики Docker Swarm или замените Compose на настоящий оркестратор (Kubernetes, Nomad, ECS). Все сервисы stateless, кроме backbone (Postgres, Redis, Kafka, MinIO, Keycloak).

Как подобрать размер под ожидаемую нагрузку — см. Требования.

Установка

Speaknode поставляется как набор Docker-образов в приватном container registry + публичный бандл `docker-compose.yml` + `.env.example`, который связывает все сервисы.

Поддерживаются три варианта установки:

1. Quickstart (один хост, Docker Compose)

Самый быстрый способ попробовать Speaknode — один `docker compose up` поднимает весь стек на машине 4 vCPU / 8 GB. Пошаговый гайд — в Быстром старте.

Подходит для: демо, локальной разработки, пилотов до ~10 одновременных сессий.

2. Compose на прод-VM

Тот же compose-файл что в quickstart, развёрнутый на правильно подобранной Linux-VM (см. Требования). Используется тот же публичный `docker-compose.yml`, но с секретами и увеличенными лимитами ресурсов.

Подходит для: небольшого/среднего прода до ~50 одновременных сессий, single-tenant.

3. Managed / Railway / Kubernetes

Каждый сервис доступен как отдельный контейнер — можно развернуть в Railway, ECS, Kubernetes или любом оркестраторе. В проде мы крутим Speaknode на Railway с managed Postgres, Redis и Kafka.

Напишите нам, если нужны Helm-чарты или Terraform-модули — поможем настроить под ваше окружение.

Что нужно перед стартом

- Deployment-токен для container registry (см. ниже).
- Хост под Требования.
- Аккаунты во внешних сервисах, которые планируете включить: LLM-провайдер, STT-провайдер, TTS-провайдер, платёжный провайдер из списка поддерживаемых биллинговым движком (для платных тарифов), Mailgun или аналог (email).

Получение deployment-токена

Образы лежат в `registry.gitlab.com/nodevoice/*`. Они не публичные — каждому клиенту выдаётся свой read-only токен.

Как запросить: откройте issue в репо quickstart или напишите нам — мы выдадим токен.

Залогиньтесь на хосте:

```
docker login registry.gitlab.com -u <имя-токена> -p <секрет-токена>
```

После этого `docker compose pull` скачает все нужные образы.

Быстрый старт

Speaknode поставляется бандлом — `docker-compose.yml` и эталонный `.env.example` — прямо с этого сайта документации. Образы лежат в приватном container registry `registry.nodul.ru` — доступ выдаётся по read-only токену (см. ниже).

1. Скачать бандл

```
mkdir Speaknode && cd Speaknode
curl -O https://test.docs.speaknode.com/cdn/docker-compose.yml
curl -O https://test.docs.speaknode.com/cdn/.env.example
```

- `docker-compose.yml` — полный стек сервисов (включает одноразовый сервис `postgres-init`, который создаёт дополнительные БД/пользователей после того как Postgres становится healthy)
- `.env.example` — эталонные переменные окружения

2. Запросить токен для реестра

Образы из `registry.gitlab.com/nodevoice/*` требуют read-only токен. Токены выдаются **каждому клиенту по запросу** — напишите нам на `help@mail.speaknode.com`, и мы пришлём токен.

После этого залогиньтесь на хосте — docker сам спросит логин и пароль:

```
docker login registry.gitlab.com/nodevoice
```

3. Настроить

```
cp .env.example .env
```

Базовый `.env.example` уже содержит всё необходимое для локального запуска — можно оставить как есть и поднимать стек. Ключи LLM/STT/TTS-провайдеров, SMTP для Keycloak, биллинг, телефония и прочие интеграции — опциональны и нужны только для соответствующего функционала.

Полный референс переменных — в Конфигурации.

4. Запустить стек

Сначала подтяните актуальные образы — чтобы не работать с устаревшим `:latest`, закешированным при прошлом `docker login`:

```
docker compose pull
docker compose up -d
```

Первый запуск занимает несколько минут — Postgres инициализируется, Keycloak импортирует realm, Lago накатывает миграции и сидит биллинг-организацию.

5. Дождаться `angular-client`

Проверить статус сервисов:

```
docker compose ps -a --format "table {{.Service}}\t{{.Status}}\t{{.Ports}}"
```

Как только строка `angular-client` покажет **`Up ... (healthy)`** — приложение готово, открывайте http://localhost:4210.

6. Создать пользователя

На странице входа нажмите **Регистрация** и создайте учётную запись.

В поставке для `docker compose` подтверждение email по умолчанию **отключено** (`KEYCLOAK_VERIFY_EMAIL=false`), регистрация моментальная.

Включить подтверждение email

Если хотите, чтобы пользователи подтверждали почту перед первым входом, в `.env`:

1. Включите флаг:

```
KEYCLOAK_VERIFY_EMAIL=true
```

2. Заполните SMTP-реквизиты Keycloak (с них уходят письма верификации и сброса пароля):

```
KEYCLOAK_SMTP_HOST=smtp.example.com
KEYCLOAK_SMTP_PORT=465
KEYCLOAK_SMTP_FROM=no-reply@example.com
KEYCLOAK_SMTP_USER=<smtp-логин>
KEYCLOAK_SMTP_PASSWORD=<smtp-пароль>
KEYCLOAK_SMTP_SSL=true       # 465
KEYCLOAK_SMTP_STARTTLS=false # переключите на true для порта 587
```

3. Перезапустите Keycloak, чтобы применить:

```
docker compose up -d --force-recreate keycloak
```

Другие точки входа

Когда стек полностью поднялся:

Сервис	URL
Speaknode app	http://localhost:4210
Keycloak admin	http://localhost:8091
Backend Swagger	http://localhost:8040/swagger
Lago billing UI	http://localhost:4203
MinIO console	http://localhost:9001
Kafka UI	http://localhost:8084

Порты настраиваются в `.env` в секции **HOST PORTS** — меняйте любой, если на вашей машине он уже занят.

Что дальше

- Требования — подобрать хост под ожидаемую нагрузку.
- Конфигурация — полный референс переменных.
- Установка — другие варианты развёртывания (managed, Kubernetes).

Требования

Speaknode — модульная платформа. Каждый сервис работает в своём контейнере, поэтому компоненты можно масштабировать независимо. Перед развёртыванием подберите железо под ожидаемое **число одновременных сессий** — остальное (бэкбон) остаётся примерно постоянным.

Программные зависимости

- Docker 24+ и Docker Compose v2.24+
- 64-битный Linux-хост (Ubuntu 22.04 LTS или аналог). macOS подойдёт для локальной разработки.
- Исходящий интернет (registry, LLM/STT/TTS провайдеры, STUN/TURN).
- Порты, доступные из браузеров пользователей: фронтенд, API бэкенда, Keycloak, LiveKit WebRTC (TCP 7881, UDP 50000–60000).

Ресурсы по сервисам

В таблице ниже — CPU / RAM / Disk для каждого сервиса при типовых нагрузках по одновременным сессиям. Значения — эмпирические замеры из прода.

Как читать:

- **Сервисы, зависящие от сессий** (`livekit-server` , `python-worker` , `egress` , `websocket-media-bridge`) растут линейно по числу активных звонков. Их и нужно планировать.
- **Backbone-сервисы** (Postgres, Kafka, Keycloak, Redis, MinIO и т.п.) имеют в основном фиксированный футпринт — разворачиваются один раз и переиспользуются между тенантами.
- **Диск** в основном растёт в Postgres (транскрипты, счета), Kafka (retention событий) и MinIO (записи звонков). Планируйте хранилище с учётом retention.

Калькулятор ресурсов

Оцените общий футпринт application-слоя под пиковую нагрузку:

Замечания

- Числа **только для application-слоя**. CPU/RAM/диск под LLM, STT и TTS провайдеров (или self-hosted модели) **оплачиваются отдельно** у вендоров и здесь не учтены. Про расчёт стоимости моделей — в отдельной странице позже.
- Добавьте запас 20–30% сверху на headroom, пики и оверхед ОС.
- Для HA-прода удвойте backbone (реплика Postgres, Kafka с replication factor ≥ 2 и т.п.).

Минимальный сайзинг машины

Нагрузка	Рекомендуемая VM
Демо / dev (1 сессия)	4 vCPU · 8 GB RAM · 50 GB SSD
Small (10 сессий)	6 vCPU · 16 GB RAM · 100 GB SSD
Medium (50 сессий)	12 vCPU · 32 GB RAM · 200 GB SSD
Large (250 сессий)	32 vCPU · 96 GB RAM · 500 GB SSD (multi-node)
XL (500 сессий)	64 vCPU · 192 GB RAM · 1 TB SSD (multi-node)

Для всего выше 50 одновременных сессий рекомендуем вынести `livekit-server` + `python-worker` на отдельные хосты, чтобы медиа и агентская нагрузка не мешали backbone.