Установка Stoat

Stoat (self-hosted чат) через Docker Compose

Развёртывание Stoat — бывший Revolt, self-hosted чат в духе Discord — на сервере, уже настроенном по статье «Первоначальная настройка сервера Ubuntu 24.04».

Основной каталог — `/opt/stoat`.

Каждый блок кода рассчитан на копирование целиком: это либо одна команда, либо цепочка через `&&`.

Предпосылки

Статья продолжает базовую настройку и считает, что на сервере уже есть:

- пользователь `mrleo1nid` с sudo, вход по ключу, root закрыт;
- UFW с открытыми 22, 80, 443;
- iptables в legacy-режиме;
- Caddy на хосте с рабочим `/etc/caddy/Caddyfile`.

Не хватает только Docker — его ставим ниже.

Про compose-файл. Собственный `compose.yaml` тут не пишется. Стек состоит примерно из десяти связанных сервисов (MongoDB, KeyDB, RabbitMQ, LiveKit, файловый сервер, веб-клиент, бэкенд, прокси превью-ссылок), готовый `compose.yml` лежит в репозитории проекта, а скрипт `generate_config.sh` генерирует конфигурацию строго под него. Поэтому «развёртывание через docker compose» здесь = склонировать репозиторий и запустить `docker compose` внутри него.

Ресурсы. Минимум 2 vCPU и 2 ГБ памяти.

Ограничение по голосу. LiveKit требует прямого доступа из интернета к `7881/tcp` и `50000-50100/udp`. Через реверс-прокси этот трафик не проходит — Caddy его не обрабатывает намеренно. Если сервер за NAT, порты нужно пробросить до него отдельно. Без этого текстовый чат работает, голос и видео — нет.

Клиенты. Официальные мобильные приложения в основном не умеют подключаться к своим инстансам. Работает веб-клиент, установленный как PWA он близок к нативному приложению.

---

1. Установка Docker

Базовые пакеты:

```
sudo apt update && sudo apt install -y ca-certificates curl git
```

Ключ репозитория:

```
sudo install -m 0755 -d /etc/apt/keyrings && sudo curl -fsSL
https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc && sudo chmod a+r
/etc/apt/keyrings/docker.asc
```

Репозиторий:

```
echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu $(. /etc/os-release && echo "$VERSION_CODENAME")
stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

Установка:

```
sudo apt update && sudo apt install -y docker-ce docker-ce-cli containerd.io docker-buildx-
plugin docker-compose-plugin
```

Добавляем себя в группу `docker`, чтобы не писать `sudo` перед каждой командой (нужно перелогиниться после):

```
sudo usermod -aG docker $USER
```

Проверка:

```
docker compose version
```

> Docker переопределяет backend iptables при старте. Если базовая настройка переключала систему в legacy-режим **после** установки Docker,

> перезапустите демон: `sudo systemctl restart docker`.

---

2. Домен и порты

A-запись домена должна указывать на сервер:

```
dig +short chat.example.com
```

Далее везде используется `chat.example.com` — замените на свой.

80 и 443 уже открыты базовой настройкой. Добавляем только RTC-порты:

```
sudo ufw allow 7881/tcp && sudo ufw allow 50000:50100/udp
```

```
sudo ufw status verbose
```

---

3. Клонирование репозитория

```
sudo mkdir -p /opt/stoat && sudo chown $USER:$USER /opt/stoat
```

```
git clone https://github.com/stoatchat/self-hosted /opt/stoat
```

В каталоге окажутся `compose.yml`, `Caddyfile`, `generate_config.sh` и `secrets.env.example`.

---

4. Генерация конфигурации

```
cd /opt/stoat && chmod +x ./generate_config.sh && ./generate_config.sh chat.example.com
```

Скрипт задаст вопрос, ставится ли Stoat за другим реверс-прокси. **Отвечайте** `y` — Caddy на хосте у нас уже есть, и два Caddy за 80/443 подраться не должны. При таком ответе встроенный прокси будет слушать порт `8880`.

Скрипт создаст `Revolt.toml` с настройками и `secrets.env` с секретами.

>

> ⚠ **Сразу сохраните `secrets.env` в надёжное место.** Потеря этого файла означает потерю доступа ко всем файлам, загруженным на инстанс.

```
cp /opt/stoat/secrets.env ~/stoat-secrets.env.backup && chmod 600 ~/stoat-secrets.env.backup
```

Секреты хранятся только в `secrets.env` — в `Revolt.toml`, `.env` и `.env.web` лежат домены и публичные настройки клиента. Файл создаётся с правами `644`, закрываем:

```
chmod 600 /opt/stoat/secrets.env
```

> ⚠ **Права остальных файлов не трогайте.** `Revolt.toml` монтируется внутрь контейнеров и читается процессами под другим UID. Если поставить на него `600`, все Rust-сервисы (`api`, `events`, `autumn`, `january`, `pushd`, `crond`, `gifbox`, `voice-ingress`) упадут в панику с `Permission denied (os error 13)` в `crates/core/config` и уйдут в цикл перезапуска. Лечится возвратом `chmod 644`.
>
> С `secrets.env` такой проблемы нет: он не монтируется, а подключается через `env_file` и читается самим docker compose от вашего пользователя.

Ещё скрипт создаёт `compose.override.yml` — там публикация порта встроенного Caddy. Посмотреть:

```
cat /opt/stoat/compose.override.yml
```

Если публикация вида `"8880:80"`, поправьте на `"127.0.0.1:8880:80"`. Прокси стоит на том же хосте, наружу порт светить незачем — тем более что Docker публикует порты в обход правил UFW:

```
nano /opt/stoat/compose.override.yml
```

Правка основной конфигурации (подтверждение email, капча, свой S3, лимиты, пуши):

```
nano /opt/stoat/Revolt.toml
```

---

5. Первый запуск

Сначала в форграунде, чтобы увидеть ошибки. Первый старт долгий — тянется около десяти образов:

```
cd /opt/stoat && docker compose up
```

Если критических ошибок нет — `Ctrl+C` и перезапуск в фоне:

```
cd /opt/stoat && docker compose up -d
```

Что поднялось и в каком состоянии:

```
cd /opt/stoat && docker compose ps
```

Проверка встроенного прокси:

```
curl -I http://127.0.0.1:8880
```

---

6. Подключение к Caddy на хосте

Открываем существующий конфиг и добавляем блок к тому, что уже есть:

```
sudo nano /etc/caddy/Caddyfile
```

```
chat.example.com {
    reverse_proxy 127.0.0.1:8880
}
```

Отдельные блоки для `/ws` и `/livekit` — как в официальном гайде для nginx — здесь **не нужны**: Caddy пробрасывает WebSocket-апгрейды прозрачно и сам проставляет `Host`, `X-Forwarded-For` и `X-Forwarded-Proto`.

```
sudo caddy validate --config /etc/caddy/Caddyfile && sudo systemctl reload caddy
```

```
sudo journalctl -u caddy -f
```

Ждём в логах запись о полученном сертификате, затем открываем `https://chat.example.com`.

Проверить снаружи, что RTC-порт доступен (запускать **не** с этого сервера):

```
nc -vz chat.example.com 7881
```

---

7. Первый аккаунт и закрытие регистрации

Первый зарегистрированный на инстансе аккаунт становится вашим. Регистрируйтесь сразу — по умолчанию она открыта для всех, и оставлять её такой не стоит.

Закрываем регистрацию, добавив в `Revolt.toml`:

```
nano /opt/stoat/Revolt.toml
```

```
[api.registration]
invite_only = true
```

Применяем:

```
cd /opt/stoat && docker compose restart
```

Приглашения создаются вручную в MongoDB. Заходим в шелл:

```
cd /opt/stoat && docker compose exec database mongosh
```

Внутри шелла:

```
use revolt
db.account_invites.insertOne({ _id: "мой_код_приглашения", used: false })
```

> **Два предупреждения.** База называется `revolt`, а не `stoat` — наследие после ребрендинга. И коллекция именно `account_invites`: в README апстрима указана `invites`, но это приглашения в каналы и серверы, а не для регистрации. Подробнее — в следующем разделе.

Дальше удобнее пользоваться скриптами.

Скрипты для управления приглашениями

Ходить каждый раз в `mongosh` руками неудобно. Три небольших обёртки закрывают весь цикл: создать, посмотреть, отозвать.

> ⚠ **Инструкция в README апстрима устарела.** Там предлагается вставлять документ `{ _id: "код" }` в коллекцию `invites`. Это не работает: `invites` хранит приглашения **в каналы и серверы**, а приглашения для **регистрации** живут в отдельной коллекции `account_invites`, которой заведует Authifier — библиотека аутентификации. Регистрация по коду из `invites` отклоняется с ошибкой `InvalidInvite`.

Модель приглашения в Authifier выглядит так:

```
pub struct Invite {
    pub id: String,                  // код, в MongoDB это _id
    pub used: bool,                  // использовано ли
    pub claimed_by: Option<String>,  // ID пользователя, который его использовал
}
```

Поле `used` обязательное, а не опциональное — документ без него не разбирается бэкендом, и код снова получает `InvalidInvite`. Поэтому минимально корректная запись такая:

```
{ _id: "код", used: false }
```

Каталог для скриптов:

```
mkdir -p /opt/stoat/bin
```

Создание приглашения

```
nano /opt/stoat/bin/invite-new.sh
```

```
#!/usr/bin/env bash
set -euo pipefail
cd /opt/stoat
```

```
# код можно передать аргументом, иначе генерируется случайный
CODE="${1:-$(openssl rand -hex 8)}"

if ! [[ "$CODE" =~ ^[A-Za-z0-9_-]{4,64}$ ]]; then
    echo "Недопустимый код. Разрешены A-Z a-z 0-9 _ - длиной от 4 до 64 символов." >&2
    exit 1
fi

docker compose exec -T -e CODE="$CODE" database mongosh revolt --quiet --eval '
  const code = process.env.CODE;
  if (db.account_invites.findOne({ _id: code })) {
      print("Приглашение с таким кодом уже существует: " + code);
      quit(1);
  }
  db.account_invites.insertOne({ _id: code, used: false });
  print("Создано приглашение: " + code);
'
```

Список приглашений

```
nano /opt/stoat/bin/invite-ls.sh
```

```
#!/usr/bin/env bash
set -euo pipefail
cd /opt/stoat

docker compose exec -T database mongosh revolt --quiet --eval '
  const rows = db.account_invites.find().toArray();
  if (rows.length === 0) {
      print("Приглашений нет.");
      quit(0);
  }
  const free = rows.filter(r => !r.used).length;
  print("Всего: " + rows.length + " | свободных: " + free + " | использованных: " +
(rows.length - free));
  print("");
  for (const r of rows) {
      const status = r.used ? "ИСПОЛЬЗОВАНО" : "свободно    ";
      const by = r.claimed_by ? "  <- " + r.claimed_by : "";
      print(status + "  " + r._id + by);
```

```
  }
'
```

Отзыв приглашения

```
nano /opt/stoat/bin/invite-rm.sh
```

```
#!/usr/bin/env bash
set -euo pipefail
cd /opt/stoat

CODE="${1:?Укажите код приглашения: invite-rm.sh КОД}"

docker compose exec -T -e CODE="$CODE" database mongosh revolt --quiet --eval '
  const code = process.env.CODE;
  const doc = db.account_invites.findOne({ _id: code });
  if (!doc) {
      print("Приглашение не найдено: " + code);
      quit(1);
  }
  if (doc.used) {
      print("Внимание: это приглашение уже использовано" + (doc.claimed_by ? " пользователем "
+ doc.claimed_by : "") + ".");
      print("Удаление записи не отключает созданный по нему аккаунт.");
  }
  db.account_invites.deleteOne({ _id: code });
  print("Удалено: " + code);
'
```

Права на запуск

```
chmod +x /opt/stoat/bin/invite-new.sh /opt/stoat/bin/invite-ls.sh /opt/stoat/bin/invite-rm.sh
```

Использование

Создать приглашение со случайным кодом:

```
/opt/stoat/bin/invite-new.sh
```

Создать приглашение с осмысленным кодом:

```
/opt/stoat/bin/invite-new.sh dlya-peti
```

Посмотреть все, с отметкой об использовании:

```
/opt/stoat/bin/invite-ls.sh
```

Отозвать:

```
/opt/stoat/bin/invite-rm.sh dlya-peti
```

Что нужно знать

- **Приглашение одноразовое.** При регистрации бэкенд проставляет `used: true` и записывает ID пользователя в `claimed_by`. Повторно тем же кодом зарегистрироваться нельзя.
- **Отзыв имеет смысл только до использования.** Удаление кода, по которому уже зарегистрировались, аккаунт не отключает — оно лишь убирает запись.
- **Код передаётся через переменную окружения**, а не подстановкой в текст запроса. Иначе кавычки и спецсимволы в коде сломали бы выполнение или дали возможность инъекции. При создании формат дополнительно проверяется регулярным выражением.
- **Только латиница, цифры, дефис и подчёркивание.** Кириллица отсечена намеренно: код вводится вручную в поле регистрации, и кириллица там источник проблем с раскладкой.
- **Нужен запущенный контейнер `database`.** При остановленном стеке `docker compose exec` вернёт ошибку.
- Приглашения проверяются только при `invite_only = true` в `Revolt.toml`. С открытой регистрацией они игнорируются.

Если приглашения не принимаются

Проверьте, что записи лежат в правильной коллекции и имеют поле `used`:

```
cd /opt/stoat && docker compose exec -T database mongosh revolt --quiet --eval
'db.account_invites.find().forEach(d => print(JSON.stringify(d)))'
```

Если ранее коды создавались по инструкции апстрима, они осели в `invites` — уберите их оттуда, чтобы не путаться:

```
cd /opt/stoat && docker compose exec -T database mongosh revolt --quiet --eval
'printjson(db.invites.deleteMany({}))'
```

> Осторожно: эта команда очищает коллекцию целиком. Она безопасна, только если вы ещё не создавали приглашений в каналы и серверы через интерфейс — иначе удалите записи точечно по `_id`.

---

8. Настройка инстанса

Как устроен Revolt.toml

Локальный `/opt/stoat/Revolt.toml` содержит **только переопределения**. Всё, чего в нём нет, берётся из дефолтного конфига бэкенда — там же полный список опций с комментариями. Поэтому сгенерированный файл такой короткий: в нём лежат адреса сервисов и пара лимитов, остальное работает на умолчаниях.

Чтобы что-то поменять, добавляете нужную секцию целиком с нужными ключами. Секции `[hosts]` и `[hosts.livekit]` руками не трогайте — их генерирует скрипт из вашего домена.

После любой правки:

```
cd /opt/stoat && docker compose restart
```

Именно `restart`, а не `up -d`. Конфиг читается процессами при старте, а `up -d` пересоздаёт контейнер только при изменении его собственных параметров — образа, переменных, портов. Правка примонтированного `Revolt.toml` для compose изменением не является: контейнеры продолжат работать со старым конфигом, и будет выглядеть так, будто настройка не применилась.

Регистрация только по приглашениям

```
[api.registration]
invite_only = true
```

Создание приглашений — в разделе 7 выше.

Почта

Пока `host` пустой, подтверждение email отключено: аккаунты активируются сразу, а восстановление пароля не работает. Для личного инстанса это допустимо, для открытого — нет.

```
[api.smtp]
host = "smtp.example.com"
port = 587
use_tls = true
username = "chat@example.com"
password = "пароль_приложения"
from_address = "chat@example.com"
```

Сроки жизни ссылок настраиваются отдельно (значения в секундах, по умолчанию неделя на верификацию и сутки на сброс пароля):

```
[api.smtp.expiry]
expire_verification = 604800
expire_password_reset = 86400
expire_account_deletion = 86400
```

Капча

Имеет смысл, если регистрация всё-таки открыта. Нужны оба ключа из панели hCaptcha:

```
[api.security.captcha]
hcaptcha_key = "секретный_ключ"
hcaptcha_sitekey = "публичный_ключ"
```

Лимиты пользователей

Лимиты заданы двумя наборами: `new_user` действует первые 72 часа после регистрации (значение меняется через `new_user_hours`), `default` — для всех остальных. Менять обычно нужно оба, иначе новички упрутся в старые значения.

```
[features.limits.new_user]
message_length = 2000
message_attachments = 5
servers = 50
bots = 2
```

```
outgoing_friend_requests = 5
voice_quality = 16000
video = true
video_resolution = [1920, 1080]
video_aspect_ratio = [0.3, 10]

[features.limits.default]
message_length = 2000
message_attachments = 5
servers = 100
bots = 5
outgoing_friend_requests = 10
voice_quality = 16000
video = true
video_resolution = [1920, 1080]
video_aspect_ratio = [0.3, 10]
```

> **Про `video_resolution`.** Это не «1080p», а максимальные ширина и высота потока. В дефолтах апстрима стоит `[1080, 720]`, из-за чего обычная демонстрация экрана превышает лимит и пользователя выкидывает из звонка — известный баг. Скрипт self-hosted при генерации ставит `[1920, 1080]`. Для 4K нужно `[3996, 2160]`.

Размеры загружаемых файлов (в байтах) задаются отдельными подсекциями:

```
[features.limits.default.file_upload_size_limit]
attachments = 20000000
avatars = 4000000
backgrounds = 6000000
icons = 2500000
banners = 6000000
emojis = 500000
```

Если увеличиваете лимит вложений, поднимите и общий предел тела запроса — он должен быть больше любого отдельного файла:

```
[features.limits.global]
body_limit_size = 20000000
new_user_hours = 72
```

```
group_size = 100
server_emoji = 100
server_roles = 200
server_channels = 200
```

Кто может создавать серверы

Пустой список означает «все». Если вписать ID пользователей, создавать серверы смогут только они — удобно для инстанса на несколько человек:

```
[features.limits.global]
restrict_server_creation = ["01ABCDEF..."]
```

Функциональные переключатели

Вебхуки по умолчанию **выключены**:

```
[features]
webhooks_enabled = true
mass_mentions_enabled = true
mass_mentions_send_notifications = true
```

`mass_mentions_enabled = false` полностью запрещает `@everyone` и упоминания ролей. Если оставить их включёнными, но выключить `mass_mentions_send_notifications`, упоминания будут работать в клиенте, но не поднимут пуш-уведомления.

Безопасность загружаемых файлов

Запрет опасных типов и антивирусная проверка:

```
[files]
blocked_mime_types = [
  "application/vnd.microsoft.portable-executable",
  "application/vnd.android.package-archive",
]
clamd_host = "clamav:3310"
scan_mime_types = [
  "application/vnd.microsoft.portable-executable",
```

```
  "application/vnd.android.package-archive",
  "application/zip",
]
```

`clamd_host` в формате `хост:порт` требует отдельно поднятого ClamAV — в стандартном стеке его нет, контейнер придётся добавить через `compose.override.yml`. Пустое значение = сканирование отключено. Пустой `scan_mime_types` означает «сканировать всё».

Превью ссылок

Сервис january генерирует превью для ссылок из сообщений, то есть ходит по внешним адресам от имени сервера. У него была уязвимость с SSRF, поэтому список исключений тут не лишний:

```
[january]
blocked_domains = ["internal.example.com"]
```

Мелочи, которые обычно хочется поменять

```
[api.users]
min_username_length = 2

[api.audit_logs]
expires_after = 2592000

[api.livekit]
call_ring_duration = 30

[features.legal_links]
terms_of_service = ""
privacy_policy = ""
guidelines = ""
```

`expires_after` — срок хранения журнала аудита в секундах (по умолчанию 30 дней), `call_ring_duration` — сколько секунд идёт дозвон в личных сообщениях.

Если инстанс стоит за Cloudflare, включите доверие к его заголовкам, иначе в логах будут IP прокси вместо клиентских:

```
[api.security]
trust_cloudflare = true
```

Чего в Revolt.toml нет

- **Секреты** — только в `secrets.env`.
- **Настройки веб-клиента** — в `.env.web`.
- **Конфигурация LiveKit** — в `livekit.yml`.
- **Адреса сервисов** в `[hosts]` — генерируются скриптом из домена, править вручную бессмысленно.

---

9. Обновление

Перед обновлением прочитайте раздел Notices в README апстрима: там периодически появляются ломающие изменения, требующие ручной миграции.

Свежая версия репозитория:

```
cd /opt/stoat && git pull
```

Убедитесь, что `secrets.env` на месте и не пустой — именно он хранит все секреты инстанса:

```
head -n 20 /opt/stoat/secrets.env
```

> Если инстанс разворачивался **до 28 февраля 2026**, секреты лежали в `Revolt.toml`, и их нужно вручную перенести в `secrets.env` до следующего шага — иначе они будут перезаписаны, а доступ к загруженным файлам потерян. Для установок после этой даты переносить нечего.

Перегенерация конфига. Флаг `--overwrite` затирает `Revolt.toml`, но скрипт предварительно делает резервную копию — свои правки (`invite_only`, лимиты видео) восстанавливаются из неё:

```
cd /opt/stoat && ./generate_config.sh --overwrite chat.example.com
```

Образы:

```
cd /opt/stoat && docker compose pull
```

Перезапуск:

```
cd /opt/stoat && docker compose up -d
```

---

10. Переопределения стека

Свои изменения вносятся через файл переопределений, а не правкой `compose.yml` — иначе их снесёт ближайший `git pull`.

Старый процессор без поддержки свежей MongoDB — пин на 4.4 с заменой healthcheck:

```
services:
  database:
    image: mongo:4.4
    healthcheck:
      test: echo 'db.runCommand("ping").ok' | mongo localhost:27017/test --quiet
```

ARM или системы без поддержки свежего KeyDB — замена на Valkey:

```
services:
  redis:
    image: valkey/valkey:8
```

Доступ к БД снаружи — только на loopback. Публикация «в мир» обойдёт правила UFW, и база станет доступна всем на запись:

```
services:
  database:
    ports:
      - "127.0.0.1:27017:27017"
```

---

11. Бэкап

Две вещи, без которых восстановление невозможно.

Секреты — уже скопированы на шаге 4, но копия должна жить не только на этом сервере.

База — в ней все сообщения, серверы и пользователи:

```
cd /opt/stoat && docker compose exec -T database mongodump --archive --gzip > ~/stoat-db-$(date +%F).archive.gz
```

Дальше выгружайте оба файла на другой хост (`restic`, `borg`, `rsync`) — бэкап, лежащий рядом с данными, защищает только от собственных ошибок.

---

12. Диагностика

Логи всего стека:

```
cd /opt/stoat && docker compose logs -f
```

Логи одного сервиса (имена смотреть в `docker compose ps`):

```
cd /opt/stoat && docker compose logs -f api
```

Что упало и с каким кодом:

```
cd /opt/stoat && docker compose ps -a
```

Типовые ситуации:

- **Веб-клиент открывается, но чат не подключается** — не проходит WebSocket. Проверьте домен в `Revolt.toml` и то, что Caddy не рвёт апгрейд соединения.
- **Текст работает, голос нет** — не проброшены `7881/tcp` и `50000-50100/udp`. Проверяется через `nc -vz` снаружи.
- **Выкидывает при включении видео** — поток превышает лимит. Увеличьте `video_resolution` в секциях `[features.limits.new_user]` и `[features.limits.default]` в `Revolt.toml`; для 4K это `[3996, 2160]`.
- **Регистрация отклоняется с `InvalidInvite`** — код лежит в коллекции `invites` вместо `account_invites` либо в документе нет поля `used: false`.
- **Правка `Revolt.toml` не применяется** — контейнеры не перезапускались. `docker compose up -d` при изменении только примонтированного файла ничего не пересоздаёт, нужен `docker compose restart`.
- **Все Rust-сервисы паникуют с `Permission denied (os error 13)` в `crates/core/config`** — на `Revolt.toml` слишком строгие права. Внутри контейнеров он читается другим UID, нужен `chmod 644`.

- **Контейнеры циклически перезапускаются после обновления** — почти всегда несовпадение секретов между `Revolt.toml` и `secrets.env`.

---

13. Безопасность

У проекта регулярно публикуются advisories — от неограниченного создания аккаунтов до SSRF в сервисе превью ссылок и хранимой XSS в веб-клиенте. Список ведётся в конце README апстрима. Обновляться стоит по факту публикации, а не когда что-то сломается.

Минимальный набор для личного инстанса: закрытая регистрация, свежие образы, `secrets.env` в бэкапе, БД не опубликована наружу.