Nginx

Это подробное руководство (уровень «с нуля») по работе с контейнером NGINX и встроенной библиотекой-обёрткой NiceSOFT. Здесь вы научитесь запускать контейнер, настраивать каталоги и переменные окружения, управлять серверными блоками (server blocks), выпускать обычные и ГОСТ-сертификаты, а также отлаживать типовые проблемы.

1) Требования

- Установленный **Docker** или **Podman**.
- Базовые знания командной строки Linux.
- Доступ для создания/монтирования директорий на хосте (права на запись).
- Для ГОСТ-сертификатов — сборка OpenSSL в контейнере с поддержкой GOST (через *engine* или *provider*).

На системах с SELinux используйте суффикс `:Z` при монтировании томов, чтобы автоматом выставлялись корректные контекстные метки.

2) Структура каталогов

Рекомендуемая структура в корне платформы `/srv/nicesoft` (можно изменить через переменные):

```
/srv/nicesoft/
├── conf/                  # Главный конфиг NGINX и включаемые файлы
│   ├── nginx.conf         # Основной конфигурационный файл NGINX
│   └── nicesoft/          # Ваши дополнительные конфиги (prefix и пр.)
├── server-blocks/         # Server blocks (виртуальные хосты)
├── stream.d/              # Конфиги stream (tcp/udp), если используется
├── scripts/
│   └── nginx/
```

```
□   □□ nicesoft-templates/    # Шаблоны server blocks/prefix
□□ nicesoft/
□  □□ certs/                  # TLS-сертификаты (tls.crt/tls.key и т.д.)
□□ tmp/                      # Временные файлы/пид
□□ init.d/                   # Пользовательские init-скрипты
```

Дополнение: Библиотека использует `NGINX_PID_FILE` (по умолчанию `${NGINX_TMP_DIR}/nginx.pid`) для проверки статуса NGINX. Убедитесь, что `tmp/` доступен для записи.

3) Ключевые переменные окружения

Переменная	Назначение	Пример
`NICESOFT_ROOT_DIR`	Корень платформы	`/srv/nicesoft`
`NGINX_CONF_DIR`	Каталог конфигов NGINX	`/srv/nicesoft/conf`
`NGINX_CONF_FILE`	Основной конфиг NGINX	`/srv/nicesoft/conf/nginx.conf`
`NGINX_SERVER_BLOCKS_DIR`	Каталог server blocks	`/srv/nicesoft/server-blocks`
`NGINX_STREAM_SERVER_BLOCKS_DIR`	Каталог stream-конфигов	`/srv/nicesoft/stream.d`
`NGINX_DEFAULT_CONF_DIR`	Каталог дефолтных конфигов для первичного копирования	`/etc/nginx-defaults`
`NGINX_VOLUME_DIR`	Корневой volume (проверки legacy-путей)	`/srv/nicesoft`
`NGINX_TMP_DIR`	Временные файлы NGINX (pid и пр.)	`/srv/nicesoft/tmp`
`NGINX_INITSCRIPTS_DIR`	Каталог пользовательских init-скриптов	`/srv/nicesoft/init.d`
`NGINX_DAEMON_USER`	Системный пользователь демона	`nginx`

Переменная	Назначение	Пример
NGINX_DAEMON_GROUP	Системная группа демона	nginx
NGINX_HTTP_PORT_NUMBER	HTTP-порт по умолчанию	8080
NGINX_HTTPS_PORT_NUMBER	HTTPS-порт по умолчанию	8443
NGINX_ENABLE_ABSOLUTE_REDIRECT	on/off для absolute_redirect	yes / no
NGINX_ENABLE_PORT_IN_REDIRECT	on/off для port_in_redirect	yes / no
NGINX_ENABLE_STREAM	Включить блок stream{} в конфиге	yes / no
NGINX_SKIP_SAMPLE_CERTS	Не создавать тестовые сертификаты	yes / no
NS_CREATE_BACKUP	Создавать .bak при атомарной записи	yes / no
NGINX_DEFAULT_HTTP_PORT_NUMBER	Дефолтный HTTP-порт (fallback для шаблонов)	80
NGINX_DEFAULT_HTTPS_PORT_NUMBER	Дефолтный HTTPS-порт (fallback для шаблонов)	443

Примечание: булевы значения допускают yes/no и true/false. Все переменные корня платформы должны использовать NICESOFT_ROOT_DIR (как указано в скрипте).

4) Быстрый старт

4.1 Подготовка каталогов

```
mkdir -p /srv/nicesoft/{conf,conf.d,server-blocks,stream.d,scripts/nginx/nicesoft-
```

```
templates,nicesoft/certs,tmp,init.d}
```

4.2 Запуск контейнера (Docker)

```
docker run -d --name nicesoft-nginx \
 -p 8080:8080 -p 8443:8443 \
 -e NICESOFT_ROOT_DIR=/srv/nicesoft \
 -e NGINX_CONF_DIR=/srv/nicesoft/conf \
 -e NGINX_CONF_FILE=/srv/nicesoft/conf/nginx.conf \
 -e NGINX_SERVER_BLOCKS_DIR=/srv/nicesoft/server-blocks \
 -e NGINX_STREAM_SERVER_BLOCKS_DIR=/srv/nicesoft/stream.d \
 -e NGINX_DEFAULT_CONF_DIR=/etc/nginx-defaults \
 -e NGINX_VOLUME_DIR=/srv/nicesoft \
 -e NGINX_TMP_DIR=/srv/nicesoft/tmp \
 -e NGINX_INITSCRIPTS_DIR=/srv/nicesoft/init.d \
 -e NGINX_DAEMON_USER=nginx \
 -e NGINX_DAEMON_GROUP=nginx \
 -e NGINX_HTTP_PORT_NUMBER=8080 \
 -e NGINX_HTTPS_PORT_NUMBER=8443 \
 -v /srv/nicesoft:/srv/nicesoft \
 your-registry/nicesoft-nginx:latest
```

На SELinux-системах добавьте `-v /srv/nicesoft:/srv/nicesoft:Z`.

4.3 Вход внутрь контейнера и подключение библиотеки

```
docker exec -it nicesoft-nginx bash

# Предположим, файл помощника: /srv/nicesoft/scripts/nginx-helper.sh

# Подключаем его в текущую оболочку:
. /srv/nicesoft/scripts/nginx-helper.sh
```

Дополнение: Библиотека загружает общие модули (`libfs.sh`, `liblog.sh` и т.д.) из `/srv/nicesoft/scripts/`. Убедитесь, что они присутствуют в образе.

5) Шаблоны NGINX (server block / prefix)

Шаблоны лежат в `${NICESOFT_ROOT_DIR}/scripts/nginx/nicesoft-templates`. Имена по умолчанию:

- app-http-server-block.conf.tpl
- app-https-server-block.conf.tpl
- app-prefix.conf.tpl
- Варианты по типу: app-<type>-http-server-block.conf.tpl, app-<type>-https-server-block.conf.tpl, app-<type>-prefix.conf.tpl

Шаблоны рендерятся утилитой render-template (подстановки из переменных окружения). Пример простого HTTP-шаблона:

```
server {
${http_listen_configuration}
${server_name_configuration}
${acl_configuration}
    root ${document_root};
${additional_configuration}
${external_configuration}
}
```

Хелпер сам готовит переменные (например, http_listen_configuration, server_name_configuration) перед рендером. Для HTTPS добавляются ssl_certificate и ssl_certificate_key (по умолчанию из certs/tls.crt и tls.key).

6) Инициализация и проверка окружения

6.1 Инициализация

Выполняет первичные действия: копирование дефолтов, чистку PID, проверку legacy-путей, настройку пользователя/группы, портов и redirect-директив.

```
nginx_initialize
```

6.2 Валидация переменных

Проверяет корректность значений NGINX_* и предупреждает о невозможности применить env-настройки, если файл недоступен для записи.

```
nginx_validate || echo "Обнаружены проблемы — исправьте и повторите."
```

Дополнение: Валидация использует validate_port из libvalidations.sh (с опцией -unprivileged

для не-root). Поддерживает порты 1–65535, с учётом привилегий.

7) Работа с server blocks

7.1 Создание конфигурации приложения

Создаёт `<app>-server-block.conf` и `<app>-https-server-block.conf` (или их отключённые версии с суффиксом `.disabled`).

```
ensure_nginx_app_configuration_exists myapp \
 --type php \
 --server-name myapp.local \
 --server-aliases "www.myapp.local api.myapp.local" \
 --hosts "127.0.0.1 ::1" \
 --http-port 8080 \
 --https-port 8443 \
 --document-root "/srv/nicesoft/myapp/public" \
 --allow-remote-connections yes \
 --additional-configuration "index index.php;" \
 --external-configuration ""
```

Если `--disable`, `--disable-http` или `--disable-https` — соответствующие файлы создаются с суффиксом `.disabled`. Шаблон выбирается по `--type` (fallback: `app`). ACL добавляется автоматически для `no` в `--allow-remote-connections` (403 для не-127.0.0.1).

7.2 Удаление конфигурации

```
ensure_nginx_app_configuration_not_exists myapp
```

8) Префикс-конфигурации (location /prefix)

Создаёт файл `${NGINX_CONF_DIR}/nicesoft/<app>.conf` со встроенным `location /<app>` (или своим префиксом).

```
ensure_nginx_prefix_configuration_exists tools \
 --type static \
 --prefix /tools \
 --document-root "/srv/nicesoft/tools" \
 --allow-remote-connections no \
```

```
 --additional-configuration "autoindex on;"
```

Дополнение: Префикс по умолчанию `/${app}`. ACL аналогично server blocks: 403 для не-localhost при `no`.

9) Обновление конфигов (порты/хосты/включение)

9.1 Перенастройка listen/имён/режима включения

```
nginx_update_app_configuration myapp \
 --hosts "0.0.0.0 ::" \
 --server-name "myapp.example.local" \
 --server-aliases "myapp api.myapp" \
 --http-port 80 \
 --https-port 443 \
 --enable-http \
 --enable-https
```

9.2 Быстрая правка директив в `nginx.conf`

```
# Включить абсолютные редиректы
nginx_configure absolute_redirect on

# Включить порт в редиректах
nginx_configure port_in_redirect on
```

9.3 Замена порта в файле server block

```
nginx_configure_port 8080 "/srv/nicesoft/server-blocks/myapp-server-block.conf"
```

Дополнение: `nginx_configure` и `nginx_configure_port` используют атомарную запись с проверкой родителя. Для портов поддержка IPv6 (паттерн матчит `[::]:80`). Если файл пустой после рендера — обновление пропускается с предупреждением.

10) Сертификаты TLS

10.1 Генерация тестовых RSA-сертификатов (без пароля)

Создаёт `tls.crt` и `tls.key` в `${NGINX_CONF_DIR}/nicesoft/certs`, если их нет, и не выставлено `NGINX_SKIP_SAMPLE_CERTS=yes`.

```
nginx_generate_sample_certs
```

Приватный ключ сохраняется с правами `0600`. Для production используйте полноценную выдачу от доверенного УЦ. SAN: `DNS:example.com,DNS:www.example.com,IP:127.0.0.1`. Совместимо с OpenSSL 1.0+ (адаптация флагов).

11) ГОСТ-сертификаты (самоподписанные)

Функция проверяет наличие поддержки GOST в OpenSSL (через *provider* или *engine*) и создаёт ключ/сертификат с SAN.

11.1 Базовый сценарий

```
# CN = example.com, SAN подставится автоматически (DNS:example.com)
nginx_generate_gost_certs --cn example.com
```

11.2 Явное указание SAN, алгоритма и срока

```
# Несколько SAN и алгоритм 512, срок 825 дней
nginx_generate_gost_certs \
 --cn example.com \
 --san "DNS:example.com,DNS:www.example.com,IP:127.0.0.1" \
 --algo 512 \
 --days 825
```

11.3 Задание путей вывода

```
nginx_generate_gost_certs \
 --cn 10.0.0.5 \
 --out-dir /srv/nicesoft/nicesoft/certs \
```

```
 --key-out /srv/nicesoft/nicesoft/certs/my.key \
 --crt-out /srv/nicesoft/nicesoft/certs/my.crt
```

Поддерживаются алгоритмы: `--algo 256` (gost2012_256) и `--algo 512` (gost2012_512). Если GOST не найден, функция завершится ошибкой и сообщит, что нужно установить и включить поддержку. SAN нормализуется автоматически (IP/DNS на основе CN).

Проверка наличия GOST в OpenSSL

```
# Провайдеры (OpenSSL 3.x)
openssl list -providers | grep -i gost

# Движки (OpenSSL 1.1.x или 3.x с engine)
openssl engine -t | grep -i gost

# Альтернатива: список алгоритмов
openssl list -public-key-algorithms | grep -i gost
```

Дополнение: Детекция: сначала провайдеры, затем engine, fallback на алгоритмы. Генерация использует `-pkeyopt paramset:A`. Fallback: `genpkey + req + x509` если `req -x509` не сработает.

12) Пользовательские init-скрипты

Все `.sh` в `${NGINX_INITSCRIPTS_DIR}` выполняются по порядку (сначала исполняются скрипты с флагом `+x`, остальные — «подключаются» через `source`).

```
# Разместите, например:
/srv/nicesoft/init.d/10-fix-perms.sh
/srv/nicesoft/init.d/20-generate-certs.sh
# Затем:
nginx_custom_init_scripts
```

После выполнения init-скриптов вызывается `nginx_stop` — это помогает корректно перезапустить сервис с новыми параметрами в вашем entrypoint. Скрипты сортируются по имени для предсказуемости.

13) Диагностика и типовые проблемы

«File not writable» / «Parent directory not writable»

Проверьте права на файл/родительский каталог. Для новых файлов важна доступность на запись родителя. На SELinux-системах убедитесь, что том смонтирован с `:Z`. При необходимости скорректируйте владельца/группу.

```
chown -R 1000:1000 /srv/nicesoft
chmod -R g+rw /srv/nicesoft
```

«OpenSSL GOST support not found»

В образе отсутствует поддержка GOST. Установите пакет/провайдер/engine GOST для OpenSSL или используйте образ с включённым GOST. Проверьте провайдеры/движки (см. раздел «ГОСТ-сертификаты»).

«legacy vhosts not supported»

Старые пути (`${NGINX_VOLUME_DIR}/conf/vhosts`) не поддерживаются. Используйте `${NGINX_SERVER_BLOCKS_DIR}`.

«Empty render skipped» или проблемы с шаблонами

Рендер шаблона дал пустой результат (ошибка в переменных или шаблоне). Проверьте экспортированные переменные (`export document_root=...`) и наличие шаблона. Логи: `sed '/^\s*$/d'` удаляет пустые строки.

NGINX не стартует после init-скриптов

`nginx_custom_init_scripts` вызывает `nginx_stop` в конце — убедитесь, что ваш entrypoint перезапускает NGINX после. Проверьте PID: `is_nginx_running`.

14) Рекомендации по безопасности

- Приватные ключи храните с правами не шире `0600`.
- Включите только необходимые `server_name` и `listen` — избегайте «лишних» `default_server`.
- Ограничивайте доступ в префикс-конфигурациях: `--allow-remote-connections no`

создаёт 403 для не-localhost.
- Регулярно обновляйте образ контейнера и пакеты внутри.
- Включайте `NS_CREATE_BACKUP=yes` при критичных изменениях — получаете `.bak` до перезаписи.
- **Дополнение:** Атомарная запись (`_ns_atomic_render`) создаёт temp-файлы в родителе и использует `mv -f` + `chmod g+rw`. Избегайте `root` для unprivileged-run.

15) Пример Docker Compose

```
services:
  nicesoft-nginx:
    image: your-registry/nicesoft-nginx:latest
    container_name: nicesoft-nginx
    ports:
      - "8080:8080"
      - "8443:8443"
    environment:
      NICESOFT_ROOT_DIR: /srv/nicesoft
      NGINX_CONF_DIR: /srv/nicesoft/conf
      NGINX_CONF_FILE: /srv/nicesoft/conf/nginx.conf
      NGINX_SERVER_BLOCKS_DIR: /srv/nicesoft/server-blocks
      NGINX_STREAM_SERVER_BLOCKS_DIR: /srv/nicesoft/stream.d
      NGINX_DEFAULT_CONF_DIR: /etc/nginx-defaults
      NGINX_VOLUME_DIR: /srv/nicesoft
      NGINX_TMP_DIR: /srv/nicesoft/tmp
      NGINX_INITSCRIPTS_DIR: /srv/nicesoft/init.d
      NGINX_DAEMON_USER: nginx
      NGINX_DAEMON_GROUP: nginx
      NGINX_HTTP_PORT_NUMBER: "8080"
      NGINX_HTTPS_PORT_NUMBER: "8443"
      NS_CREATE_BACKUP: "yes"
    volumes:
      - /srv/nicesoft:/srv/nicesoft:Z
```

16) Пример systemd-юнита для Docker

```
[Unit]
Description=NiceSOFT NGINX Container
Wants=network-online.target
After=network-online.target docker.service

[Service]
Restart=always
ExecStart=/usr/bin/docker start -a nicesoft-nginx
ExecStop=/usr/bin/docker stop -t 10 nicesoft-nginx
```

```
[Install]
WantedBy=multi-user.target
```

17) Часто задаваемые вопросы (FAQ)

Как перезагрузить NGINX после изменений?

Внутри контейнера выполните стандартную команду (в зависимости от образа):

```
nginx -t && nginx -s reload
```

Или используйте ваш процесс-менеджер/entrypoint, если он предусмотрен. Некоторые init-скрипты могут завершаться вызовом `nginx_stop`, после чего сервис поднимется с новыми файлами.

Где искать логи?

Зависит от вашего образа и настроек NGINX (директивы `access_log` и `error_log`). Часто логи доступны через `docker logs nicesoft-nginx`, а также в файлах, указанных в конфиге `nginx.conf`. **Дополнение:** Библиотека использует `liblog.sh` для debug/info/warn/error — проверьте `NGINX_ERROR_LOG` в env.

Можно ли создавать server block без HTTPS?

Да. При создании можете указать `--disable-https` (или при обновлении — не включать HTTPS). Аналогично можно отключить HTTP: `--disable-http`.

Как проверить статус NGINX?

Используйте `is_nginx_running` (возвращает 0 если запущен) или `is_nginx_not_running`. Основывается на PID из `NGINX_PID_FILE` и `is_service_running`.

Что делать, если stream не включается?

Установите `NGINX_ENABLE_STREAM=yes` перед `nginx_initialize`. Проверяет отсутствие дублирующего `include` в `nginx.conf`.
Готово! Теперь вы можете уверенно управлять конфигурациями NGINX в контейнере,

использовать атомарные обновления файлов, подключать пользовательские init-скрипты и выпускать как RSA-, так и ГОСТ-сертификаты.