# Nginx

<https://nginx.org/en/#basic_http_features>

nginx («engine x») — это HTTP-веб-сервер, обратный прокси-сервер, кэширование контента, балансировщик нагрузки, TCP/UDP-прокси-сервер и почтовый прокси-сервер. Изначально разработан Игорем Сысоевым и распространяется по лицензии BSD с двумя пунктами.

## Определения

Forward proxy – когда клиент скрывает себя от сервера через клиентский прокси.

Reverse proxy – когда сервер скрывает себя от клиента через серверный прокси.

## Способности Nginx

**Basic HTTP server features**

* Serving static and [index](https://nginx.org/en/docs/http/ngx_http_index_module.html) files, [autoindexing](https://nginx.org/en/docs/http/ngx_http_autoindex_module.html); [open file descriptor cache](https://nginx.org/en/docs/http/ngx_http_core_module.html#open_file_cache);
* [Accelerated reverse proxying with caching](https://nginx.org/en/docs/http/ngx_http_proxy_module.html); [load balancing and fault tolerance](https://nginx.org/en/docs/http/ngx_http_upstream_module.html);
* Accelerated support with caching of [FastCGI](https://nginx.org/en/docs/http/ngx_http_fastcgi_module.html), [uwsgi](https://nginx.org/en/docs/http/ngx_http_uwsgi_module.html), [SCGI](https://nginx.org/en/docs/http/ngx_http_scgi_module.html), and [memcached](https://nginx.org/en/docs/http/ngx_http_memcached_module.html) servers; [load balancing and fault tolerance](https://nginx.org/en/docs/http/ngx_http_upstream_module.html);
* Modular architecture. Filters include [gzipping](https://nginx.org/en/docs/http/ngx_http_gzip_module.html), byte ranges, chunked responses, [XSLT](https://nginx.org/en/docs/http/ngx_http_xslt_module.html), [SSI](https://nginx.org/en/docs/http/ngx_http_ssi_module.html), and [image transformation](https://nginx.org/en/docs/http/ngx_http_image_filter_module.html) filter. Multiple SSI inclusions within a single page can be processed in parallel if they are handled by proxied or FastCGI/uwsgi/SCGI servers;
* [SSL and TLS SNI support](https://nginx.org/en/docs/http/ngx_http_ssl_module.html);
* Support for [HTTP/2](https://nginx.org/en/docs/http/ngx_http_v2_module.html) with weighted and dependency-based prioritization;
* Support for [HTTP/3](https://nginx.org/en/docs/http/ngx_http_v3_module.html).

**Other HTTP server features**

* Name-based and IP-based [virtual servers](https://nginx.org/en/docs/http/request_processing.html);
* [Keep-alive](https://nginx.org/en/docs/http/ngx_http_core_module.html#keepalive_timeout) and pipelined connections support;
* [Access log formats](https://nginx.org/en/docs/http/ngx_http_log_module.html#log_format), [buffered log writing](https://nginx.org/en/docs/http/ngx_http_log_module.html#access_log), [fast log rotation](https://nginx.org/en/docs/control.html#logs), and [syslog logging](https://nginx.org/en/docs/syslog.html);
* 3xx-5xx error codes [redirection](https://nginx.org/en/docs/http/ngx_http_core_module.html#error_page);
* The rewrite module: [URI changing using regular expressions](https://nginx.org/en/docs/http/ngx_http_rewrite_module.html);
* [Executing different functions](https://nginx.org/en/docs/http/ngx_http_rewrite_module.html#if) depending on the [client address](https://nginx.org/en/docs/http/ngx_http_geo_module.html);
* Access control based on [client IP address](https://nginx.org/en/docs/http/ngx_http_access_module.html), [by password (HTTP Basic authentication)](https://nginx.org/en/docs/http/ngx_http_auth_basic_module.html) and by the [result of subrequest](https://nginx.org/en/docs/http/ngx_http_auth_request_module.html);
* Validation of [HTTP referer](https://nginx.org/en/docs/http/ngx_http_referer_module.html);
* The [PUT, DELETE, MKCOL, COPY, and MOVE](https://nginx.org/en/docs/http/ngx_http_dav_module.html) methods;
* [FLV](https://nginx.org/en/docs/http/ngx_http_flv_module.html) and [MP4](https://nginx.org/en/docs/http/ngx_http_mp4_module.html) streaming;
* [Response rate limiting](https://nginx.org/en/docs/http/ngx_http_core_module.html#limit_rate);
* Limiting the number of simultaneous [connections](https://nginx.org/en/docs/http/ngx_http_limit_conn_module.html) or [requests](https://nginx.org/en/docs/http/ngx_http_limit_req_module.html) coming from one address;
* [IP-based geolocation](https://nginx.org/en/docs/http/ngx_http_geoip_module.html);
* [A/B testing](https://nginx.org/en/docs/http/ngx_http_split_clients_module.html);
* [Request mirroring](https://nginx.org/en/docs/http/ngx_http_mirror_module.html);
* Embedded [Perl](https://nginx.org/en/docs/http/ngx_http_perl_module.html);
* [njs](https://nginx.org/en/docs/njs/index.html) scripting language.

**Mail proxy server features**

* User redirection to [IMAP](https://nginx.org/en/docs/mail/ngx_mail_imap_module.html) or [POP3](https://nginx.org/en/docs/mail/ngx_mail_pop3_module.html) server using an external HTTP [authentication](https://nginx.org/en/docs/mail/ngx_mail_auth_http_module.html) server;
* User authentication using an external HTTP [authentication](https://nginx.org/en/docs/mail/ngx_mail_auth_http_module.html) server and connection redirection to an internal [SMTP](https://nginx.org/en/docs/mail/ngx_mail_smtp_module.html) server;
* Authentication methods:
  + [POP3](https://nginx.org/en/docs/mail/ngx_mail_pop3_module.html#pop3_auth): USER/PASS, APOP, AUTH LOGIN/PLAIN/CRAM-MD5;
  + [IMAP](https://nginx.org/en/docs/mail/ngx_mail_imap_module.html#imap_auth): LOGIN, AUTH LOGIN/PLAIN/CRAM-MD5;
  + [SMTP](https://nginx.org/en/docs/mail/ngx_mail_smtp_module.html#smtp_auth): AUTH LOGIN/PLAIN/CRAM-MD5;
* [SSL](https://nginx.org/en/docs/mail/ngx_mail_ssl_module.html) support;
* [STARTTLS and STLS](https://nginx.org/en/docs/mail/ngx_mail_ssl_module.html#starttls) support.

**TCP/UDP proxy server features**

* [Generic proxying](https://nginx.org/en/docs/stream/ngx_stream_proxy_module.html) of TCP and UDP;
* [SSL](https://nginx.org/en/docs/stream/ngx_stream_ssl_module.html) and TLS [SNI](https://nginx.org/en/docs/stream/ngx_stream_ssl_preread_module.html) support for TCP;
* [Load balancing and fault tolerance](https://nginx.org/en/docs/stream/ngx_stream_upstream_module.html);
* Access control based on [client address](https://nginx.org/en/docs/stream/ngx_stream_access_module.html);
* Executing different functions depending on the [client address](https://nginx.org/en/docs/stream/ngx_stream_geo_module.html);
* Limiting the number of simultaneous [connections](https://nginx.org/en/docs/stream/ngx_stream_limit_conn_module.html) coming from one address;
* [Access log formats](https://nginx.org/en/docs/stream/ngx_stream_log_module.html#log_format), [buffered log writing](https://nginx.org/en/docs/stream/ngx_stream_log_module.html#access_log), [fast log rotation](https://nginx.org/en/docs/control.html#logs), and [syslog logging](https://nginx.org/en/docs/syslog.html);
* [IP-based geolocation](https://nginx.org/en/docs/stream/ngx_stream_geoip_module.html);
* [A/B testing](https://nginx.org/en/docs/stream/ngx_stream_split_clients_module.html);
* [njs](https://nginx.org/en/docs/njs/index.html) scripting language.

## Basic HTTP server features

1. Serving static files

Nginx может напрямую отдавать файлы с диска: HTML, CSS, JS, изображения. Фронтенд SPA, документация, картинки — без участия бекенда.

Напрямую отдавать» значит, что Nginx берёт файл с диска и отправляет его клиенту (браузеру) **без передачи запроса на бэкенд и без выполнения кода приложения**.

Пример:

* Браузер запрашивает /static/app.js.
* Nginx ищет app.js на диске и возвращает его как есть.
* FastAPI или другой сервер **не участвуют**, код Python/Rust не исполняется.

1. Serving index files

Когда пользователь заходит на /some/path/, Nginx автоматически ищет index.html или другой заданный индексный файл и отдает его. Это удобно для каталогов, SPA или статических сайтов.

1. Autoindexing

Если в директории нет index.html, Nginx может генерировать список файлов в каталоге. Полезно для быстрого просмотра содержимого папок на сервере.

1. Open file descriptor cache

Каждый открытый файл на сервере использует дескриптор (ресурс ОС). Nginx кеширует дескрипторы, чтобы не открывать и закрывать файлы каждый раз, что ускоряет отдачу статики.

Nginx кэширует файлы, чтобы моментально отдать их при таком же запросе и не тратить время на повторные операции

## Other HTTP server features

## Дефолтные способности

Дефолтные способности - это способности nginx которые будут автоматически использоваться при самой минимальной его настройке (например просто возвращение фронта при запросе в рут, простое проксирование и всё).

| **Способность / Фича** | **Дефолтная (работает сразу)** | **Требует настройки** |
| --- | --- | --- |
| Отдача статических файлов (index.html, JS, CSS, изображения) | ✅ | — |
| Reverse proxy (проксирование запросов на бекенд) | ✅ (без особых правил, через proxy\_pass) | Часто требуется настройка для конкретных URL |
| Keep-alive соединения | ✅ | — |
| Логирование доступа (access.log) | ✅ | Формат, буферизация, ротация требуют настройки |
| Логирование ошибок (error.log) | ✅ | Формат и уровень логирования можно настроить |
| Обработка ошибок HTTP (404, 500, 502 и др.) | ✅ | Кастомные страницы требуют настройки |
| Gzip-сжатие | ❌ | Нужно включить через конфиг |
| Load balancing (балансировка между бекендами) | ❌ | Нужно настраивать upstream и алгоритмы |
| Кэширование ответов бекенда | ❌ | Нужно включить proxy\_cache или fastcgi\_cache |
| Rate limiting / ограничение запросов | ❌ | Нужно настроить limit\_req / limit\_conn |
| IP-геолокация | ❌ | Нужно подключить geoip-модуль и базу |
| A/B тестирование | ❌ | Нужно настроить через конфигурацию |
| Request mirroring | ❌ | Требует явной настройки |
| SSI (Server Side Includes) | ❌ | Включается через ssi on; |
| HTTP/2, HTTP/3 | ❌ | Нужно включить и настроить SSL/TLS |
| Embedded Perl / njs | ❌ | Нужно включить и писать скрипты |
| Автоиндексация (autoindex) | ❌ | Включается через autoindex on; |
| PUT, DELETE, MKCOL, COPY, MOVE методы | ❌ | Нужно явно разрешить через конфиг |

| **Функциональность** | **Описание** |
| --- | --- |
| Обслуживание статического контента | NGINX по умолчанию обслуживает статические файлы (HTML, изображения и т.д.) из указанной директории (например, /usr/share/nginx/html). При запросе к корню (/) возвращает index.html или аналогичный файл, если он существует. |
| Базовое проксирование (reverse proxy) | С минимальной настройкой (добавлением директивы proxy\_pass в location) NGINX может проксировать запросы к бэкенду (например, к приложению на другом порту), передавая запросы и возвращая ответы. |
| Проксирование FastCGI | Поддержка проксирования к FastCGI-серверам (например, для PHP), с передачей параметров вроде SCRIPT\_FILENAME. Это доступно с базовой конфигурацией. |
| Логирование доступа и ошибок | Автоматически логирует запросы в access.log и ошибки в error.log с дефолтными форматами. |
| Сжатие gzip | Включено по умолчанию для определенных MIME-типов (например, text/html), чтобы уменьшить размер передаваемых данных. |
| Обработка MIME-типов | Включает файл mime.types для правильного определения типов контента в ответах. |
| Прослушивание порта 80 | По умолчанию слушает HTTP-запросы на порту 80 и отвечает на localhost. |
| Базовая обработка ошибок | Возвращает стандартные страницы ошибок (например, 404 для несуществующих файлов) и 50x.html для внутренних ошибок. |

## Мониторинг