# Установка и настройка веб-сервера NGINX

## Краткое руководство и демонстрация конфигураций

# Аннотация

- установка NGINX в Ubuntu;
- запуск и проверка службы;
- создание виртуального хоста;
- настройка HTTPS (Let's Encrypt, самоподписанный, коммерческие варианты, TLS);
- ограничение частоты запросов (Rate Limiting);
- рекомендации по безопасности;
- логирование и диагностика;
- балансировка нагрузки и проксирование (proxy).

# Что такое NGINX

NGINX — высокопроизводительный веб-сервер и обратный прокси.
Основные преимущества:

- поддержка тысяч одновременных соединений;
- удобная текстовая конфигурация;
- поддержка HTTPS, HTTP/2, gzip, кэширования;
- может выступать как load balancer и API-gateway.

# Установка NGINX в Ubuntu

```
sudo apt update
sudo apt install -y nginx
```

# Запуск службы

```
sudo systemctl enable --now nginx
sudo systemctl status nginx
```

Проверка занятых портов:

```
ss -tulpn | grep -E ':80|:443'
```

# UFW и открытие портов

UFW (Uncomplicated Firewall) — удобный фронтенд для iptables. Для работы веб-сервера нужны:

- 80/tcp — HTTP (негашёный трафик, ACME challenge для Let's Encrypt);
- 443/tcp — HTTPS (TLS);
- опционально: 22/tcp — SSH (для управления сервером).

Откройте порты командой:

```
sudo ufw allow 'Nginx Full'    #                                80
      443
sudo ufw allow OpenSSH         #
   SSH
sudo ufw enable                #                              UFW (
                                                 )
sudo ufw status verbose
```

# UFW и открытие портов

- Если используешь нестандартные порты — открой именно их (например, 8080).
- Для автоматизированного получения Let's Encrypt требуется, чтобы порт 80 был доступен извне (HTTP-01).
- В облаках (AWS/GCP) проверь также сетевые группы / ACL.

# Создание корня сайта

```
sudo mkdir -p /var/www/example.local/public_html
echo "OK: example.local" \
 | sudo tee /var/www/example.local/public_html/index.
    html
```

Права доступа:

```
sudo chown -R www-data:www-data /var/www/example.local
sudo find /var/www/example.local -type d -exec chmod
   755 {} \;
sudo find /var/www/example.local -type f -exec chmod
   644 {} \;
```

# Конфигурация и активация сайта

Файл: /etc/nginx/sites-available/example.local

```
server {
    listen 80;
    server_name example.local www.example.local;
    root /var/www/example.local/public_html;
    index index.html;

    access_log /var/log/nginx/example.local.access.log
        ;
    error_log  /var/log/nginx/example.local.error.log
        warn;

    location / {
        try_files $uri $uri/ =404;
    }
}
```

# Конфигурация и активация сайта

Активация:

```
sudo ln -s /etc/nginx/sites-available/example.local \
           /etc/nginx/sites-enabled/
sudo nginx -t
sudo systemctl reload nginx
```

# Let's Encrypt (публичный домен)

```
sudo apt install -y certbot python3-certbot-nginx
sudo certbot --nginx -d example.com -d www.example.com
```

**Требования:**

- DNS A/AAAA записи должны указывать на сервер;
- порт 80 должен быть доступен извне (для HTTP-01 challenge);
- certbot может автоматически обновлять конфигурацию NGINX и настроить редирект на HTTPS.

# Самоподписанный сертификат (локально)

```
sudo mkdir -p /etc/ssl/localcerts
sudo openssl req -x509 -newkey rsa:2048 \
 -nodes -days 365 \
 -keyout /etc/ssl/localcerts/example.local.key \
 -out /etc/ssl/localcerts/example.local.crt \
 -subj "/CN=example.local" \
 -addext "subjectAltName=DNS:example.local,DNS:www.
    example.local"
```

Браузер будет предупреждать о ненадёжности — подходит только для разработки.

# HTTPS-конфигурация (пример)

```
server {
    listen 443 ssl http2;
    server_name example.local www.example.local;

    ssl_certificate     /etc/ssl/localcerts/example.
        local.crt;
    ssl_certificate_key /etc/ssl/localcerts/example.
        local.key;

    add_header X-Content-Type-Options "nosniff" always
        ;
    add_header Strict-Transport-Security "max-age
        =31536000; includeSubDomains" always;

    location / {
        try_files $uri $uri/ =404;
    }
}
```

# Варианты HTTPS/TLS

**Варианты сертификатов и TLS:**

- **Let's Encrypt** — бесплатные, автоматические (ACME). Срок 90 дней с автообновлением.
- **Самоподписанные** — для локальной разработки, не доверяются браузерами.
- **Коммерческие (CA)** — платные сертификаты от DigiCert, Sectigo и т.д.
- **Wildcard** — покрывают все поддомены: `*.example.com`.
- **Multi-domain (SAN)** — несколько доменов в одном сертификате.
- **TLS версии:** TLS 1.2 (широко), TLS 1.3 (рекомендуется). TLS 1.0/1.1 — устаревшие.

# Файловая структура

**Краткая файловая структура NGINX:**

- `/etc/nginx/nginx.conf` — главный конфиг;
- `/etc/nginx/conf.d/` — дополнительные файлы конфигурации (обычно `*.conf`);
- `/etc/nginx/sites-available/` — доступные виртуальные хосты;
- `/etc/nginx/sites-enabled/` — символические ссылки включённых сайтов;
- `/var/log/nginx/` — `access.log` и `error.log`;
- `/var/www/` — корни сайтов;
- `/etc/ssl/` или `/etc/ssl/localcerts/` — TLS-сертификаты и ключи.

# Ограничение частоты запросов (Rate Limiting)

Определение зоны в блоке `http`:

```
limit_req_zone $binary_remote_addr zone=req_limit:10m
    rate=5r/s;
```

Применение:

```
location /api/ {
    limit_req zone=req_limit burst=10 nodelay;
    proxy_pass http://app_upstream;
}
```

Параметры:

- `burst` — буфер сверх лимита;
- `nodelay` — отклонять вместо задержки.

# Рекомендации по безопасности

- `server tokens off;` — скрыть версию NGINX;
- Необходимо выставлять файлам соответствующие флаги доступа(rwx);
- включить HTTPS используя HSTS;
- использовать заголовки безопасности в HTTP запросах и ответах (CSP, X-Frame-Options и т.д.);
- резервные копии конфигураций перед изменениями;
- подумать о WAF (ModSecurity) и rate limiting;
- регулярные обновления пакетов.

# Логи и отладка

Просмотр логов:

```
sudo tail -f /var/log/nginx/access.log /var/log/nginx/
    error.log
sudo journalctl -u nginx -e
```

Полезные команды:

- `nginx -t` — проверка конфигурации;
- `nginx -T` — вывод всей конфигурации;
- `nginx -V` — параметры сборки и модули.

# Upstream — балансировка нагрузки

Для многосерверной архитектуры

Пример блока `upstream` (round-robin по умолчанию):

```
upstream myapp1 {
    server 10.0.0.11:8000;
    server 10.0.0.12:8000;
    server 10.0.0.13:8000;
}
```

Использование:

```
server {
    listen 80;
    location / {
        proxy_pass http://myapp1;
    }
}
```

# Proxy — основные директивы (введение)

Когда NGINX работает как обратный прокси, он принимает запросы от клиента и перенаправляет их на бэкенд (app server). Основные директивы:

- `proxy pass` — адрес бэкенда;
- `proxy set header` — управление HTTP-заголовками к бэкенду;
- `proxy buffering`, `proxy buffers` — поведение буферизации;
- `proxy read timeout`, `proxy connect timeout` — таймауты соединения.

# Proxy — примеры и пояснения

**Базовый пример:**

```
location / {
    proxy_pass http://myapp1;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For
        $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
}
```

# Proxy — примеры и пояснения

**Пояснения:**

- `proxy pass` — может быть с именем upstream (`http://myapp1`) или конкретным URL (`http://127.0.0.1:8000`).
- `proxy set header Host $host` — передаёт исходный Host, полезно для виртуальных хостов на бэкенде.
- `X-Forwarded-For` & `X-Real-IP` — сохраняют реальный IP клиента для логов и логики приложения.
- `X-Forwarded-Proto` — полезен для определения, был ли запрос через HTTPS.

# Proxy — примеры и пояснения

**Буферизация и производительность:**

```
proxy_buffering on;
proxy_buffers 8 16k;
proxy_buffer_size 32k;
proxy_busy_buffers_size 64k;
```

- Отключение буферизации (`proxy buffering off`) полезно при потоковой передаче (WebSockets, SSE).
- Корректная настройка буферов уменьшает количество системных вызовов и нагрузки на бэкенд.

# Proxy — примеры и пояснения

**Таймауты и соединения:**

```
proxy_connect_timeout 5s;
proxy_send_timeout 30s;
proxy_read_timeout 30s;
send_timeout 30s;
```

- `proxy connect timeout` — время ожидания установки соединения к бэкенду.
- `proxy read timeout` — время ожидания ответа от бэкенда.

# Proxy — примеры и пояснения

**WebSocket и HTTP upgrade:**

```
proxy_set_header Upgrade $http_upgrade;
proxy_set_header Connection $connection_upgrade;
map $http_upgrade $connection_upgrade {
    default upgrade;
    ''      close;
}
```

Это позволяет проксировать WebSocket соединения корректно.

# Proxy — примеры и пояснения

**SSL на бэкенде:**

```
proxy_pass https://secure_backend;
proxy_ssl_server_name on;
proxy_ssl_name backend.example.com;
proxy_ssl_trusted_certificate /etc/ssl/certs/ca-
    certificates.crt;
```

Позволяет устанавливать TLS-соединение с бэкендом.

# Кэширование прокси

**Кэширование прокси (proxy cache) — кратко:**

- Настраивается зона кэша в `http` с `proxy cache path`;
- В location: `proxy cache mycache;` & правила `proxy cache valid`.

# Заключение

NGINX — гибкий и мощный инструмент:

- простая установка и конфигурация;
- поддержка HTTPS и HTTP/2;
- мощные возможности проксирования и балансировки;
- богатые инструменты для логирования и диагностики;
- подходит и для локальной разработки, и для продакшена.