NGINX

NA PRÁTICA

DO BÁSICO À ARQUITETURA DE SISTEMAS WEB RESILIENTES

Alexandre Ladeira

Introdução ao NGINX

O que é o NGINX?

NGINX (pronuncia-se "engine-x") é um servidor web de alta performance, também usado como proxy reverso, balanceador de carga e cache HTTP. Criado em 2004 pelo russo Igor Sysoev, o NGINX foi projetado para lidar com milhares de conexões simultâneas com baixo uso de memória, o que o tornou popular em sites de grande tráfego.

Porque usar NGINX?

Ao contrário de servidores web tradicionais, como o Apache, que criam um novo processo para cada conexão, o NGINX utiliza uma arquitetura baseada em eventos e assincronismo. Isso permite alta escalabilidade, eficiência e baixa latência.

Objetivo deste ebook

Aqui você aprenderá, passo a passo, desde a instalação básica até arquiteturas avançadas e resilientes usando NGINX, incluindo configurações reais e dicas para manter seu sistema web rápido, seguro e disponível.

01

INSTALANDO E CONFIGURANDO

Instalando e Configurando o NGINX

Instalando o NGINX no Linux (Ubuntu/Debian)

1. Atualize a lista de pacotes:

```
sudo apt update
```

2. Instale o NGINX

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

3. Verificando se o serviço esta ativo

```
sudo systemctl status nginx
```

4. Para iniciar/parar/reiniciar:

```
sudo systemctl start nginx
sudo systemctl stop nginx
sudo systemctl restart nginx
```

Instalando e Configurando o NGINX

Instalando no Windows

Para Windows, você pode usar o NGINX oficial disponível em https://nginx.org/en/download.html. Basta baixar o zip, extrair e rodar o nginx.exe via prompt de comando.

Rodando NGINX com Docker

Se preferir usar Docker, execute:

```
1  docker run --name nginx-test -p 8080:80 -d nginx
```

Realizada a instalação, no ambiente escolhido, acesse: http://localhost:8080.

Estrutura básica do arquivo de configuração

O arquivo principal fica geralmente em /etc/nginx/nginx.conf (Linux) ou na pasta onde extraiu o Windows.
Para facilitar, o mais comum é configurar sites individuais na pasta /etc/nginx/sites-available e criar links simbólicos para /etc/nginx/sites-enabled.

Exemplo prático (ambiente Linux):

Configuração básica para rodar no localhost

1. Crie o arquivo /etc/nginx/sites-available/meusite com o conteúdo:

```
server {
    listen 80;
    server_name localhost;

    root /var/www/meusite;
    index index.html;

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

```

2. Crie a pasta e um arquivo index simples:

```
sudo mkdir -p /var/www/meusite
echo "<h1>Olá, NGINX no localhost!</h1>" | sudo tee /var/www/meusite/index.html
```

Exemplo prático (Linux) continuação...

3. Ative o site, criando o link simbólico:

```
sudo ln -s /etc/nginx/sites-available/meusite /etc/nginx/sites-enabled/
```

4. Teste a configuração:

```
sudo nginx -t
```

5. Reinicie o NGINX para aplicar:

```
sudo systemctl restart nginx
```

6. Acesse no navegador: http://localhost

Você verá a mensagem “Olá, NGINX no localhost!”

02

USANDO NGINX COMO PROXY REVERSO

NGINX como Proxy Reverso

O que é Proxy Reverso?

Proxy reverso é um servidor que recebe requisições do cliente e as encaminha para um ou mais servidores backend. Ele atua como intermediário, escondendo a arquitetura real dos servidores que respondem às requisições, além de adicionar camadas de segurança, cache e balanceamento.

Para que usar?

- Encapsular servidores backend
- Controlar acesso e autenticação
- Fazer SSL offloading (descriptografar HTTPS)
- Cachear respostas para melhorar performance
- Monitorar e limitar requisições

A seguir, será apresentado um exemplo de uma servidor rodando uma aplicação em Node.js e o NGINX recebendo as requisições e repassando para o servidor de aplicação.

Exemplo prático (ambiente Linux):

Exemplo básico de proxy reverso para uma aplicação Node.js rodando local na porta 3000

1. Configuração do arquivo .conf

```
server {
    listen 80;
    server_name localhost;

    location / {
        proxy_pass http://localhost:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}
```

2. Passos para testar:

- Rode uma aplicação simples na porta 3000 (Node.js, Python, etc).
- Configure o NGINX com o arquivo acima
- Acesse http://localhost — a requisição vai passar pelo NGINX para o backend.

03

NGINX COMO API GATEWAY

NGINX como API Gateway

O que é API Gateway?

É um ponto único de entrada para múltiplas APIs ou microserviços, fazendo roteamento, autenticação, limitação de taxa, logging, entre outras funções. O NGINX pode ser usado para isso, com configurações avançadas.

Recursos comuns usados como API Gateway no NGINX:

- Roteamento baseado em URL ou cabeçalhos
- Limitação de requisição (rate limiting)
- Autenticação básica ou via tokens
- Transformação e manipulação de headers
- Cache para melhorar performance

A seguir, será apresentado um exemplo simples de roteamento para múltiplos serviços.

Exemplo prático (ambiente Linux) API Gateway:

Exemplo de roteamento para múltiplos serviços:

1. Configuração do arquivo .conf

```
server {
    listen 80;
    server_name api.meusistema.local;

    location /usuario/ {
        proxy_pass http://localhost:3001/;
    }

    location /pedido/ {
        proxy_pass http://localhost:3002/;
    }

    location /produto/ {
        proxy_pass http://localhost:3003/;
    }
}
```

Explicação: Requisições para /usuario/* vão para a aplicação rodando na porta 3001, /pedido/* para a 3002, e assim por diante.

04

NGINX COMO LOAD BALANCER

NGINX como Load Balancer

O que é Load Balancer?

Balanceador de carga distribui o tráfego entre múltiplos servidores backend para garantir escalabilidade e alta disponibilidade..

Tipos de balanceamento comuns no NGINX:

- Round Robin: padrão, distribui em sequência
- Least Connections: envia para o servidor com menos conexões ativas
- IP Hash: direciona sempre para o mesmo backend com base no IP do cliente

A seguir, será apresentado um exemplos simples de load balancer com arquivos .conf para cada tipode balanceamento.

Exemplo prático (ambiente Linux) Load Balancer:

Exemplo de configuração básica de load balancing com round robin:

1. Configuração do arquivo .conf

```
upstream meuservidores {
    server localhost:3001;
    server localhost:3002;
    server localhost:3003;
}

server {
    listen 80;
    server_name localhost;

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

Exemplo prático (ambiente Linux) Load Balancer (continuação...) :

Exemplo com least connections:

1. Configuração do arquivo .conf

```
upstream meuservidores {
    least_conn;
    server localhost:3001;
    server localhost:3002;
    server localhost:3003;
}
```

Exemplo com IP Hash (útil para sessões persistentes):

1. Configuração do arquivo .conf

```
upstream meuservidores {
    ip_hash;
    server localhost:3001;
    server localhost:3002;
    server localhost:3003;
}
```

Testando o Load Balancer

Tenha 3 aplicações rodando nas portas 3001, 3002 e 3003 que retornam um texto identificando o servidor (ex: "Servidor 1").
Acesse http://localhost várias vezes e veja o balanceamento.

05

LABORATÓRIO PRÁTICO COM NGINX

Passo a passo prático: Ambiente local com NGINX proxy reverso, API Gateway e Load Balancer

1. Criar apps backend simples em Node.js

Vamos criar 3 apps Node.js que escutam em portas diferentes e respondem com uma mensagem que identifica o servidor.

Código comum para os 3 apps (server.js):

```
const http = require('http');

const port = process.env.PORT || 3000;
const serverName = process.env.SERVER_NAME || 'Servidor';

const server = http.createServer((req, res) => {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end(`Olá do ${serverName} na porta ${port}\n`);
});

server.listen(port, () => {
  console.log(`${serverName} rodando na porta ${port}`);
});
```

Passo a passo prático: Ambiente local com NGINX proxy reverso, API Gateway e Load Balancer

Criar as pastas dos servidores e rodar:

No terminal:

```
mkdir backend1 backend2 backend3
```

Crie o arquivo server.js acima em cada pasta (copiar e colar).

Depois, rode os 3 servidores com variáveis diferentes:

```
# Backend 1
cd backend1
PORT=3001 SERVER_NAME="Servidor 1" node server.js

# Em outro terminal, Backend 2
cd backend2
PORT=3002 SERVER_NAME="Servidor 2" node server.js

# Em outro terminal, Backend 3
cd backend3
PORT=3003 SERVER_NAME="Servidor 3" node server.js
```

Passo a passo prático: Ambiente local com NGINX proxy reverso, API Gateway e Load Balancer

2. Configurar NGINX para proxy reverso, API Gateway e Load Balancer

Crie um arquivo de configuração para NGINX (ex: /etc/nginx/sites-available/meusite):

```
http {
    upstream meuservidores {
        least_conn;
        server localhost:3001;
        server localhost:3002;
        server localhost:3003;
    }

    server {
        listen 80;
        server_name localhost;

        # Proxy Reverso básico (para backend 1)
        location /proxy/ {
            proxy_pass http://localhost:3001/;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For
$proxy_add_x_forwarded_for;
        }
```

Passo a passo prático: Ambiente local com NGINX proxy reverso, API Gateway e Load Balancer

Continuação do arquivo de configuração:

```
        # API Gateway simulando roteamento por caminho
        location /api/usuario/ {
            proxy_pass http://localhost:3001/;
        }

        location /api/pedido/ {
            proxy_pass http://localhost:3002/;
        }

        location /api/produto/ {
            proxy_pass http://localhost:3003/;
        }

        # Load Balancer usando upstream
        location /loadbalance/ {
            proxy_pass http://meuservidores/;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For
$proxy_add_x_forwarded_for;
        }
    }
}

```

Passo a passo prático: Ambiente local com NGINX proxy reverso, API Gateway e Load Balancer

3. Ativar a configuração no NGINX

```
1  sudo ln -s /etc/nginx/sites-available/meusite /etc/nginx/
   sites-enabled/
2  sudo nginx -t
3  sudo systemctl restart nginx
```

4. Testando no navegador ou curl

- Proxy Reverso:
  Acesse: http://localhost/proxy/
  Deve retornar resposta do Servidor 1.

- API Gateway:
  Acesse:
    http://localhost/api/usuario/ → responde do Servidor 1
    http://localhost/api/pedido/ → responde do Servidor 2
    http://localhost/api/produto/ → responde do Servidor 3

- Load Balancer:
  Acesse várias vezes: http://localhost/loadbalance/
  Vai alternar entre os 3 servidores (balanceamento least_conn).

Passo a passo prático: Ambiente local com NGINX proxy reverso, API Gateway e Load Balancer

5. Dicas extras

- Para facilitar, use curl no terminal:

```
curl http://localhost/proxy/
curl http://localhost/api/usuario/
curl http://localhost/loadbalance/
```

- Para parar os servidores Node.js, use Ctrl+C no terminal onde rodaram.

AGRADECIMENTOS

OBRIGADO POR LER ATÉ AQUI

Esse Ebook foi gerado por IA, e diagramado por humano.
O passo a passo se encontra no meu Github

Esse conteúdo foi gerado com fins didáticos de construção, e foi realizado uma validação cuidadosa humana no conteúdo prático proposto.

https://github.com/Ladeiraalexandre/prompts-recipe-to-create-a-ebook/tree/main