

Microserviços na prática com a ajuda do Nginx

Realizaremos um deploy localmente, pondo em prática os conceitos de **API Gateway** e **Load Balancer** utilizando o **Web Server Nginx**, um dos mais populares do mercado, em paralelo com o **Apache**.

Primeiramente, crie duas aplicações básicas com Flask, ou utilize duas que você já possui. Configure uma para rodar na **porta 5001** do seu computador e outra na **porta 5002**, e vamos chamá-las de aplicação de **1 e 2**, respectivamente.

Agora, antes de instalar o **Nginx** tente rodar uma aplicação dessas usando a **porta 80** e verifique que não conseguirá. Isso acontece porque a **porta 80** por padrão é autorizada somente para usuários com permissões de **super usuário** do Unix. A **porta 80** é a padrão para **HTTP**, e a **porta 443** é a padrão para **SSL** ou **HTTPS**. Localmente você só consegue usar **HTTP** e a **porta 80**. **SSL/HTTPS** e a **porta 443** somente em **ambiente de produção**.

Para instalar o Nginx no Ubuntu:

```
$ sudo apt install nginx
```

Após instalado, acesse a **porta 80** do seu computador pelo navegador digitando **localhost:80** ou somente **localhost**. Essa informação é importante: O **Nginx** utiliza a **porta 80** por padrão. Como ele foi instalado usando **sudo** (permissão de administrador), é possível ter acesso a ela, e a partir de agora sempre que o sistema iniciar, ele vai iniciar junto e ocupar essa porta.

Caso não apareça nada no **localhost:80**, rode o seguinte comando no terminal:

```
$ sudo service nginx start
```

Resultado esperado:

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

API Gateway

O **Nginx** consegue fazer quase tudo o que precisamos em ambientes complexos de produção. Vamos ver como fazer um **API Gateway**:

Dica!

Você pode utilizar o comando `nginx -t` para testar as configurações.

Edita o arquivo de configurações padrão do **Nginx** usando o seguinte comando:

```
$ sudo vim /etc/nginx/sites-enabled/default
```

Importante!

Como o diretório está localizado no `/etc`, é necessário acessá-la com o comando `sudo`.

Se você não tem o **VIM**, instale-o usando:

```
$ sudo apt install vim
```

Continuando com a edição do arquivo /etc/nginx/sites-enabled/default :

Originalmente ele apresenta o seguinte conteúdo:

```
##  
# You should look at the following URL's in order to grasp a solid u-  
# of Nginx configuration files in order to fully unleash the power o-  
# https://www.nginx.com/resources/wiki/start/  
# https://www.nginx.com/resources/wiki/start/topics/tutorials/config_  
# https://wiki.debian.org/Nginx/DirectoryStructure  
  
#  
# In most cases, administrators will remove this file from sites-ena-  
# leave it as reference inside of sites-available where it will cont-  
# updated by the nginx packaging team.  
  
#  
# This file will automatically load configuration files provided by  
# applications, such as Drupal or Wordpress. These applications will  
# available underneath a path with that package name, such as /drupa-  
#  
# Please see /usr/share/doc/nginx-doc/examples/ for more detailed ex-  
##  
  
# Default server configuration  
#  
server {  
    listen 80 default_server;  
    listen [::]:80 default_server;  
  
    # SSL configuration  
    #  
    # listen 443 ssl default_server;  
    # listen [::]:443 ssl default_server;  
    #  
    # Note: You should disable gzip for SSL traffic.  
    # See: https://bugs.debian.org/773332  
    #  
    # Read up on ssl_ciphers to ensure a secure configuration.  
    # See: https://bugs.debian.org/765782
```

```
#  
# Self signed certs generated by the ssl-cert package  
# Don't use them in a production server!  
#  
# include snippets/snakeoil.conf;  
  
root /var/www/html;  
  
# Add index.php to the list if you are using PHP  
index index.html index.htm index.nginx-debian.html;  
  
server_name _;  
  
location / {  
    # First attempt to serve request as file, then  
    # as directory, then fall back to displaying a 404.  
    try_files $uri $uri/ =404;  
}  
  
# pass PHP scripts to FastCGI server  
#  
#location ~ \.php$ {  
#    include snippets/fastcgi-php.conf;  
#  
#    # With php-fpm (or other unix sockets):  
#    fastcgi_pass unix:/var/run/php/php7.4-fpm.sock;  
#    # With php-cgi (or other tcp sockets):  
#    fastcgi_pass 127.0.0.1:9000;  
#}  
  
# deny access to .htaccess files, if Apache's document root  
# concurs with nginx's one  
#  
#location ~ /\.ht {  
#    deny all;  
#}  
}  
  
# Virtual Host configuration for example.com
```

```
#  
# You can move that to a different file under sites-available/ and s  
# to sites-enabled/ to enable it.  
#  
#server {  
#        listen 80;  
#        listen [::]:80;  
#  
#        server_name example.com;  
#  
#        root /var/www/example.com;  
#        index index.html;  
#  
#        location / {  
#            try_files $uri $uri/ =404;  
#        }  
#}
```

Importante!

Nós vamos apagar todo o conteúdo acima (**conteúdo de exemplo / conteúdo original**) e deixar somente o básico, inclua os comentários se te ajudarem. Seu arquivo ficará assim:

```
# Declarando um servidor  
server {  
  
    # Esse servidor escuta na porta 80 da nossa maquina  
listen 80 default_server;  
listen [::]:80 default_server;  
  
    # Um nome qualquer para o servidor, em producao se tiver um .  
server_name _;  
  
    # Aqui vamos criar rotas para esse servidor escutando na porta 80  
location / {
```

```
}
```

```
}
```

Para fazermos um **redirecionamento** para cada serviço, conforme as requisições chegam em nossa máquina, vamos usar o comando `proxy_pass` :

O arquivo ficará assim:

```
# Declarando um servidor
server {

    # Esse servidor escuta na porta 80 da nossa maquina
    listen 80 default_server;
    listen [::]:80 default_server;

    # Um nome qualquer para o servidor, em producao se tiver um domini
    server_name _;

    # Aqui vamos criar rotas para esse servidor escutando na porta 80
    # Vamos utilizar a raiz "/" seguido do nome do nosso servico para
    location /app1 {
        # Utilize 0.0.0.0 ao inves de 127.0.0.1 ou localhost, evita
        # Utilize ; ao final, sintaxe parecida com JS.
        proxy_pass http://0.0.0.0:5001/;

    }

    location /app2 {
        proxy_pass http://0.0.0.0:5002/;

    }

    # Aqui pode criar quantos mais quiser nesse mesmo padrao para te
}
```

Para que as alterações no arquivo façam efeito, é necessário **reiniciar o serviço Nginx** usando:

```
$ sudo service nginx restart
```

Aviso!

Caso funcione, não deve retornar nenhuma mensagem, se retornar mensagem de erro abra o arquivo novamente e revise.

Teste abrindo **dois terminais**, rode cada aplicação na respectiva porta e acesse `localhost/app1` ou `localhost/app2`. Veja que as requisições chegam em cada aplicação alternadamente. Pronto, temos um exemplo de dois serviços (ou microsserviços) rodando atrás de um **API Gateway**.

Load Balancer

Agora, vamos criar um **balanceador de carga** entre aplicações. Para fazer mais sentido, vamos rodar uma mesma aplicação duas vezes, em portas diferentes.

Utilize um mesmo app e rode primeiro usando a **porta 5001** e depois a **porta 5002**. Ambos devem rodar simultaneamente e responder nas respectivas portas.

Agora vamos editar o arquivo do Nginx `/etc/nginx/sites-enabled/default` para criar nosso **Load Balancer**, adicione um novo `location` que vai responder na raiz:

```
# responde na raiz ou seja http://localhost/
location / {
    proxy_pass http://backend;
}
```

Repare que estamos encaminhando nossos requests para `backend` ao invés de um endereço específico. Agora vamos criar a definição do que é o **app chamado backend**:

Edite o arquivo `/etc/nginx/nginx.conf`

```
$ sudo vim /etc/nginx/nginx.conf
```

Na seção HTTP crie um bloco iniciado por `upstream` como abaixo:

```
upstream backend {  
    # Adicionando a aplicacao da porta 5001 ao nosso grupo backend  
    server 0.0.0.0:5001 weight=1;  
    # adicionando a aplicacap da porta 5002 ao nosso grupo porem  
    server 0.0.0.0:5002 weight=2;  
  
    # Aqui voce poderia continuar adicionando outros servicos para rotear  
}
```

Aviso!

Esse bloco deve estar contido na seção `http{}` do arquivo `/etc/nginx/nginx.conf`, de preferência no final dessa seção, antes de fechar o `}`.

Reinic peace o **Nginx** (`sudo service nginx restart`) e teste usando o navegador acessando `http://localhost/` e observe o comportamento das aplicações nos respectivos terminais. Nessa configuração, uma mesma rota é respondida por diferentes aplicações alternadamente e como demos **peso 2** para a da **porta 5002**, ela responde 2x e depois passa para a **porta 5001**.

Experimente diferentes configurações e reflita sobre as vantagens de ter um servidor robusto como o **Nginx** à frente das nossas aplicações em produção.

VIM Dicas

VI é um clássico editor de texto para o shell, **VIM** é sua versão melhorada (VI Improved ou VI Melhorado)

Se você não está familiarizado com o editor **VIM**, seguem algumas dicas:

- Alguns comandos ativam somente pressionando a tecla desejada, outros precisam ser passados com `:` antes, aparecendo na barra inferior o comando sendo digitado;
- Após abrir um arquivo para editar, tecle `i`, e então você estará no modo **inserção de caracteres**;
- Repare na barra inferior que agora está no **modo de inserção**;
- Insira os caracteres navegando pelos arquivos através das setas do teclado;
- Após concluir a edição, pressione **ESC** para sair do modo **INSERT** e digite `:wq` para executar os comandos **write** e depois **quit**, confirme pressionando **ENTER**;
- Caso cometa algum erro, pressione **ESC** para sair de qualquer modo. Digite `:` e depois `q!` para sair (**quit**) sem confirmação para salvar as alterações;
- O `!` indica que você tem certeza e não precisa de confirmação;
- Para deletar uma linha inteira pressione a tecla `d` duas vezes seguidas;
- Consulte manuais sobre uso de **VIM** e pratique bastante. Em ambientes de testes e de produção, os servidores não possuem interface gráfica e todas as operações são feitas através de ferramentas de **CLI**.

Referências!

[Deploying NGINX as an API Gateway - Nginx](#)

[HTTP Load Balancing - Nginx](#)