Propozycja

1. Prosta strona w nginx
2. Serwowanie obrazków z nginx po podaniu ścieżki/regex
3. Odwrotne proxy do prostej aplikacji w Javie (Docker?)
5. Rozbudowa do load balancera dla kilku (np. 2) aplikacji
6. Rozbudowa o obsługę SSL (self-signed)

Instalacja

Instalacja

- sudo apt update
- sudo apt install nginx

Firewall

sudo ufw app list

```
Output

Available applications:
  Nginx Full
  Nginx HTTP
  Nginx HTTPS
    OpenSSH
```

Firewall

sudo ufw allow 'Nginx HTTP'

Pierwsza strona

- Sprawdzamy firewall (i ew. inne reguły)
- Zajrzyjmy do /var/www/html/

Podstawowe użycia

Zajrzyjmy do configa

- /etc/nginx/nginx.conf

Prosty serwis www

```
server {
        location / {
                root /data/www;
        }

        location /images/ {
                root /data/images;
        }
}
```

Prosty serwis www – własne 404

```
server {
        ...
        error_page 404 /error404.html;
        ...
}
```

Dodajemy proxy

```
server {
        listen 8080;
        root /data/up1;

        location / {
        }
}
```

Dodajemy katalog /data/up1 wraz z index.html

Wykorzystujemy nowe proxy

```
server {
        location / {
                proxy_pass http://localhost:8080;
        }

        location /images/ {
                root /data/images;
        }
}
```

Sprawdzamy wyrażenia regularne

```
server {
        location / {
                proxy_pass http://localhost:8080/;
        }

        location ~ \.(gif | jpg | png)$ {
                root /data/images;
        }
}
```

Zmieniamy proxy na load balancer

```
http {
        upstream myapp1 {
                server srv1.example.com;
                server srv2.example.com;
                server srv3.example.com;
        }
        server {
                listen 80;
                location / {
                        proxy_pass http://myapp1;
                }
        }
}
```

FastCGI proxy (np. dla PHP)

```
server {
        location / {
                fastcgi_pass localhost:9000;
                fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
                fastcgi_param QUERY_STRING $query_string;
        }

        location ~ \.(gif|jpg|png)$ {
                root /data/images;
        }
}
```

Wiele serwerów wirtualnych

```
server {
        listen *:80 default_server;
        server_name example.com;
        return 200 "Hello from example.com";
}
server {
        listen *:80;
        server_name foo.com;
        return 200 "Hello from foo.com";
}
server {
        listen *:81;
        server_name bar.com;
        return 200 "Hello from bar.com";
}
```

Kolejność:

Request to foo.com:80 => "Hello from foo.com"

Request to www.foo.com:80 => "Hello from example.com"

Request to bar.com:80 => "Hello from example.com"

Request to bar.com:81 => "Hello from bar.com"

Request to foo.com:81 => "Hello from bar.com"

Przekierowania (rewrite)

```
server {
        rewrite ^ /foobar;

        location /foobar {
                rewrite ^ /foo;
                rewrite ^ /bar;
        }
        ...
}
```

SSL

```
server {
        listen 443 ssl;
        server_name www.example.com;
        ssl_certificate www.example.com.crt;
        ssl_certificate_key www.example.com.key;
        ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
        ssl_ciphers HIGH:!aNULL:!MD5;
        ...
}
```

SSL - przekierowanie

```
server {
        listen 80 default_server;
        server_name www.example.com;
        return 301 https://www.example.com$request_uri;
}
```

301 – przekierowanie stałe

302 – przekierowanie tymczasowe

GIXY

https://github.com/yandex/gixy

Cache / nginx

Proxy cache

```
proxy_cache_path /path/to/cache levels=1:2 keys_zone=my_cache:10m max_size=10g
                 inactive=60m use_temp_path=off;

server {
    # ...
    location / {
        proxy_cache my_cache;
        proxy_pass http://my_upstream;
    }
  }
```

Splitting the Cache Across Multiple Hard Drives

```
proxy_cache_path /path/to/hdd1 levels=1:2 keys_zone=my_cache_hdd1:10m
                 max_size=10g inactive=60m use_temp_path=off;
proxy_cache_path /path/to/hdd2 levels=1:2 keys_zone=my_cache_hdd2:10m
                 max_size=10g inactive=60m use_temp_path=off;

split_clients $request_uri $my_cache {
              50%          “my_cache_hdd1”;
              50%          “my_cache_hdd2”;
}
server {
    # ...
    location / {
        proxy_cache $my_cache;
        proxy_pass http://my_upstream;
    }
  }
```

HTTP Load Balancing / nginx

Upstream

```
http {
    upstream backend {
        server backend1.example.com weight=5;
        server backend2.example.com;
        server 192.0.0.1 backup;
    }
}
```

```
server {
    location / {
        proxy_pass http://backend;
    }
}
```

Load-Balancing Method

Domyślnie : Round Robin

algorytm karuzelowy

Least Connections

```
upstream backend {
    least_conn;
    server backend1.example.com;
    server backend2.example.com;
}
```

Żądanie jest wysyłane do serwera z najmniejszą liczbą aktywnych połączeń

IP Hash

```
upstream backend {
    ip_hash;
    server backend1.example.com;
    server backend2.example.com;
}
```

Serwer, do którego wysyłane jest żądanie, określany jest na podstawie adresu IP klienta. W takim przypadku do obliczenia wartości skrótu używane są pierwsze trzy oktety adresu IPv4 lub cały adres IPv6.

Generic Hash

Serwer, do którego wysyłane jest żądanie, jest określany na podstawie klucza zdefiniowanego przez użytkownika, który może być ciągiem tekstowym, zmienną lub kombinacją.

```
upstream backend {
    hash $request_uri consistent;
    server backend1.example.com;
    server backend2.example.com;
}
```

Least Time (NGINX Plus only)

Dla każdego żądania NGINX Plus wybiera serwer z najniższym średnim opóźnieniem i najmniejszą liczbą aktywnych połączeń, przy czym najniższe średnie opóźnienie jest obliczane na podstawie tego, który z poniższych parametrów jest uwzględniony w dyrektywie less_time:

Server Weights

```
upstream backend {
    server backend1.example.com weight=5;
    server backend2.example.com;
    server 192.0.0.1 backup;
}
```

Server Slow-Start

```
upstream backend {
    server backend1.example.com slow_start=30s;
    server backend2.example.com;
    server 192.0.0.1 backup;
}
```

Limiting the Number of Connections

```
upstream backend {
    server backend1.example.com max_conns=3;
    server backend2.example.com;
    queue 100 timeout=70;
}
```

Configuring Health Checks

https://docs.nginx.com/nginx/admin-guide/load-balancer/http-health-check/