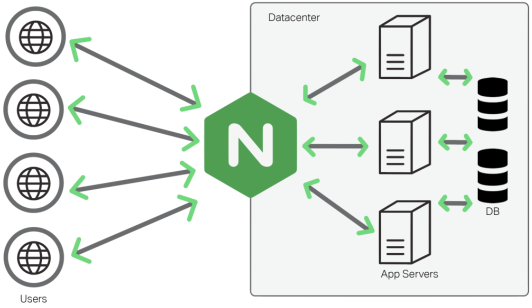
A Byte of Nginx

Sheng Yuan

Contents

Introduce



Nginx is a web server which can also be used as a reverse proxy, load balancer, mail proxy and HTTP cache[3].

Basics

Basic usage

```
$ nginx
$ nginx -s reload
$ nginx -s stop
$ nginx -s quit
$ nginx -s reopen
$ nginx -c /conf/nginx.conf
$ nginx -c /conf/nginx.conf -t


$ service nginx start
$ service nginx stop
$ service nginx restart

$ kill -QUIT $( cat /var/run/nginx.pid )
```

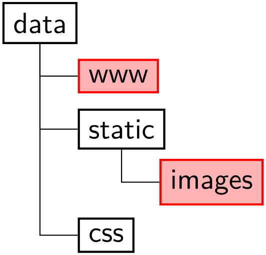
Config Syntax

- simple directive the name and parameters separated by spaces and ends with a semicolon (;)
- block directive the same structure as a simple directive, but instead of the semicolon it ends with a set of additional instructions surrounded by braces ({ and })
- The lines starting with # sign are comments.
- Configurations can be split into multiple files.

Web Server

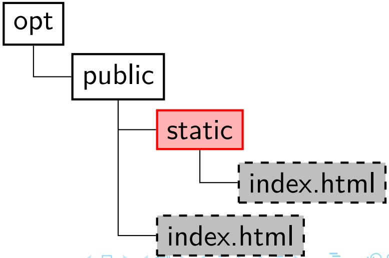
Simple Web Server

```
http {
  server {
    location / {
      root /data/www;
    }
    location /images/ {
      root /data/static;
    }
  }
}
```



Web Server with alias

```
http {
  server {
    location /static {
      alias /opt/public;
    }
    location /static {
      root /opt/public;
    }
  }
}
```



Proxy Server

Proxy Server

```
server {
  location /debug {
    proxy_pass http://127.0.0.1:8000/debug;
  }

  location /static {
    root /data/public;
  }
}
```

CORS

```
server {
  location /api/v2 {
    proxy_http_version 1.1;
    add_header Access-Control-Allow-Origin *.mydomain.com;
    proxy_pass http://127.0.0.1:8080/api/v2;
  }
}
```

HTTP Load Balance

Problem

You need to distribute load between two or more HTTP servers.

Solution

```
upstream backend {
  server 10.10.12.45:80 weight=1;
  server app.example.com:80 weight=2;

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

TCP Load Balance

Problem

You need to distribute load between two or more TCP servers.

Solution

```
stream {
  upstream mysql_read {
    server read1.example.com:3306 weight=5;
    server read1.example.com:3306;
    server 10.10.12.34:3306 backup;
  }
  server {
    listen 3306;
    proxy_pass mysql_read;
  }
}
```

Load Balancing methods

Problem

You need to choose a LB method.

Solution

- Round robin Requests are distributed evenly across the servers, with server weights taken into consideration. This method is used by default (there is no directive for enabling it)
- Least connections A request is sent to the server with the least number of active connections, again with server weights taken into consideration.
- IP hash The server to which a request is sent is determined from the client IP address. In this case, either the first three octets of the IPv4 address or the whole IPv6 address are used to calculate the hash value. The method guarantees that requests from the same address get to the same server unless it is not available.
- Least time Only supported in Nginx plus.

Massively Scalable Content Caching

Problem

You need to cache content and need to define where the cache is stored.

Solution

```
proxy_cache_path /var/nginx/cache

  keys_zone=CACHE:60m levels=1:2
  inactive=3h
  max_size=20g;

proxy_cache CACHE;
```

Connection Limiting

- limit_conn and limit_conn_zone – Limit the number of client connections NGINX accepts, for example from a single IP address. Setting them can help prevent individual clients from opening too many connections and consuming more than their share of resources.
- limit_rate – Limits the rate at which responses are transmitted to a client, per connection (so clients that open multiple connections can consume this amount of bandwidth for each connection). Setting a limit can prevent the system from being overloaded by certain clients, ensuring more even quality of service for all clients.
- limit_req and limit_req_zone – Limit the rate of requests being processed by NGINX, which has the same benefits as setting limit_rate. They can also improve security, especially for login pages, by limiting the request rate to a value reasonable for human users but too slow for programs trying to overwhelm your application with requests (such as bots in a DDoS attack).
- max_conns parameter to the server directive in an upstream configuration block – Sets the maximum number of simultaneous connections accepted by a server in an upstream group. Imposing a limit can help prevent the upstream servers from being overloaded. Setting the value to 0 (zero, the default) means there is no limit.

Controlling Access

Problem

You need to control access based on the IP address of clients.

Solution

```
location /admin/ {
  deny 10.0.0.1;
  allow 10.0.0.0/20;
  allow 2001:0db8::/32;
  deny all;
}
```

The given location block allows access from any IPv4 address in 10.0.0.0/20 except 10.0.0.1, allows access from IPv6 addresses in the 2001:0db8::/32 subnet, and returns a 403 for requests originating from any other address.

Force Https

Problem

You want to implememt a HTTPS server and force HTTP to HTTPS.

Solution

```
server {
  listen 80;
  return 301 https://testai.tclking.com$request_uri;
}

server {
  listen 443;
  ssl on;
  ssl_certificate /usr/share/certificates/tcl/xxx.crt;
  ssl_certificate_key /usr/share/certificates/tcl/xxx.key;
  ssl_session_timeout 5m;
}

add_header Strict-Transport-Security max-age=31536000;
```

Configuring Logs

Problem

You want to add log for nginx, which is key to analysis when error occurs.

Solution

```
http {
  log_format geoproxy
    '[$time_local] $remote_addr '
    '$realip_remote_addr $remote_user '
    '$request_method $server_protocol '
    '$scheme $server_name $uri $status '
    '$request_time $body_bytes_sent '
    '$geoip_city_country_code3 $geoip_region '
    '$geoip_city" $http_x_forwarded_for '
    '$upstream_status $upstream_response_time '
    '"$http_referer" "$http_user_agent"';

  access_log /var/log/nginx/access.log geoproxy buffer=32k
  flush=1m;
  error_log /var/log/nginx/error.log main buffer=32k
  flush=1m;
}
```

The buffer parameter of the access_log directive denotes the size of a memory buffer that can be filled with log data before being written to disk. The flush parameter of the access_log directive sets the longest amount of time a log can remain in a buffer before being written to disk.

Performance Tuning

Keeping Connections Open to Clients

```
http {
  keepalive_requests 320;
  keepalive_timeout 300s;
}
```

keepalive_requests: The number of requests a client can make over a single keepalive connection. The default is 100, but a much higher value can be especially useful for testing with a load-generation tool, which generally sends a large number of requests from a single client.
keepalive_timeout: How long an idle keepalive connection remains open.

Keeping Connections Open Upstream

```
proxy_http_version 1.1;
proxy_set_header Connection "";
upstream backend {
  server 10.0.0.42;
  server 10.0.2.56;
  keepalive 32;
}
```

keepalive: The number of idle keepalive connections to an upstream server that remain open for each worker process. There is no default value.

Global Parameters

worker_processes
: This is the number of worker processes that will be started. These will handle all connections made by the clients. A good rule of thumb is to set this equal to the number of processor cores for CPU-bound loads and to multiply this number by 1.5 to 2 for I/O bound loads[1].

worker_connections
: This directive configures the maximum number of simultaneous connections that a worker process may have open. This includes, but is not limited to, client connections and connections to upstream servers. This is especially important on reverse proxy servers. some additional tuning may be required at the operating system level in order to reach this number of simultaneous connections.

Global Parameters

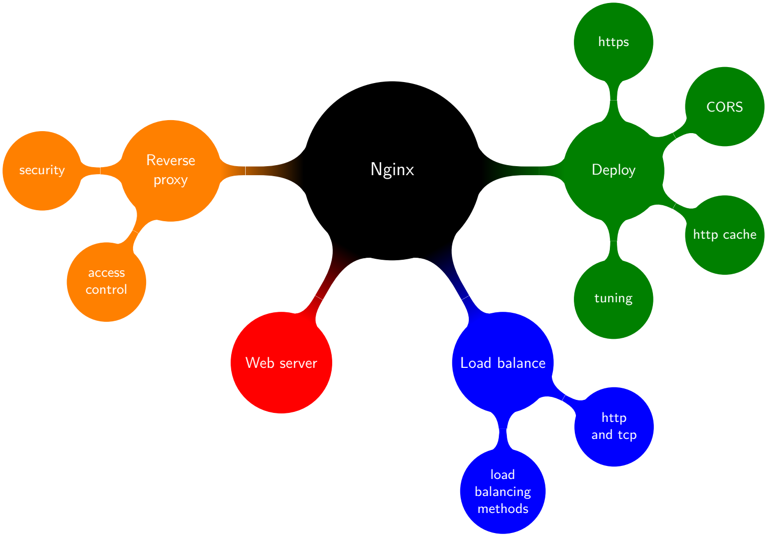
- worker_rlimit_nofile Changes the limit on the maximum of open files for worker process.
- client_max_body_size The maximum allowed size of the client request body, specified in the "Content-Length" of HTTP header field. This is important for file upload specially.

OS Tuning

HTTP is a TCP/IP-based system[4]; therefore, an administrator trying to extract the last drop of performance will not only optimize the web server, but also look at the TCP network stack.

- Raising the number of open file descriptors is a more common need.
- Check the kernel setting for net.core.somaxconn, which is the maximum number of connections that can be queued by the kernel for NGINX to process.
- Enable more ephemeral ports.

Summary


Nginx
Reverse proxy
security
access control
Web server
Load balance
http and tcp
load balancing methods
Deploy
https
CORS
http cache
tuning

References

Dimitri Aivaliotis. *Mastering Nginx*. Packt Publishing Ltd, 2016.

DeJonghe Derek. *Complete Nginx Cookbook*. O'Reilly Media, Inc., 2017.

Clement Nedelcu. *Nginx HTTP Server*. Packt Publishing Ltd, 2015.

Rahul Sharma. *Nginx high performance*. Packt Publishing Ltd, 2015.

Thank U