Chapter 1 命令行和基本配置

2020年1月28日 星期二 14:04

★1. Key Files, Commands and directories

/etc/nginx	Configure directory
/etc/nginx/nginx.conf	default configuration entry
/etc/nginx/conf.d	Sub-configure entries
/etc/nginx/sites-available	Deprecated
/etc/nginx/sites-enable	Deprecated
/var/log/nginx	log directory

nginx -h	show help menu
nginx -v	show Nginx Version
nginx -V	show Nginx Version, Modules and Build Information
nginx -t	test configuration files
nginx -s signal	to send a signal to nginx. Signal: 1. stop: stop Nginx immidiately 2. quit: quit Nginx after all

	2. quit: quit Nginx after all connection served 3. reload: reload configuration files 4. reopen: reopen log files

2. Serving Static Content

```
server {
    listen 80 default_server;
    server_name www.example.com;

    location / {
        root /usr/share/nginx/html;
        # alias /usr/share/nginx/html;
        index index.html index.htm;
    }
}
```

Chapter 2 负载平衡

2020年1月28日 星期二 14:35

1. Basic Load Balancing

★ 1.1 HTTP Load Balancing

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

use it as:

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

1.2 TCP Load Balancing

```
stream {
    upstream mysql_read {
        server read1.example.com:3306  weight=5;
```

```
            server read2.example.com:3306;
            server 10.10.12.34:3306        backup;
        }

        server {
            listen 3306;
            proxy_pass mysql_read;
        }
    }
```

1.3 UDP Load Balancing

```
stream {
    upstream ntp {
        server ntp1.example.com:123  weight=2;
        server ntp2.example.com:123;
    }

    server {
        listen 123 udp;
        proxy_pass ntp;
    }
}
```

If the service you're load balancing over requires multiple packets to be sent back and forth between client and server, you can specify the reuseport parameter

★2. Load Balancing Method

Nginx Load Balancing Methods Includes:

Round Robin	Default Solution, use weight as a peremeter.
least_conn	Connect to the least connection upstream
least_time	Available in Nginx Plux only
hash	The administrator defines a hash with the given text, variables of the request or runtime, or both. • consistent : to minimize the effect of redistribution
random	To random decide a upsteam • The optional two [method] parameter directs NGINX to ran- domly select two servers and then use the provided load- balancing method to balance between those two. By default the least_conn method is used if two is

	method is used if two is passed without a method.
ip_hash	Work only for HTTP, Hash uses the client IP Address. • weight : parameter into consideration when distributing the hash

1.3 Options for Nginx Plus

- stick cookie: to bind a downstream client to an upstream server
- stick learn: bind a downstream client to an upstream server by using an existing cookie
- stick routing: granular control over how your persistent sessions are routed to the upstream server
- Connection Draining: gracefully remove servers for maintenance or other reasons while still serving sessions

2. Healthy Check

2.1 Passive Healthy Check

max_files: define the max fails number
file_timeout: define the timeout

```
upstream backend {
    server backend1.example.com:1234 max_fails=3 fail_timeout=3s;
    server backend2.example.com:1234 max_fails=3 fail_timeout=3s;
}
```

2.2 Active Healthy Check

Available in Nginx Plus

2.3 Slow Start

Use the slow_start parameter on the server directive to gradually increase the number of connections over a specified time as a server is reintroduced to the upstream load-balancing pool

```
upstream {
    zone backend 64k;

    server server1.example.com slow_start=20s;
    server server2.example.com slow_start=15s;
}
```

2.4 TCP Healthy Check

Use the health_check directive in the server

Use the health_check directive in the server block for an active health check:

- interval: interval time
- passes: 进行几次检查后，认为连接健康
- fails: 失败几次检查后，认为连接不健康

```
stream {
    server {
        listen       3306;
        proxy_pass   read_backend;
        health_check interval=10 passes=2 fails=3;
    }
}
```

Chapter 3 流量管理

2020年1月28日 星期二 15:17

1. A/B Testing

Use the split_clients module to direct a percentage of your clients to a different upstream pool

```
split_clients "${remote_addr}AAA" $variant {
    20.0%     "backendv2";
    *         "backendv1";
}
```

```
location / {
    proxy_pass http://$variant
}
```

2. Using the GeoIP Module

install GeoIP Module:
apt install libnginx-mod-http-geoip

With the GeoIP database for countries and cities on the local disk, you can now instruct the NGINX GeoIP module to use them to expose embedded variables based on the client IP address:

```
load_module "/usr/lib64/nginx/modules/ngx_http_geoip_module.so";
```

```
http {
    geoip_country /etc/nginx/geoip/GeoIP.dat;
    geoip_city /etc/nginx/geoip/GeoLiteCity.dat;
...
}
```

load_module	加载一个动态模块
geoip_country	加载 GeoIP.dat
geoip_proxy	define your proxy IP address range
geoip_proxy_recursive	look for the original IP

GeoIP 模块引入了如下一些变量：

- $geoip_coun try_code, $geoip_country_code3, and $geoip_country_name
- $geoip_city_country_code, $geoip_city_country_code3, and $geoip_city_country_name
- $geoip_city, $geoip_city_continent_code, $geoip_latitude, $geoip_longi tude, and $geoip_postal_code
- $geoip_region and $geoip_region_name

[illegible]

使用方式：

```
load_module
  "/usr/lib64/nginx/modules/ngx_http_geoip_module.so";

http {
    map $geoip_country_code $country_access {
        "US"    0;
        "RU"    0;
        default 1;
    }
    ...
}
```

```
server {
    if ($country_access = '1') {
      return 403;
    }
    ...
}
```

★3. Limiting

3.1 Limit Connections

limit_conn [zone] [max]	server	限制最大连接为 max
limit_conn_zone ...	http	设置连接限制的存储空间
limit_conn_status [code=503]	http	超出范围时候的连接大小

```
http {
    limit_conn_zone $binary_remote_addr zone=limitbyaddr:10m;
    limit_conn_status 429;
    limit_conn_status 429;
    ...
    server {
        ...
            limit_conn limitbyaddr 40;
        ...
    }
}
```

3.2 Limit Rate

limit_req [zone] [burst]	server	设置连接速率
limit_req_zone	http	设置连接存储的空间
limit_req_status [code=503]	http	设置超出速率的状态
limit_req_log_level		错误级别，默认是 error

```
http {
   limit_req_zone $binary_remote_addr
       zone=limitbyaddr:10m rate=1r/s;
   limit_req_status 429;
   ...
   server {
       ...
           limit_req zone=limitbyaddr burst=10 nodelay;
       ...
   }
}
```

3.3 Limit Bandwidth

limit_rate	http server loc	限制的值

limit_rate [bandwidth]	http,server,location	限制的值
limit_rate_after [bandwidth]	http,server,location	当达到这个值后触发

```
location /download/ {
    limit_rate_after 10m;
    limit_rate 1m;
}
```

Chapter 4 缓存机制

2020年1月28日 星期二 16:20

proxy_cache_path	定义一个Cache地址
proxy_cache	指定cache使用的Zone
proxy_cache_key	定义如何检索一个缓存，key决定了缓存的有效性，因而非常征用。默认值 `"$scheme$proxy_host$request_uri"`.
proxy_cache_bypass [value]	定义如何去旁路一个缓存。 当 value非0的时候，旁路缓存
expires [time]	定义一个缓存多久过期
proxy_cache_purge	Nginx Plus Only,清理一个缓存

```
proxy_cache_path /var/nginx/cache
                 keys_zone=CACHE:60m
                 levels=1:2
                 inactive=3h
                 max_size=20g;
proxy_cache CACHE;
```

上述例子中定义了一块缓存空间：

- 名称是CACHE，内存大小60M
- 地址是 /var/nginx/cache
- 非活跃时间是3小时
- 最大大小20G

```
proxy_cache_bypass $http_cache_bypass;
```

当header中 cache_bypass 不是0的时候，旁路缓存。

Chapter 5 自动化

2020年1月28日 星期二 16:41

1. Nginx Plus API

Available for Nginx Plus

2. Installing with Puppet
3. Installing with Chef
4. Installing with Ansible
5. Installing with SaltStack
6. Automating Configurations with Consul Templating

```
upstream backend { {{range service "app.backend"}}
    server {{.Address}};{{end}}
}
```

```
# consul-template -consul consul.example.internal -template \
 template:/etc/nginx/conf.d/upstream.conf:"nginx -s reload"
```

[illegible]

Chapter 6 认证

2020年1月28日 星期二　　16:45

★1. HTTP Basic Authentication

使用 openssl 或者 htpasswd 命令创建一个用户名单列表

openssl passwd MyPassword1234
htpasswd c -b {username} {passwd}

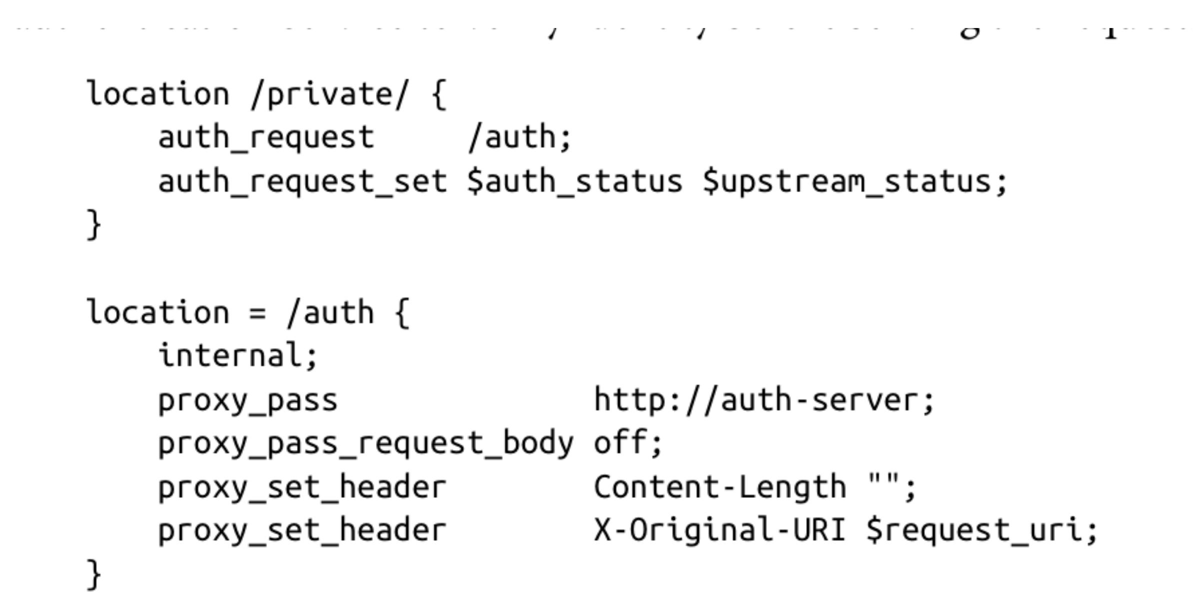
auth_basic	定义 basic_auth 的域
auth_basic_user_file	定义用户名和密码的文件

建议在 HTTPS 上使用 basic_auth

2. Authentication Subrequests

定义一个第三方网页也进行认证, 使用了 http_auth_request_module 模块。

使用一个 Subrequests 来判定是否有权限进行访问。当返回 200 时候，则允许访问；否则返回 401 或者 403。



```
location /private/ {
    auth_request     /auth;
    auth_request_set $auth_status $upstream_status;
}

location = /auth {
    internal;
    proxy_pass              http://auth-server;
    proxy_pass_request_body off;
    proxy_set_header        Content-Length "";
    proxy_set_header        X-Original-URI $request_uri;
}
```

3. Validating JWTS

Nginx Plus Only

4. Creating JSON Web Keys

Nginx Plus Only

5. Authenticate Users via Existing OpenID Connect SSO

Nginx Plus Only

6. Obtaining the JSON Web Key from Google

Nginx Plus Only

Chapter 7 安全控制（SSL等）

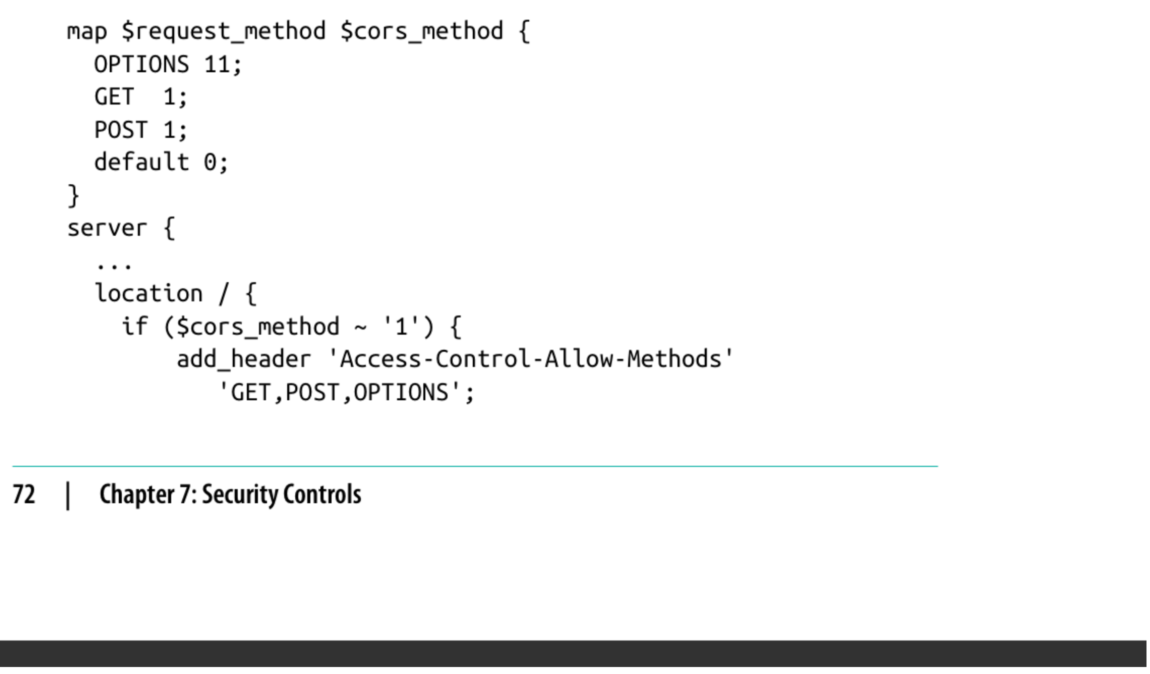
2020年1月29日 星期三　　10:34

★1. Access Based On IP Address

allow [CIDR | IP | Socket | all]	允许指定IP或者IP段访问
deny [CIDR | IP | Socket | all]	拒绝指定IP、IP段访问

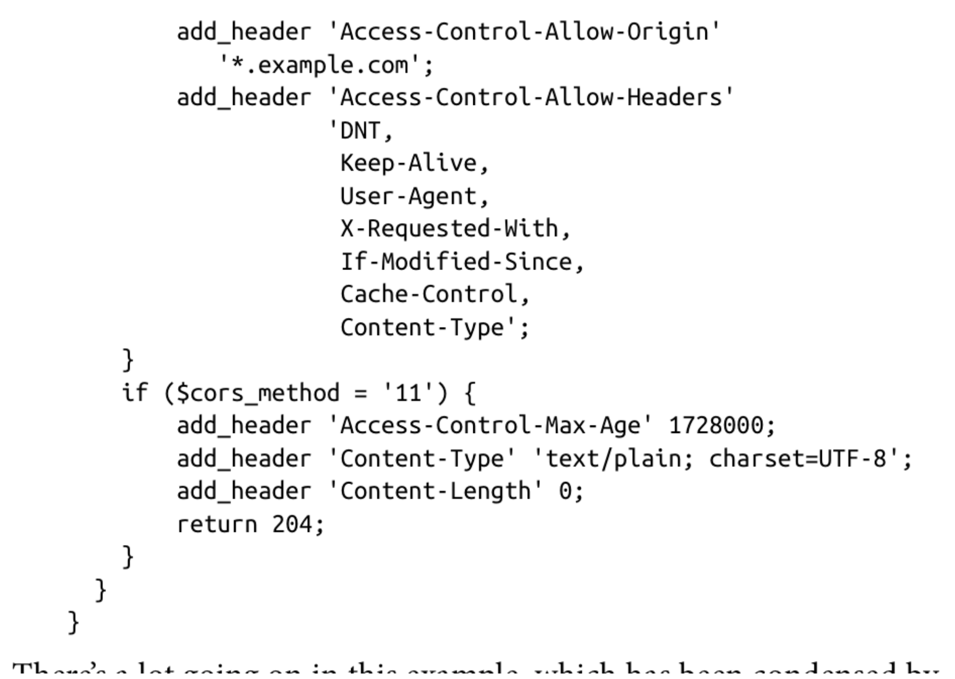
★2. Allowing Cross-Origin Resource Sharing

例子：



```
map $request_method $cors_method {
  OPTIONS 11;
  GET  1;
  POST 1;
  default 0;
}
server {
  ...
  location / {
    if ($cors_method ~ '1') {
        add_header 'Access-Control-Allow-Methods'
           'GET,POST,OPTIONS';
```

72 | Chapter 7: Security Controls



```
            add_header 'Access-Control-Allow-Origin'
               '*.example.com';
            add_header 'Access-Control-Allow-Headers'
                       'DNT,
                        Keep-Alive,
                        User-Agent,
                        X-Requested-With,
                        If-Modified-Since,
                        Cache-Control,
                        Content-Type';
        }
        if ($cors_method = '11') {
            add_header 'Access-Control-Max-Age' 1728000;
            add_header 'Content-Type' 'text/plain; charset=UTF-8';
            add_header 'Content-Length' 0;
            return 204;
        }
      }
    }
```

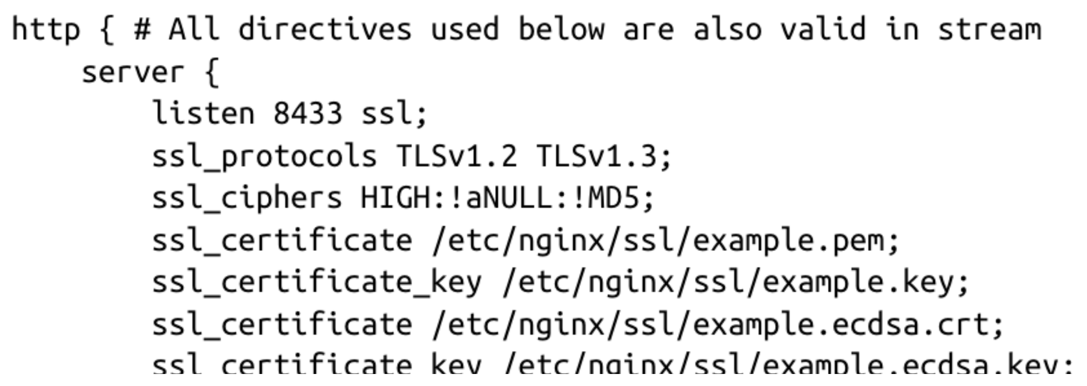
There's a lot going on in this example, which has been condensed by

通过 add_header 来提供 CORS 访问控制。

★3. SSL/TLS

3.1 Client-Side SSL

使用 ngx_http_ssl_module 或者 ngx_steam_ssl_module 来提供 SSL/TLS 功能。



```
http { # All directives used below are also valid in stream
    server {
        listen 8433 ssl;
        ssl_protocols TLSv1.2 TLSv1.3;
        ssl_ciphers HIGH:!aNULL:!MD5;
        ssl_certificate /etc/nginx/ssl/example.pem;
        ssl_certificate_key /etc/nginx/ssl/example.key;
        ssl_certificate /etc/nginx/ssl/example.ecdsa.crt;
        ssl_certificate_key /etc/nginx/ssl/example.ecdsa.key;
```

```
            ssl_session_cache shared:SSL:10m;
            ssl_session_timeout 10m;
        }
    }
```

ssl_protocols	指定TLS或者SSL 版本	推荐 TLSv1.2 TLSv1.3
ssl_ciphers	指定TLS的密码学套件	HIGH:!aNULL:!MD5
ssl_certificate	指定证书	指定公钥证书
ssl_certificate_key	指定私钥	私钥的位置
ssl_session_cache	指定用于保存session的share memory	shared:SSL:10m;
ssl_session_timeout	指定session的保存时间	10m

3.2 Upstream Encryption

用于向上游请求的时候，进行ssl通信：

proxy_pass	上游地址
proxy_ssl_verify [on/off]	开启ssl验证
proxy_ssl_verify_depth [num]	指定最大验证深度
proxy_ssl_protocols	指定ssl版本
proxy_ssl_certificate	指定客户端证书

proxy_ssl_certificate_key	指定客户端私钥
proxy_ssl_crl	指定客户端的证书撤销列表

3.3 HTTPS Redirects

将 80 端口重定向到 443 端口

```
server {
    listen 80 default_server;
    listen [::]:80 default_server;
    server_name _;
    return 301 https://$host$request_uri;
}
```

当 SSL/TLS 在NGINX之前终止的时候，使用 HTTP 的 X-Forwarded-Proto 头来重定向

```
server {
    listen 80 default_server;
    listen [::]:80 default_server;
    server_name _;
    if ($http_x_forwarded_proto = 'http') {
        return 301 https://$host$request_uri;
    }
}
```

3.4 HTTP Strict Transport Security(HSTS)

添加如下 HTTP头部来启用 HSTS

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

4. Secureing Link Module

4.1 Securing a Location

使用 secure link module 提供 Securing Link 功能。使用 secure_link_secret 模块。

```
location /resources {
    secure_link_secret mySecret;
    if ($secure_link = "") { return 403; }

    rewrite ^ /secured/$secure_link;
}

location /secured/ {
    internal;
```

76 | Chapter 7: Security Controls

```
    root /var/www;
}
```

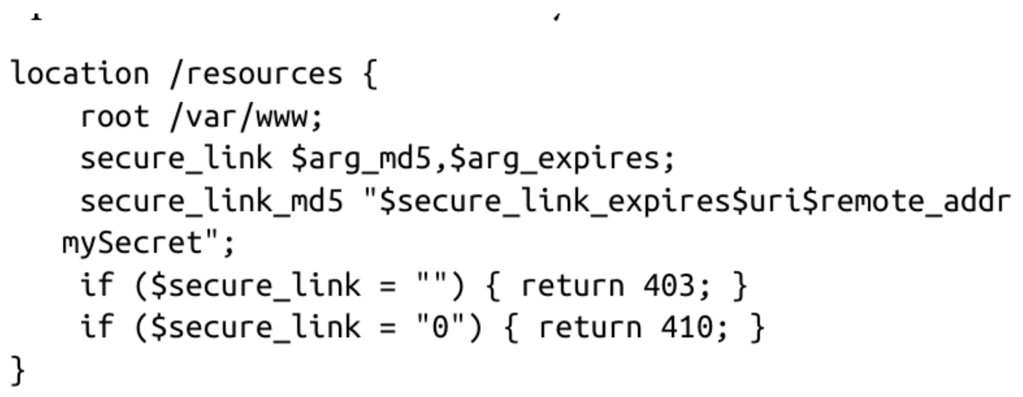
当公开访问 /resources 时候，将返回403.

Forbidden unless the request URI includes an md5 hash string that can be verified with the secret provided to the secure_link_secret directive

md5(index.htmlmySecret) = a53bee08a4bf0bbea978ddf736363a12

当访问 /resources/a53bee08a4bf0bbea978ddf736363a12 时候将内部重定向 /secured/$secure_link 进而访问 /var/www/secured/index.html

4.2 Securing a Location with an Expire Data



```
location /resources {
    root /var/www;
    secure_link $arg_md5,$arg_expires;
    secure_link_md5 "$secure_link_expires$uri$remote_addr
   mySecret";
    if ($secure_link = "") { return 403; }
    if ($secure_link = "0") { return 410; }
}
```

secure_link 接受逗号分隔的两个参数，一个是HTTP参数的md5值，一个是过期时间。

secure_link_md 接受一个参数申明了用于生产md5值的参数。

★5. 综合使用多个访问控制方法

使用 satisfy 来申明如何协调多个访问控制方法：

satisfy [any|all]	申明多个访问控制是 or 还是 and 的方式来综合。

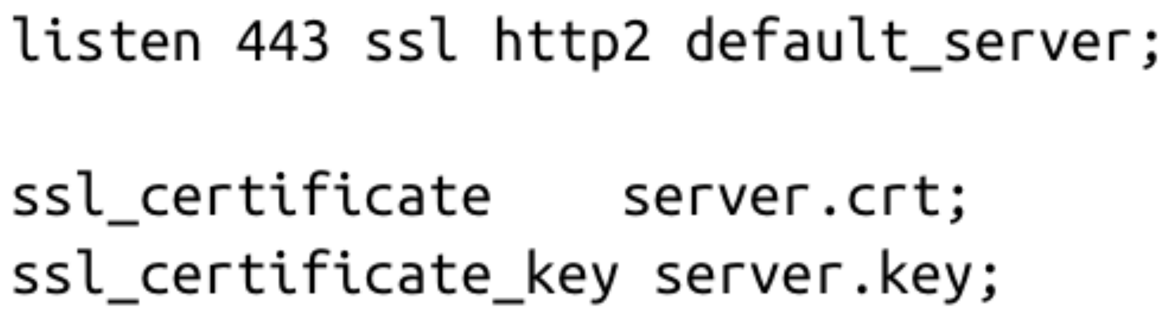
```
location / {
    satisfy any;

    allow 192.168.1.0/24;
    deny  all;

    auth_basic           "closed site";
    auth_basic_user_file conf/htpasswd;
}
```

6. Dynamic DDoS Mitigation

Nginx Plus

Chapter 8 HTTP/2

2020年1月29日 星期三 11:17

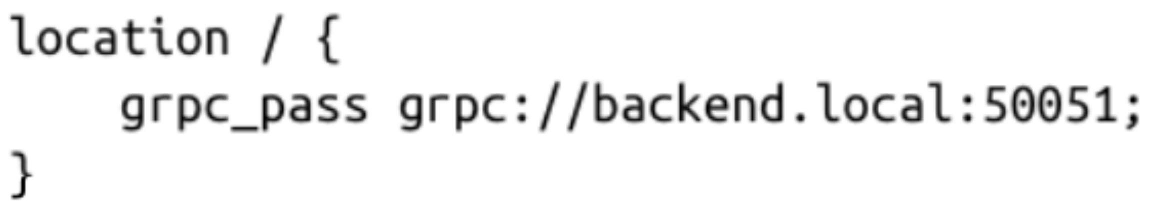


```
listen 443 ssl http2 default_server;

ssl_certificate     server.crt;
ssl_certificate_key server.key;
```

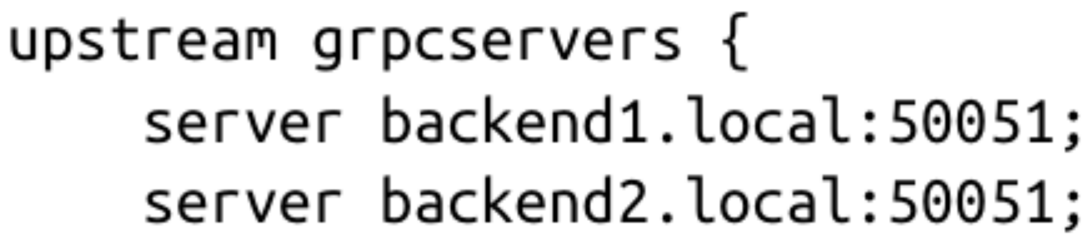
使用 http2 和 ssl 属性来开启 HTTP/2

1. 启用 gRPC 调用

使用 grpc_pass 来告诉 NGINX 将通信使用 gRPC 协议。



```
location / {
    grpc_pass grpc://backend.local:50051;
}
```



```
upstream grpcservers {
    server backend1.local:50051;
    server backend2.local:50051;
}
```

2. HTTP/2 Server Push

使用 http2_push 来进行 HTTP/2 的主动推送功能

```
server {
    listen 443 ssl http2 default_server;

    ssl_certificate     server.crt;
    ssl_certificate_key server.key;
    root /usr/share/nginx/html;

    location = /demo.html {
        http2_push /style.css;
        http2_push /image1.jpg;
    }
}
```

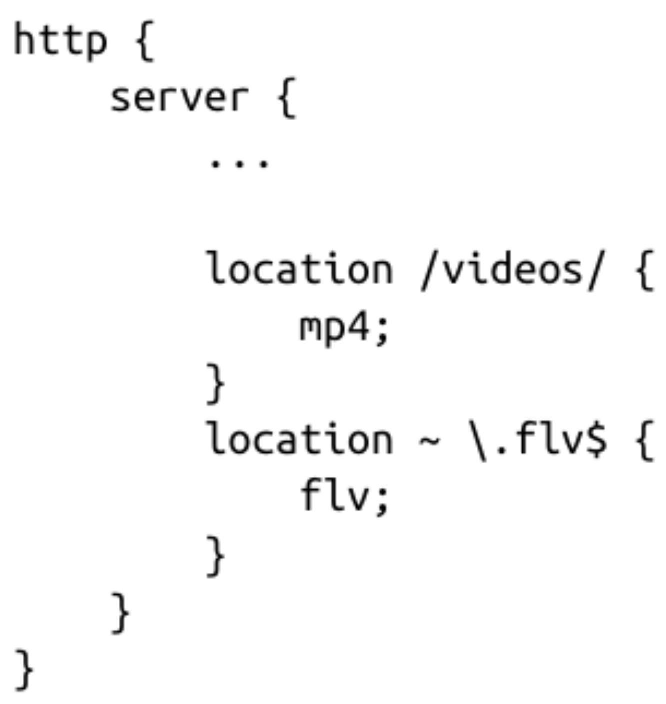
Chapter 9 Sophisticated Media Streaming

2020年1月29日 星期三   11:23

本章介绍如何处理流媒体（MPEG-4或者FLV）

1. 处理MP4和FLV

使用 mp4 或者 flv 指令来提供 progressive downloads 和 HTTP pseudostreaming and support seeking。



```
http {
    server {
        ...

        location /videos/ {
            mp4;
        }
        location ~ \.flv$ {
            flv;
        }
    }
}
```

mp4	开启MP4 缓存
flv	开启flv缓存

flv	开启flv缓存
mp4 _limit_rate_after 15s	运行先下载15s 后再触发流量限制
mp4_limit_rate 1.2	允许按照 120% 的速率缓存。比如播放10s缓存12s。

2. 处理 HLS

Nginx Plus Only

3. 处理 HDS

Nginx Plus Only

Chapter 10 云部署

2020年1月29日 星期三　　12:42

本章介绍了在AWS，Azure，Google Cloud 上部署Nginx的方法。跳过。

Chapter 11 容器和微服务

2020年1月29日 星期三 12:43

1. DNS SRV Records

NGINX Plus Only

```
http {
    resolver 10.0.0.2;

    upstream backend {
        zone backends 64k;
        server api.example.internal service=http resolve;
    }
}
```

2. Using the Official NGINX Image in Docker

```
$ docker run --name my-nginx -p 80:80 \
    -v /path/to/content:/usr/share/nginx/html:ro -d nginx
```

```
FROM centos:7

# Install epel repo to get nginx and install nginx
RUN yum -y install epel-release && \
    yum -y install nginx nginx-mod-http-perl

# add local configuration files into the image
ADD /nginx-conf /etc/nginx

EXPOSE 80 443

CMD ["nginx"]
```

使用 ngx_http_perl_module 来获取环境变量。

```
daemon off;
env APP_DNS;
```

```
include /usr/share/nginx/modules/*.conf;
...
http {
  perl_set $upstream_app 'sub { return $ENV{"APP_DNS"}; }';
  server {
    ...
    location / {
      proxy_pass https://$upstream_app;
    }
  }
}
```

3. Kubernetes Ingress Controller

TBD

4. OpenShift Router

TBD

Chapter 12 高可用性部署

2020年1月29日 星期三　　13:02

1. NGINX HA Mode

Nginx Plus Only

2. Load-Balancing Load Balancers with DNS

使用多个 A 记录的 DNS 来提供负载平衡。

3. Load Balancing on EC2

Available on AWS.

4. Configuration Synchronization

Nginx Plus Only

5. State Sharing with Zone Sync

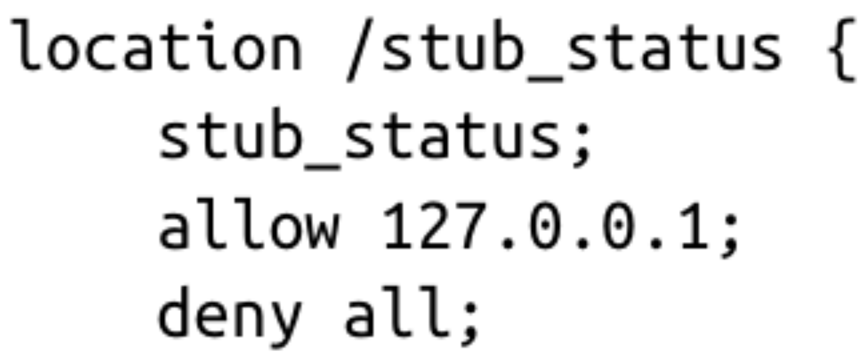
Nginx Plus Only

Chapter 13 先进活动监控

2020年1月29日 星期三 13:06

1. Enable NGINX Open Source Stub Status

使用 Nginx 的 stub_status 模块来启用。



```
location /stub_status {
    stub_status;
    allow 127.0.0.1;
    deny all;
```

就可以访问 localhost/stub_status 来获取当前信息。这个模块还定义了一些变量：

- $connections_reading
- $connections_writing
- $connections_waiting

2. Enabling the Nginx Plus Monitoring Dashboard Provided by Nginx Plus

3. Collecting Metrics Using the NGINX Plus API

Chapter 14 调试和日志

2020年1月29日 星期三 13:11

★1. Access Logs

log_format [log_format_name] [format]	设置日志格式
access_log [location] [log_format_name]	设置 access_log

```
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;
```

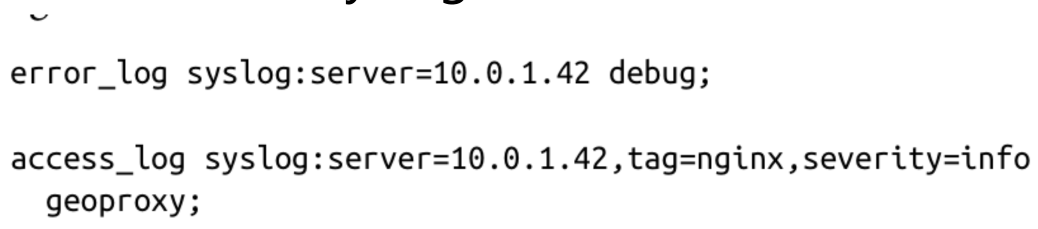
★2. Error Logs

error_log [location] [levels]	设置 error_log。参数是位置和级别。

级别有 debug, info, notice, warn, error, crit, alert, emery.

3. Forwarding to Syslog

设置将日志转发到 syslog。



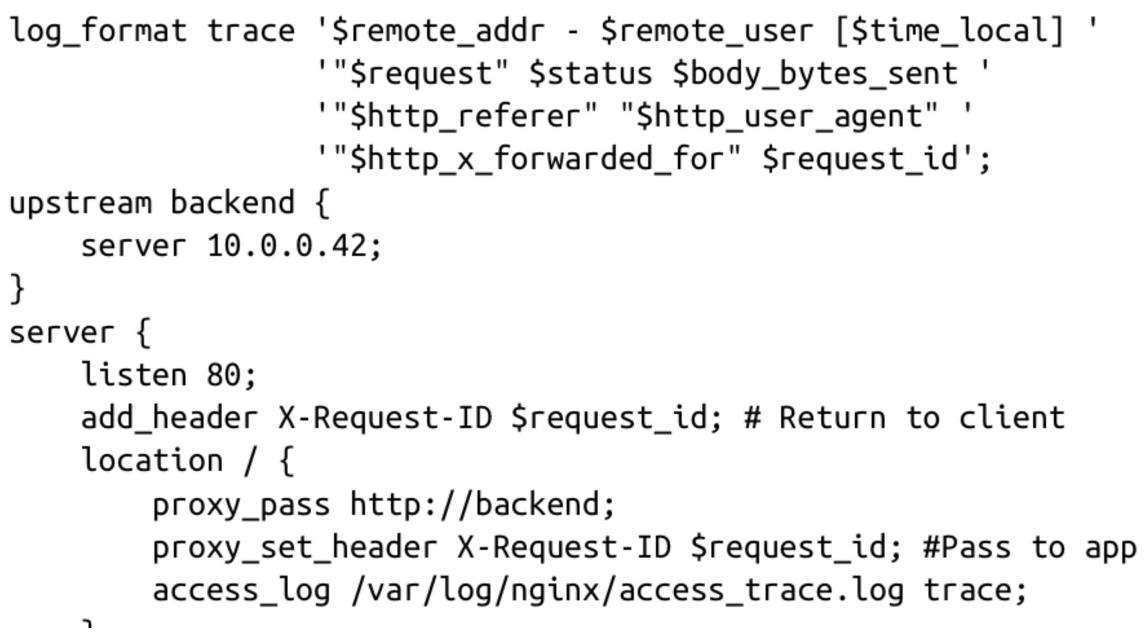
```
error_log syslog:server=10.0.1.42 debug;

access_log syslog:server=10.0.1.42,tag=nginx,severity=info
  geoproxy;
```

server 可以使用 ip，domain 或者unix socket。

可选项有 facility，severity，tag， nohostname 等。

4. Request Tracing

通过设置 request-id 来设置跟踪日志。



```
log_format trace '$remote_addr - $remote_user [$time_local] '
                 '"$request" $status $body_bytes_sent '
                 '"$http_referer" "$http_user_agent" '
                 '"$http_x_forwarded_for" $request_id';
upstream backend {
    server 10.0.0.42;
}
server {
    listen 80;
    add_header X-Request-ID $request_id; # Return to client
    location / {
        proxy_pass http://backend;
        proxy_set_header X-Request-ID $request_id; #Pass to app
        access_log /var/log/nginx/access_trace.log trace;
```

```
          }
      }
```

Chapter 15 性能调优

2020年1月29日 星期三　　13:28

1. Automating Tests with Load Drivers

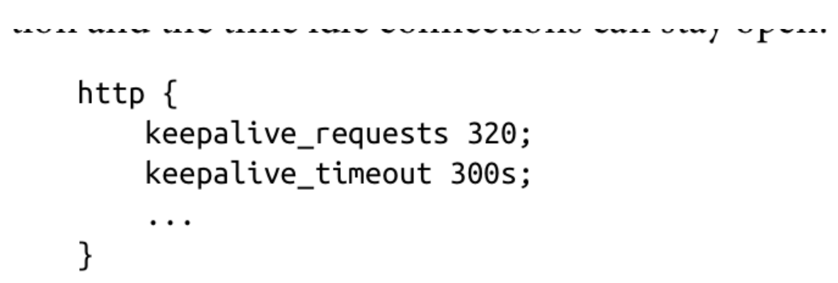
Use an HTTP load-testing tool such as Apache JMeter, Locust, Gatling, or whatever your team has standardized on

2. KeepAlive

2. Keeping Connections Open to Clients

提供长连接选项。

keepalive_requests [num]	设置单个连接的请求数量
keepalive_timeout [time]	设置连接的空闲时间



tion and the time idle connections can stay open.

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

2.2 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;
}
```

★3. Buffering Responses

proxy_buffering [on/off]	设置是否开启 buffering
proxy_buffer_size [4k/8k]	设置读取response第一部分缓存的大小
proxy_buffers [num] [size]	设置 buffer的数量和大小
proxy_busy_buffer_size [size = 64k]	limits the size of buffers that can be busy, sending a response to the client while the response is not fully read

4. Buffeering Access Logs

使用access_log 的 buffer 和flush选项来提供缓存选项。

```
access_log /var/log/nginx/access.log main buffer=32k
```

```
        flush=1m;
```

★5. OS Tuning

- net.core.somaxconn 大于 512 时候，需要在 listen 后加入backlog参数。
- 设置 sys.fs.file_max 或者 /etc/secyrity/limits.conf
- 设置 Nginx 配置中的 worker_connections 和 worker_rlimit_nofile 选项。
- 配置 net.ipv4.ip_local_port_range
- 配置 use 选项为 epoll, kuene 还是 select

Chapter 16 其他 Tips 和总结

2020年1月29日 星期三　　13:40

★1. 使用 include 指令来引入其他的 Configs

```
http {
    include config.d/compression.conf;
    include sites-enabled/*.conf
}
```

- 当使用factcgi时，可以引入 fastcgi_param
- 当配置SSL时候，也可单独引入 ssl.conf

★2. Debugging Configs

- NGINX processes requests looking for the most specific matched rule
- You can turn on debug logging. 比如：
    - error_log /var/log/nginx/error.log debug.
- You can enable debugging for particular connections
    - debug_connection 指令在 event 块中有效，使用IP或者CIDR作为参数。
- You can debug for only particular virtual servers
    - 在单独的 server 或者 location 块中使用 error_log 指令

    - error_log 指令
- You can enable core dumps and obtain backtrace from them.
- You are able to log what's happening in rewrite statements with rewrite_log 指令：
    - rewrite_log on