Introduce Nginx

with lua-nginx-module

郑帆

Outline

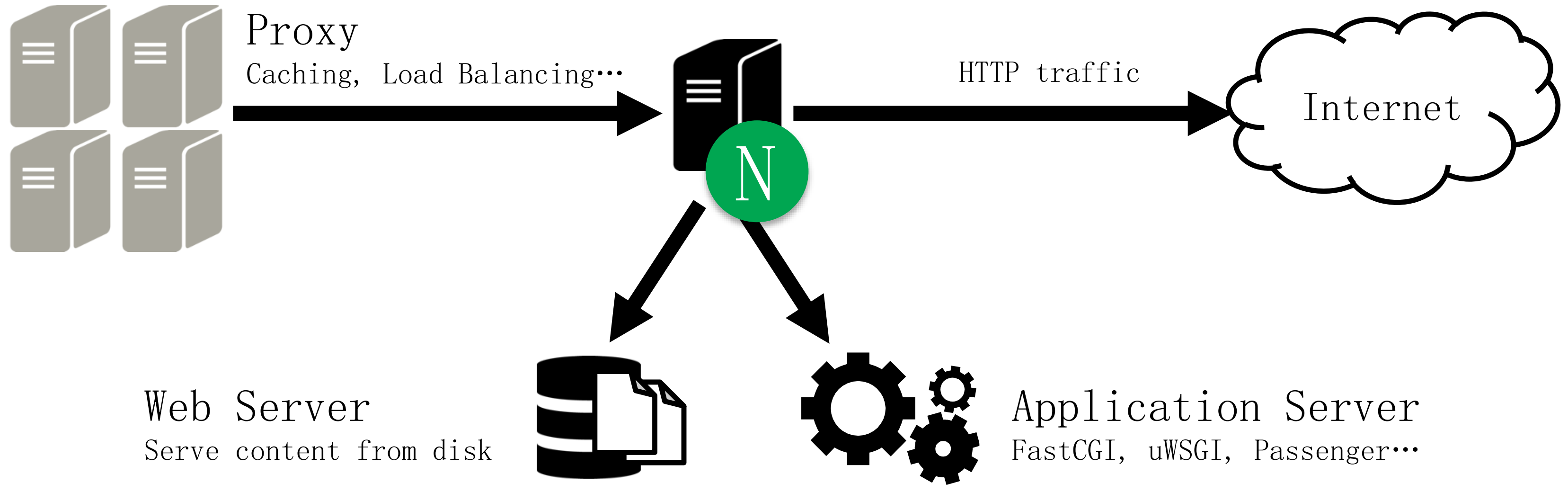
- Nginx
- lua-nginx-module
- example

Nginx

History

- Created by Igor Sysoev in 2002（Russia！ ）
- Since its public launch in 2004.
- Company founded in July 2011（Nginx .inc）
- Commercial support in February 2012
- Nginx Plus in August 2013.
- …

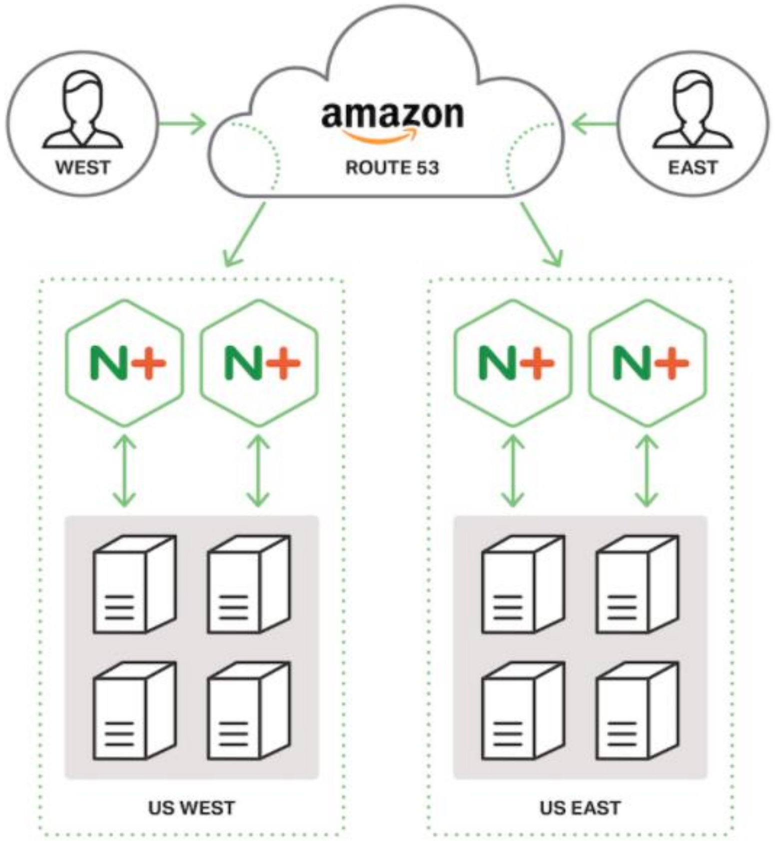
What is NGINX?



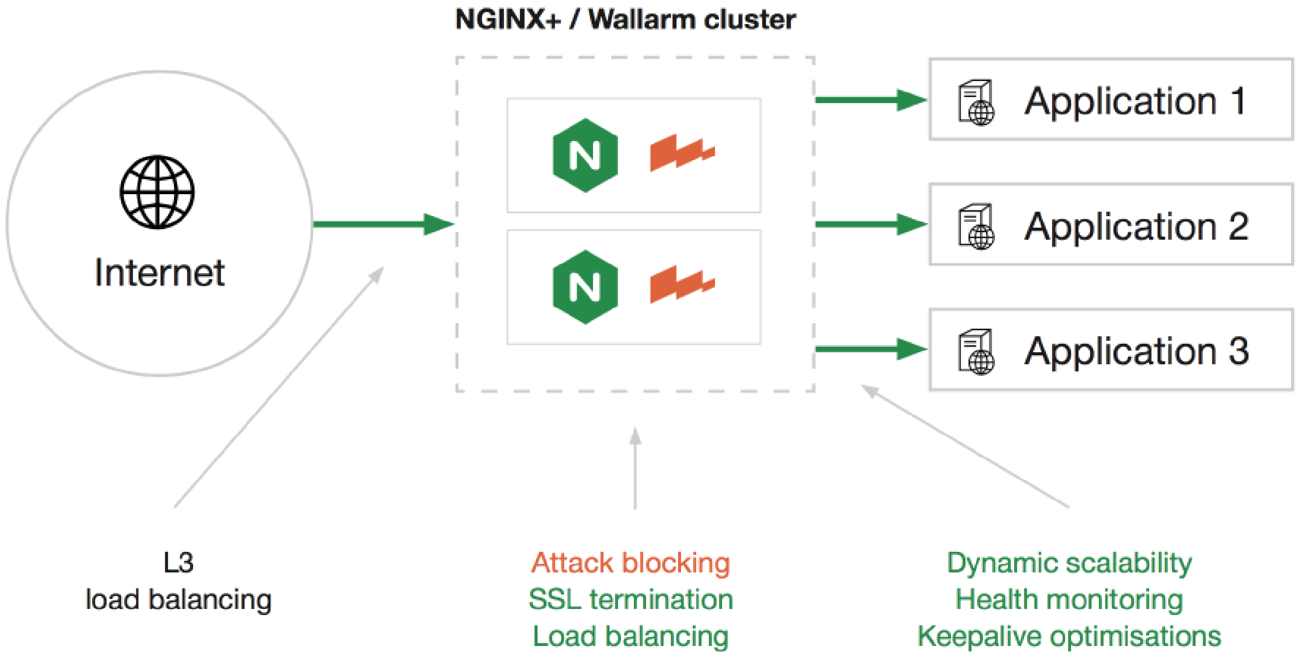
Advanced Features:

☑Application Acceleration	☑Bandwidth Management	☑Authentication
☑SSL and SPDY termination	☑Content-based Routing	☑Video Delivery
☑Performance Monitoring	☑Request Manipulation	☑Mail Proxy
☑High Availability	☑Response Rewriting	☑GeoLocation

A real word



Architecture



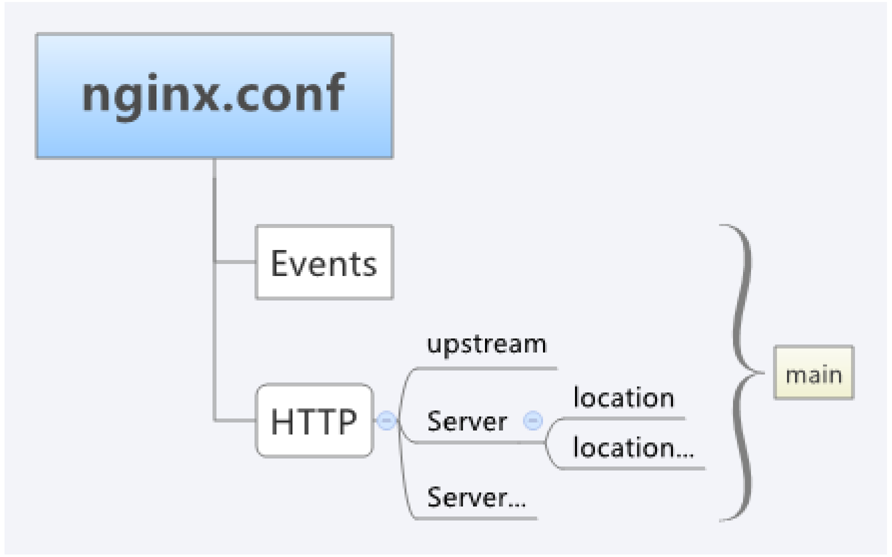
A Quick Tutorial

How to start it?

```
#目录
drwx------ 2 nobody work 4096 May 12  2016 client_body_temp
drwxrwxr-x 2 work   work 4096 May 18  2016 conf              ←配置文件
drwx------ 2 nobody work 4096 May 12  2016 fastcgi_temp
drwxr-xr-x 2 work   work 4096 May 12  2016 html              ←静态文件
drwxrwxr-x 2 work   work 4096 May 18  2016 logs              ←日志
drwx------ 2 nobody work 4096 May 12  2016 proxy_temp
drwxrwxr-x 2 work   work 4096 May 12  2016 sbin              ←命令
drwx------ 2 nobody work 4096 May 12  2016 scgi_temp
drwx------ 2 nobody work 4096 May 12  2016 uwsgi_temp
```

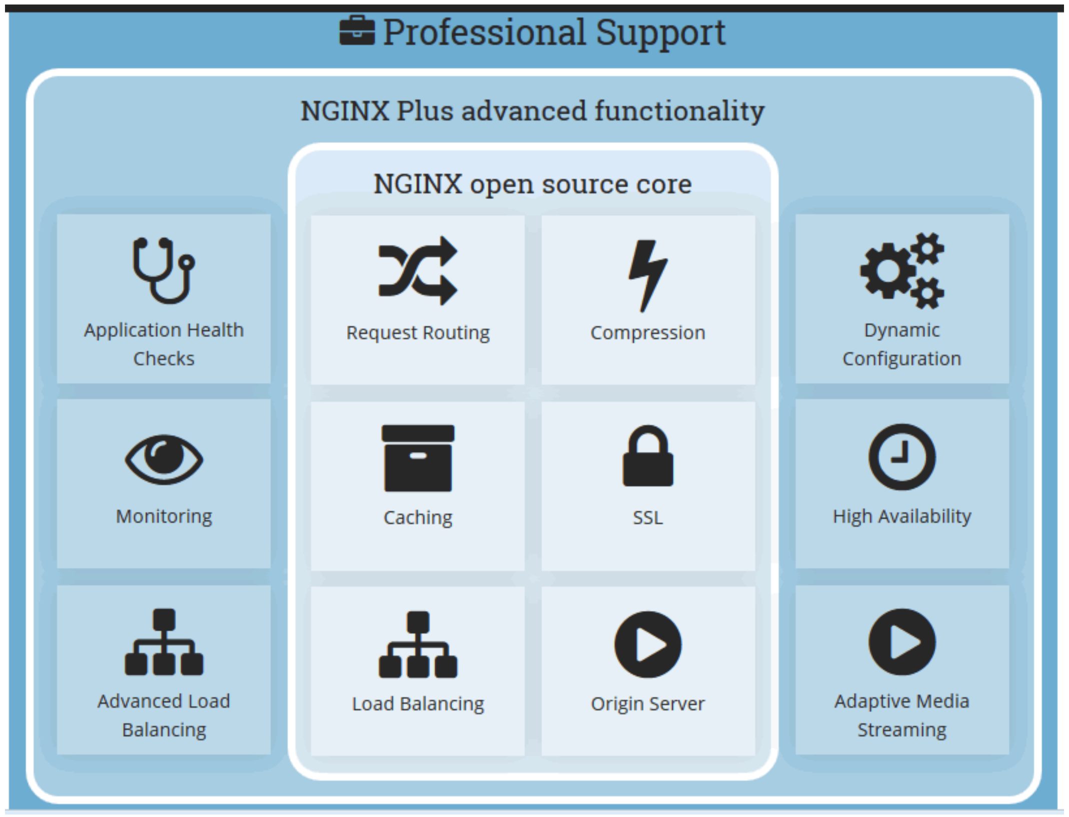
```
#操作
sbin/nginx -c conf/nginx.conf      ←启动
sbin/nginx -t -c conf/nginx.conf  ←检查配置文件
sbin/nginx -s stop                 ←停止服务
sbin/nginx -c reload               ←重启
```

Nginx.conf

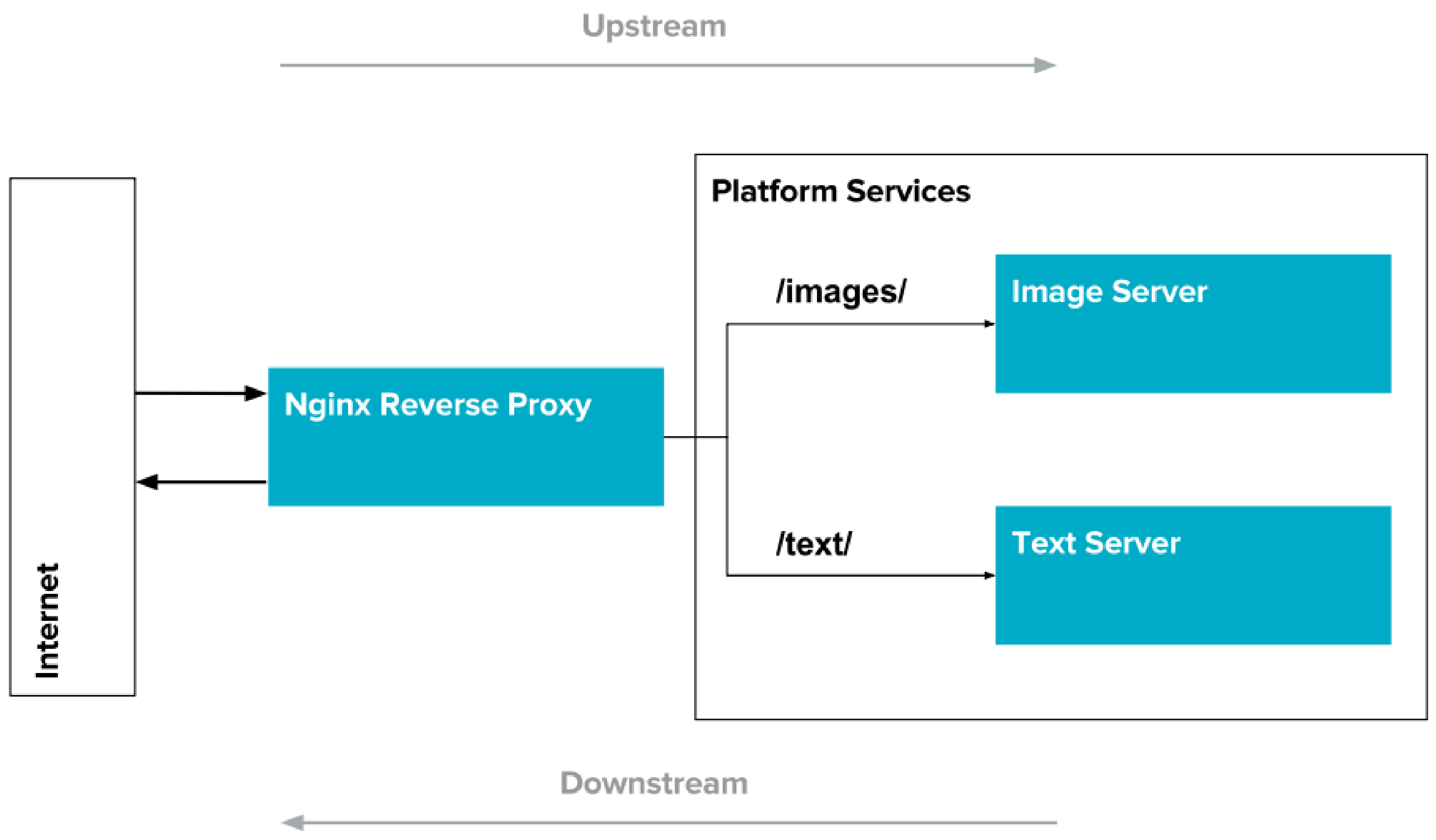


```
events {
     worker_connections  1024;
}

http {
     include        mime.types;
     default_type   ...;
     ...

     server {
        listen 80;

        location / {
            echo 'ok'
        }
     }
}
```

Function Architecture



Request Routing



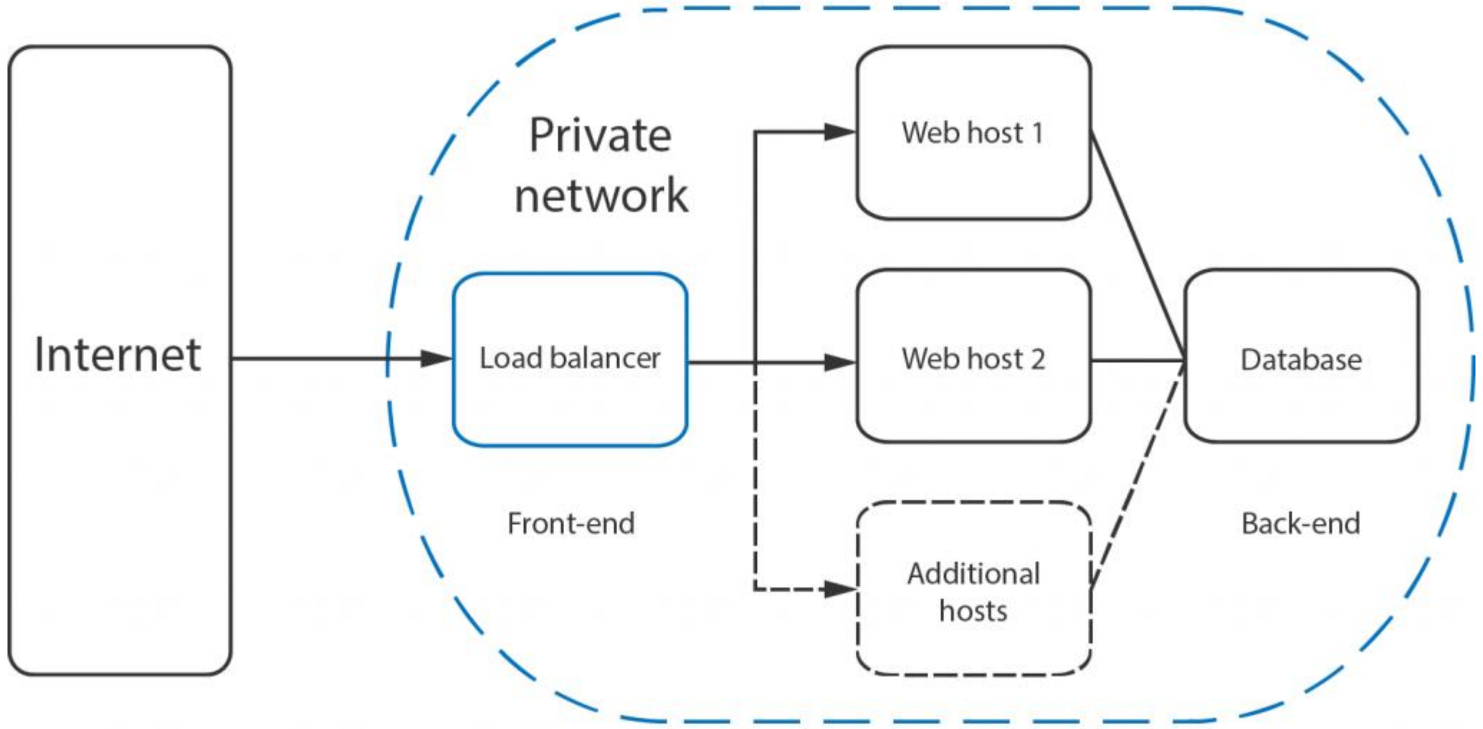
Request Routing

- location = /
  - if found , the search terminates
- location /
  - prefix locations
    longest matching prefix
- location ~ /
  - regular expressions
    terminates on the first match

```
location = / {  #精准匹配
    [ configuration A ]
}

location / {    #普通字符匹配
    [ configuration B ]
}

location ~ /documents/ {#正则匹配，不区分大小写
    [ configuration C ]
}

location ^~ /images/ {#普通字符匹配,不是正则
    [ configuration D ]
}

#正则匹配，不区分大小写
location ~* \.(gif|jpg|jpeg)$ {
    [ configuration E ]
}
```

(prefix strings) -> (regular expressions) -> (prefix location remembered earlier '/')

Load Balancing



Load Balancing

```
server {
    listen 80;

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

upstream backend {

    server webserver1:80 weight=5;
    server webserver2:80 max_fails=3 fail_timeout=30s;

    keepalive 20;
}
```

Load Balancing

```
upstream backend {
    server webserver1:80;
    server webserver2:80;
}
```

```
upstream backend {
    least_conn;
    server webserver1:80;
    server webserver2:80;
}
```

```
upstream backend {
    ip_hash;
    server webserver1:80;
    server webserver2:80;
}
```

- Round-robin is the default
  - Suitable for consistent pages
- Least Connections
  - Suitable for varying pages
- IP Hash
  - Fixed mapping, basic session persistence

Load Balancing

```
upstream backend {
    hash $request_uri consistent;

    server backend1.example.com;
    server backend2.example.com;
}
```

- generic hash method
  - determined from a user-defined key which may be a text, variable,

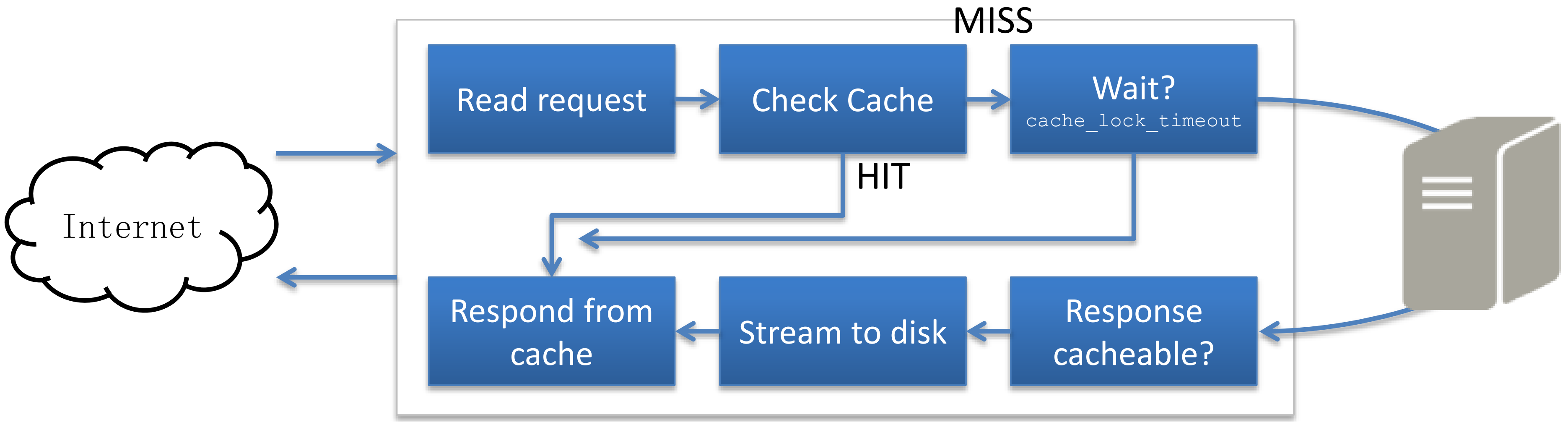
```
upstream backend {
    least_time header;

    server backend1.example.com;
    server backend2.example.com;
}
```

- least_time
  - for each request, selects the server with the lowest average latency and the least number of active connections
  - NGINX Plus

High-performance Caching

- Cache GET and HEAD with no Set-Cookie response
- Cache time defined by
  - X-Accel-Expires
  - Cache-Control
  - Expires
  - https://www.w3.org/Protocols/rfc2616/rfc2616-sec13.html

High-performance Caching



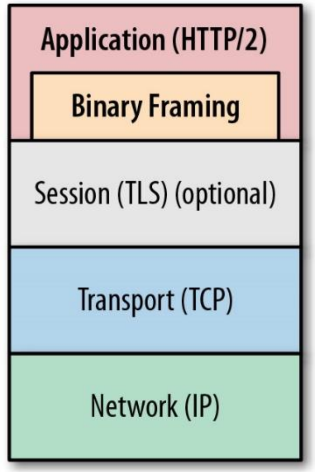
NGINX can use stale content under the following circumstances:

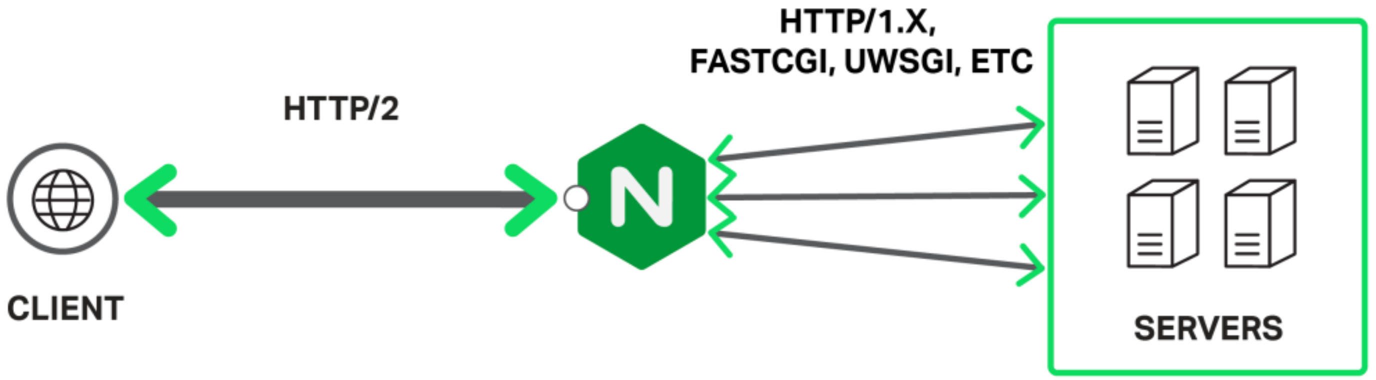
```
proxy_cache_use_stale error | timeout | invalid_header |
    updating | http_500 | http_502 | http_503 | http_504 |
    http_403 | http_404 | off
```

High-performance Caching

```
proxy_cache_path /tmp/cache keys_zone=one:10m levels=1:2 inactive=60m;

server {
    listen      80;
    server_name  localhost;

    location / {
        proxy_pass http://localhost:8080;
        proxy_cache one;
   }
}
```

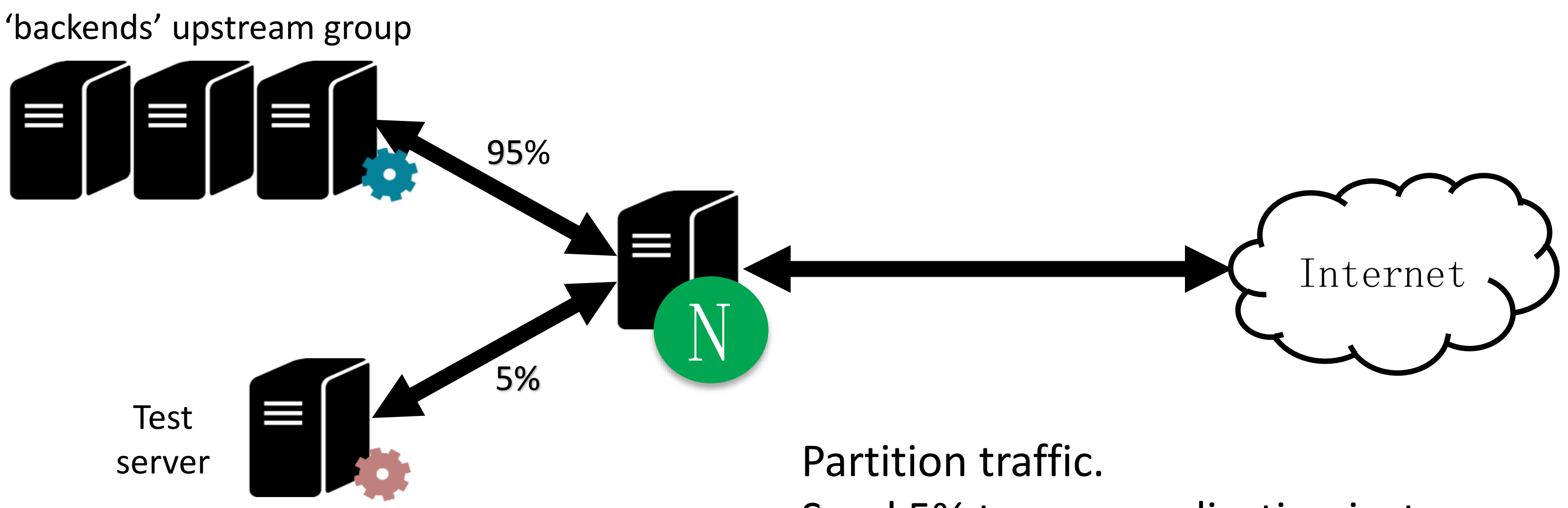
Support With HTTP/2





```
server {
    listen 443 ssl http2 default_server;

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

A/B Testing



Partition traffic.
Send 5% to new application instance

A/B Testing

```
split_clients "${remote_addr}AAA" $servers {
    95% backends;
    *   192.168.56.1:80;
}

server {
   listen 80;

   location / {
      proxy_pass http://$servers;

   }
}
```

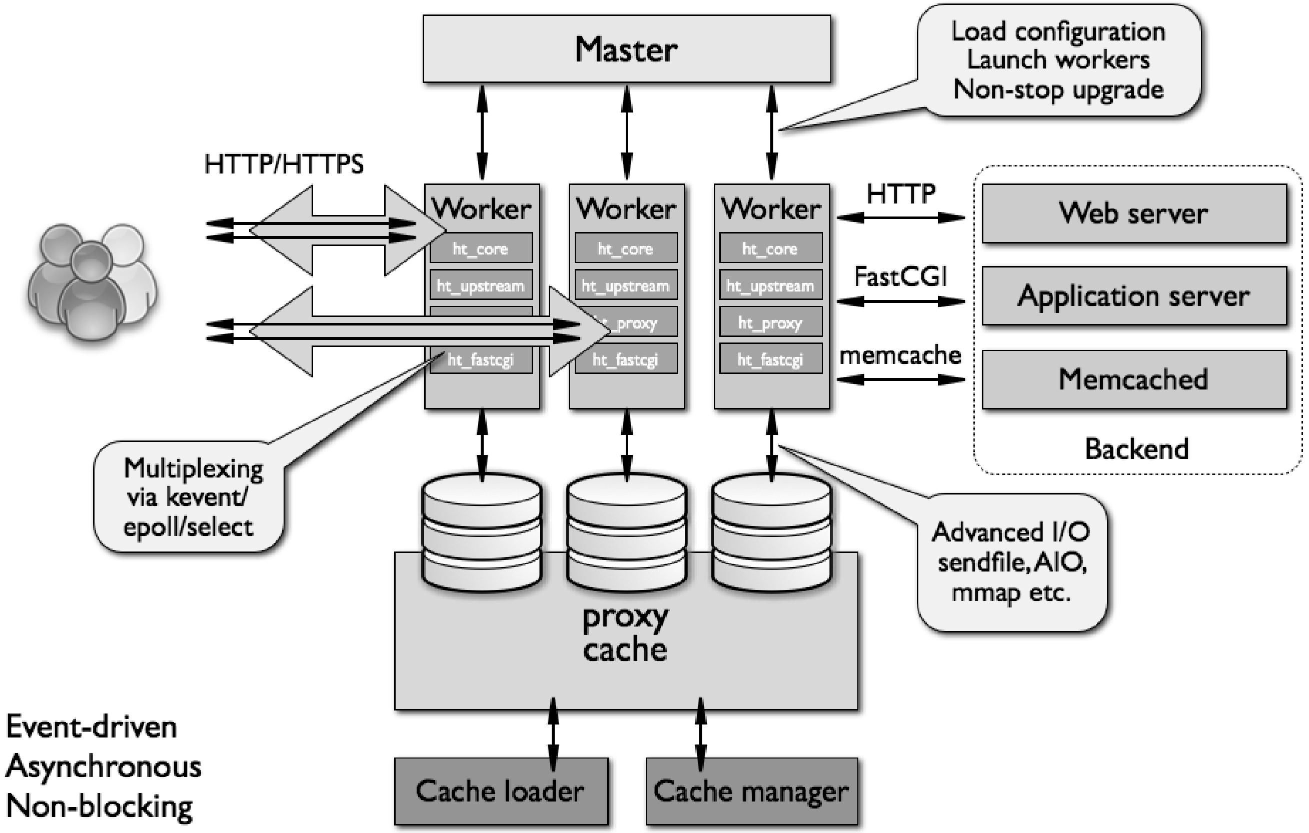
Fight DDoS Attacks

```
limit_req_zone $binary_remote_addr zone=one:10m rate=30r/m;

server {
    ...
    location /login.html {
        limit_req zone=one;
    ...
    }
}
```

```
location / {
    allow 192.168.1.0/24;
    deny all;
}

limit_conn_zone $binary_remote_addr zone=addr:10m;
location /store/ {
        limit_conn addr 10;
        ...
}
```

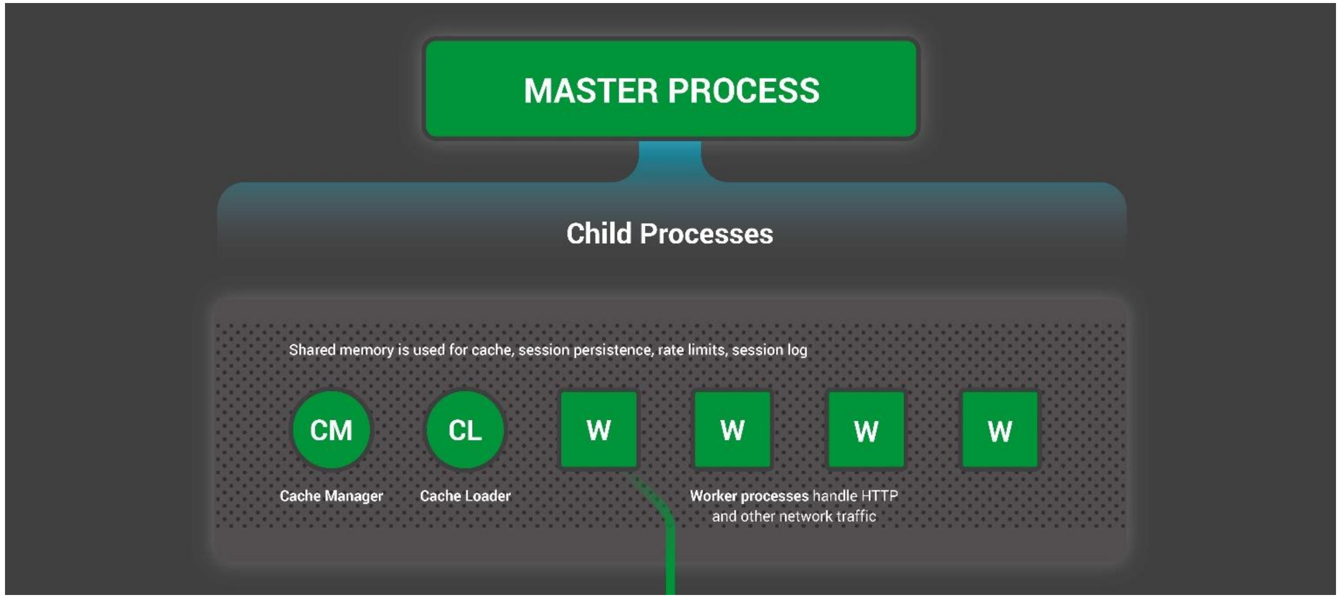
How Nginx works?

How Nginx works



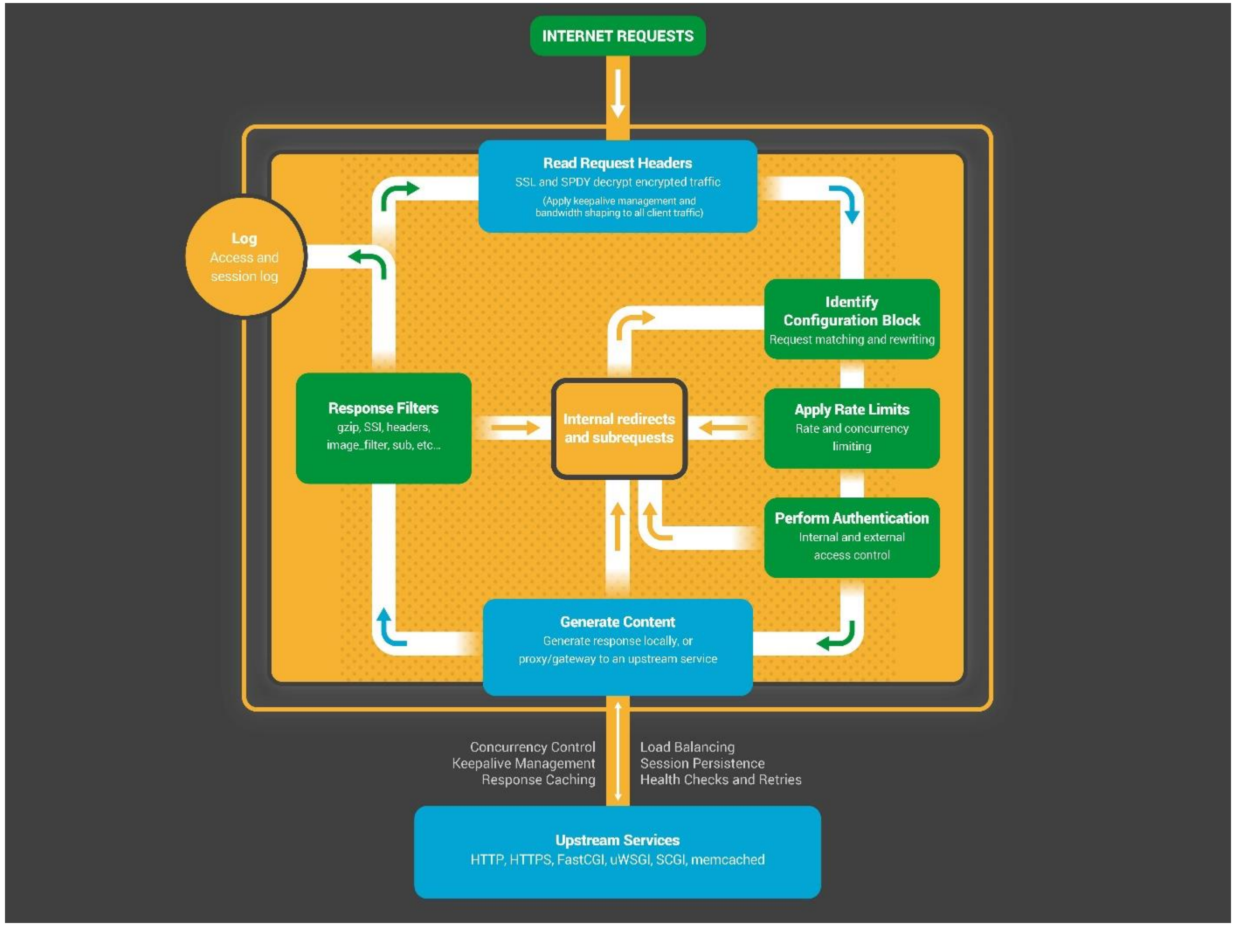
- Event-driven
- Asynchronous
- Non-blocking

How Nginx works


MASTER PROCESS
Child Processes
Shared memory is used for cache, session persistence, rate limits, session log
CM
CL
W
W
W
W
Cache Manager
Cache Loader
Worker processes handle HTTP and other network traffic

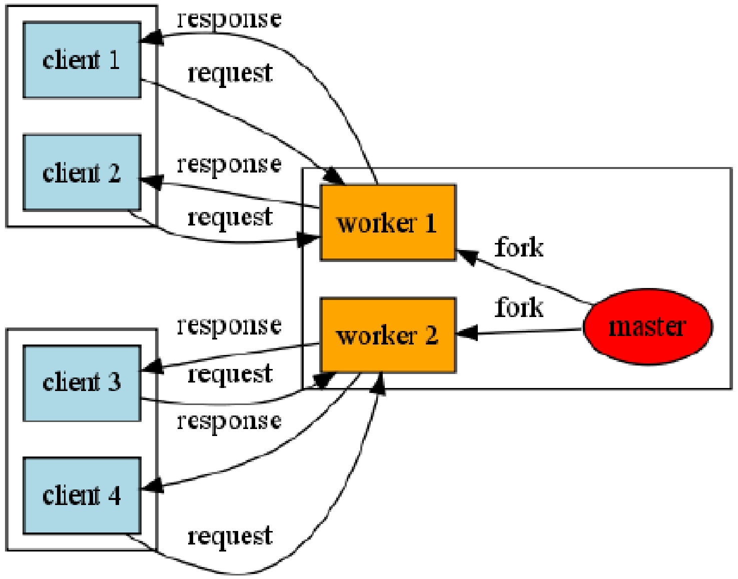
```
# ps -ef --forest | grep nginx
root      32475     1  0 13:36 ?        00:00:00 nginx: master process /usr/sbin/nginx \
                                                   -c /etc/nginx/nginx.conf
nginx     32476 32475  0 13:36 ?        00:00:00  \_ nginx: worker process
nginx     32477 32475  0 13:36 ?        00:00:00  \_ nginx: worker process
nginx     32481 32475  0 13:36 ?        00:00:00  \_ nginx: cache manager process
nginx     32482 32475  0 13:36 ?        00:00:00  \_ nginx: cache loader process
```

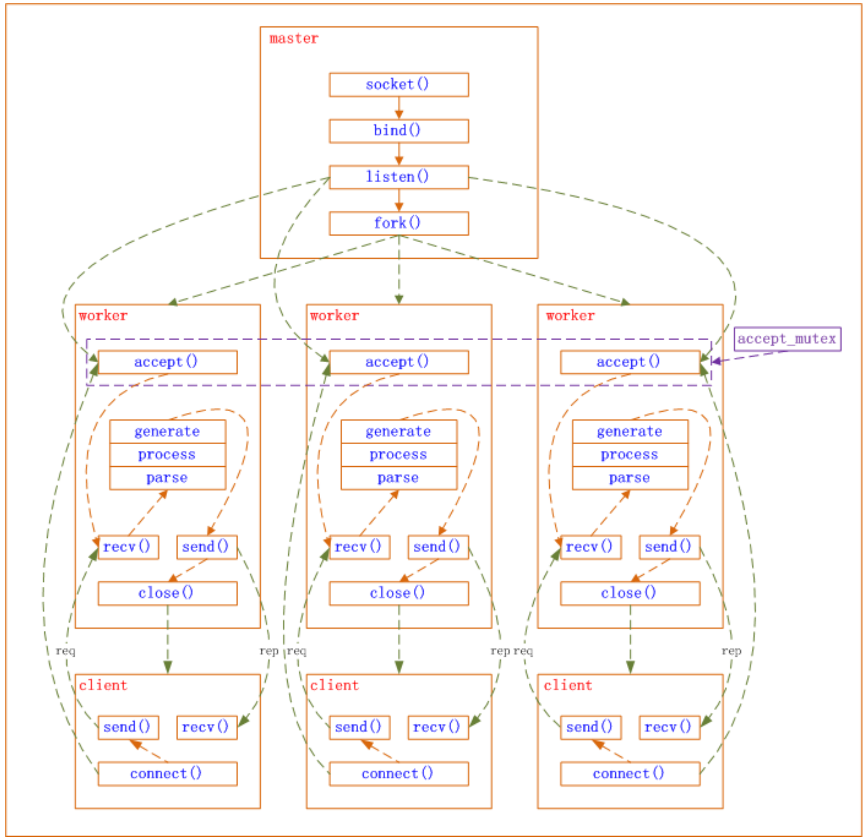
How Nginx works



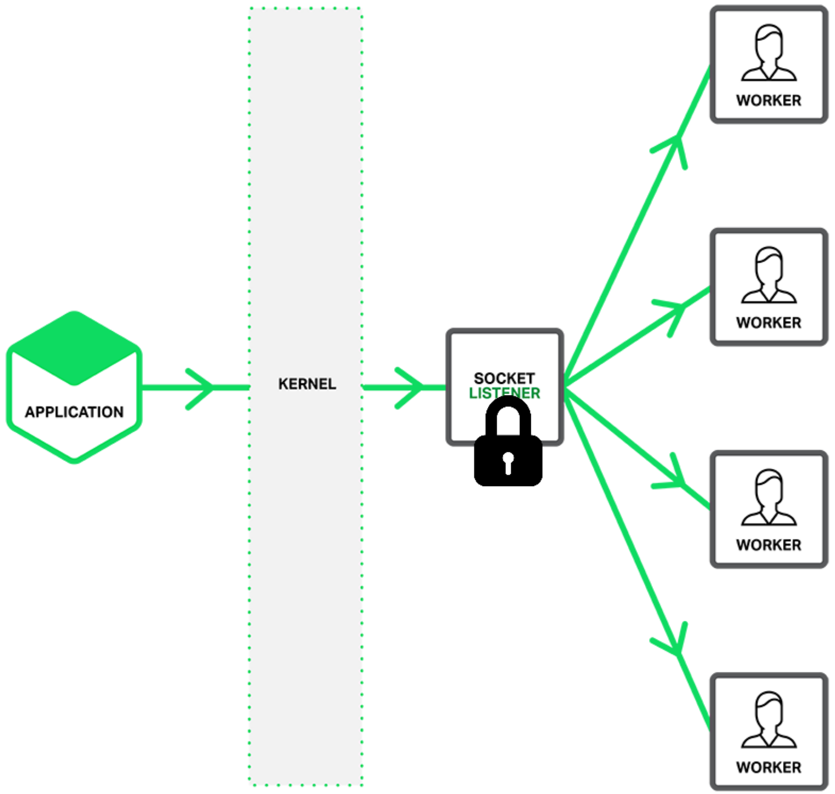
INTERNET REQUESTS
Read Request Headers
SSL and SPDY decrypt encrypted traffic
(Apply keepalive management and bandwidth shaping to all client traffic)
Log
Access and session log
Identify Configuration Block
Request matching and rewriting
Response Filters
gzip, SSI, headers, image_filter, sub, etc...
Internal redirects and subrequests
Apply Rate Limits
Rate and concurrency limiting
Perform Authentication
Internal and external access control
Generate Content
Generate response locally, or proxy/gateway to an upstream service
Concurrency Control
Keepalive Management
Response Caching
Load Balancing
Session Persistence
Health Checks and Retries
Upstream Services
HTTP, HTTPS, FastCGI, uWSGI, SCGI, memcached

How Nginx works

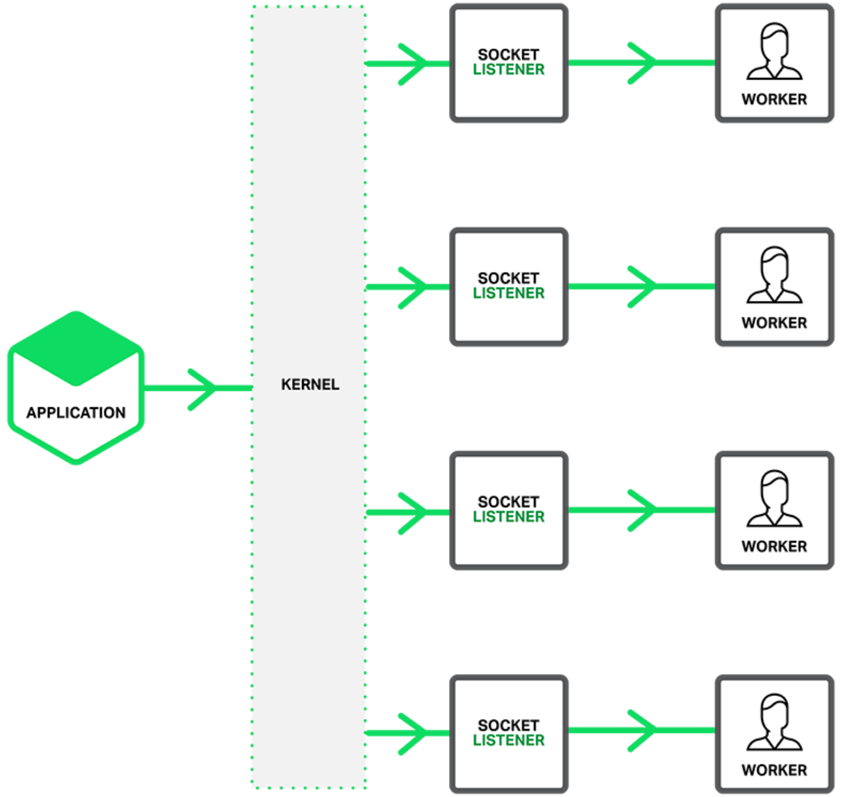




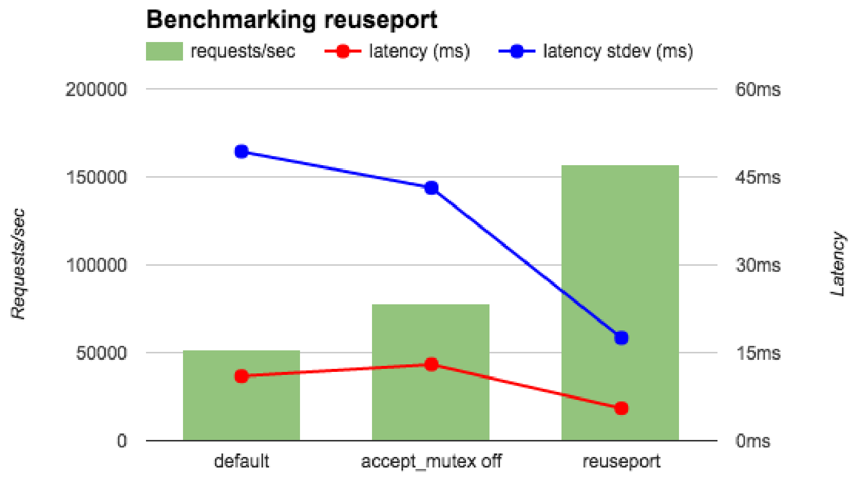
How Nginx works



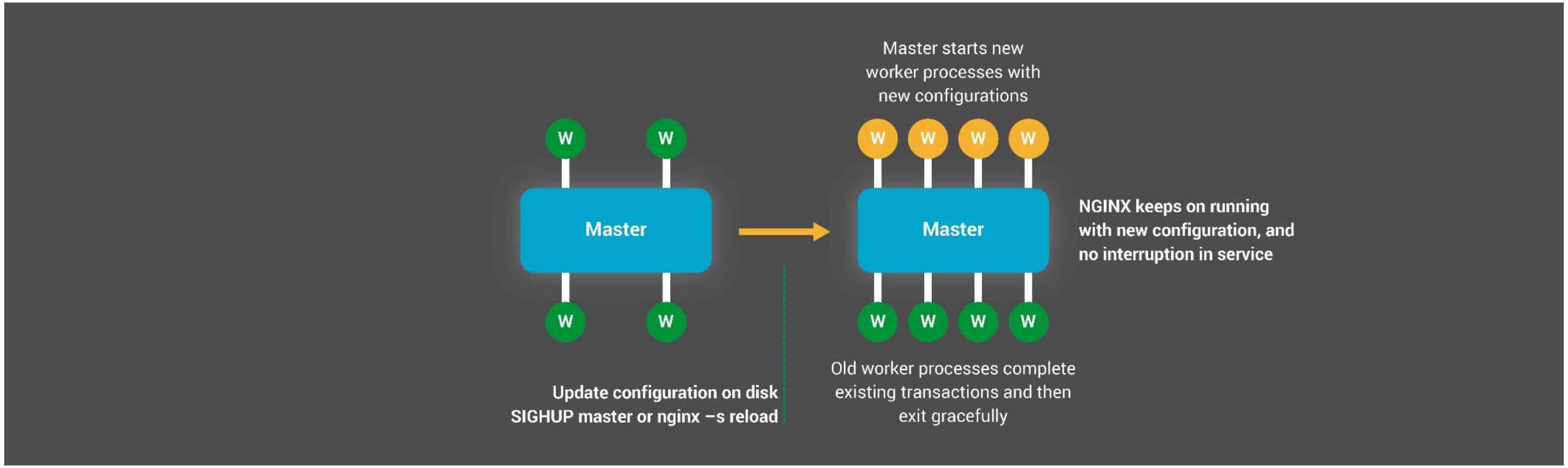
How Nginx works

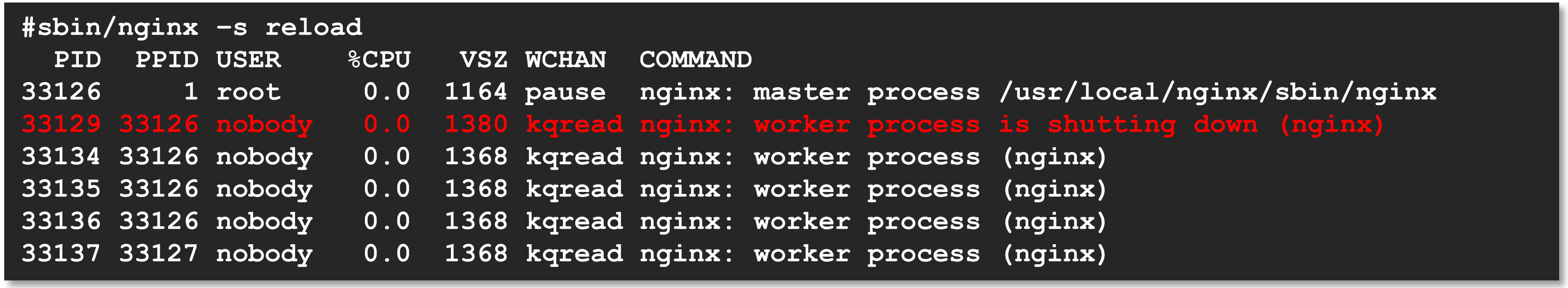


```
#Nginx 1.9.1
server {
          listen 80 reuseport;
          server_name  localhost;
          ...
     }
```



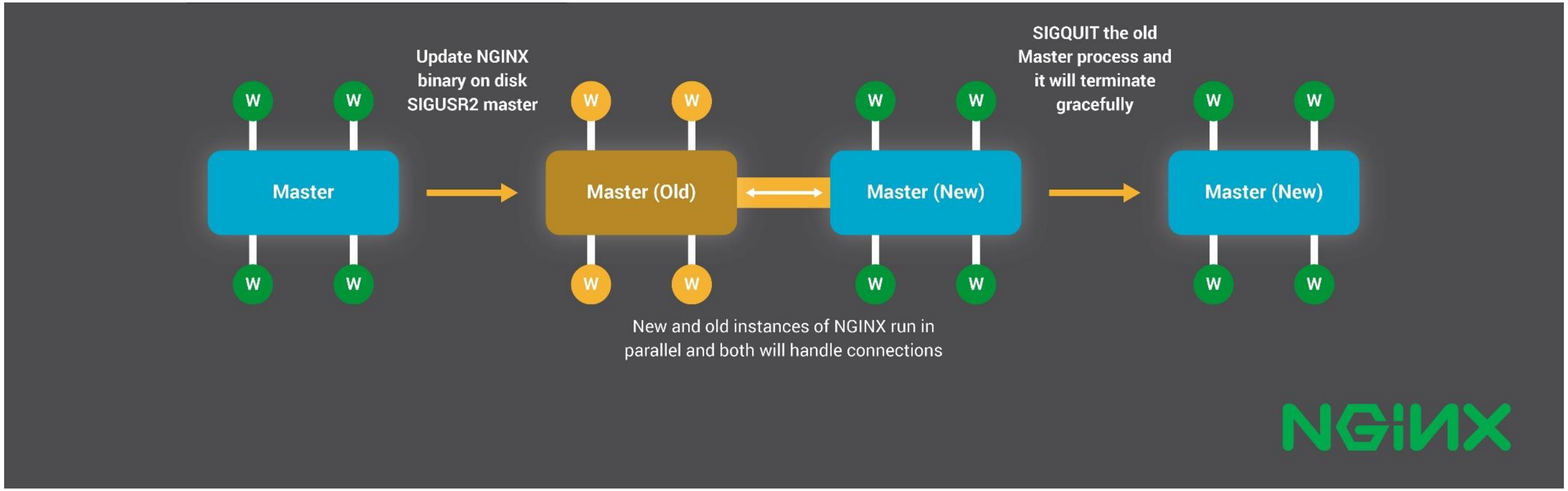
How Nginx works


Master starts new worker processes with new configurations
W
Master
Master
NGINX keeps on running with new configuration, and no interruption in service
Update configuration on disk SIGHUP master or nginx –s reload
Old worker processes complete existing transactions and then exit gracefully



```
#sbin/nginx -s reload
  PID  PPID USER    %CPU   VSZ WCHAN  COMMAND
33126     1 root     0.0  1164 pause  nginx: master process /usr/local/nginx/sbin/nginx
33129 33126 nobody   0.0  1380 kqread nginx: worker process is shutting down (nginx)
33134 33126 nobody   0.0  1368 kqread nginx: worker process (nginx)
33135 33126 nobody   0.0  1368 kqread nginx: worker process (nginx)
33136 33126 nobody   0.0  1368 kqread nginx: worker process (nginx)
33137 33127 nobody   0.0  1368 kqread nginx: worker process (nginx)
```

How Nginx works


Update NGINX binary on disk SIGUSR2 master
SIGQUIT the old Master process and it will terminate gracefully
W
Master
Master (Old)
Master (New)
Master (New)
New and old instances of NGINX run in parallel and both will handle connections
NGINX

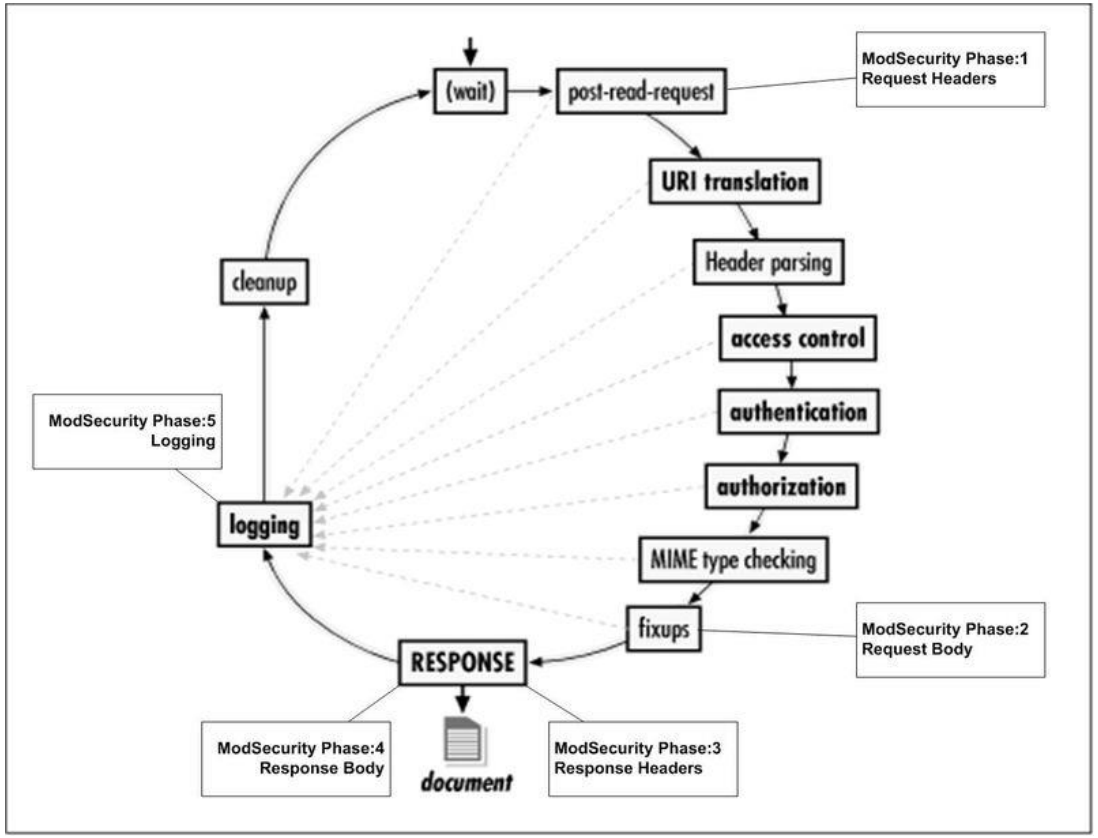
A Simple Question

```
location /test {
     set $a 32;

     echo $a;

     set $a 56;
}
```

```
set $a 32;

set $a 56;

echo $a;
```

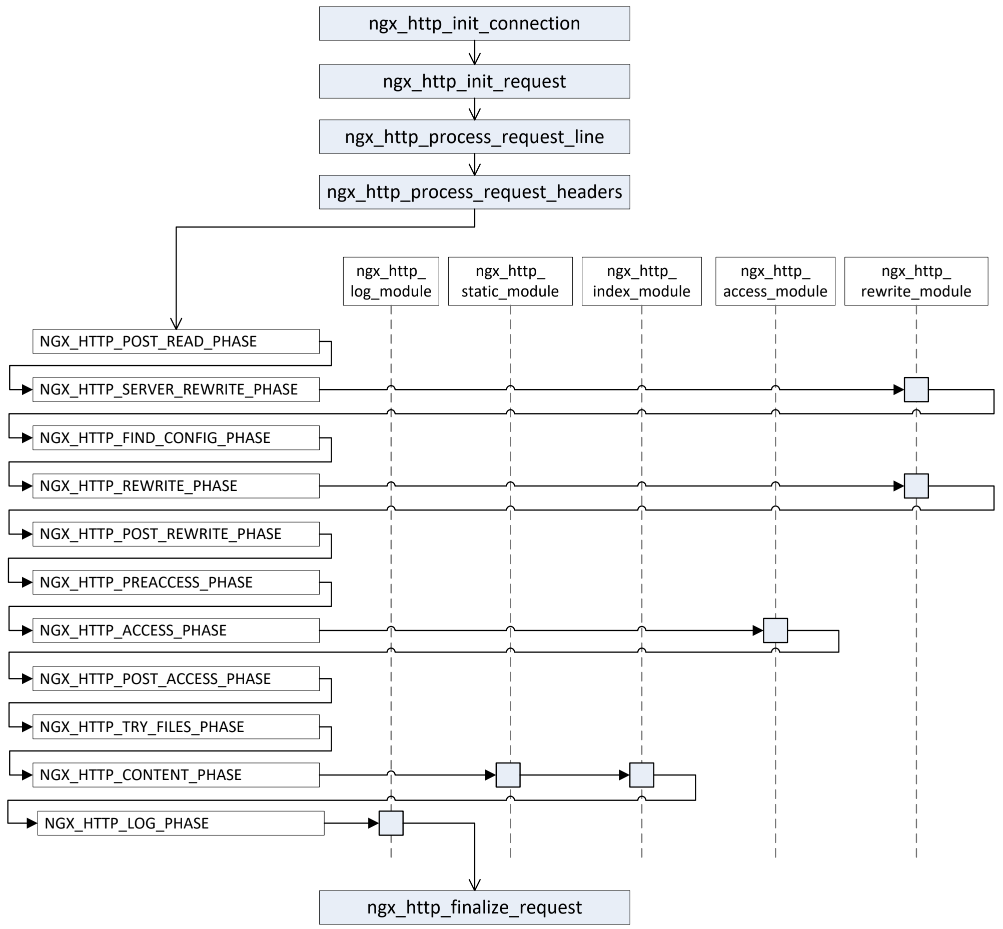
```
$ curl -i http://localhost/test
56
```

Nginx phase


(wait)
post-read-request
ModSecurity Phase:1 Request Headers
URI translation
Header parsing
access control
authentication
authorization
MIME type checking
fixups
ModSecurity Phase:2 Request Body
RESPONSE
ModSecurity Phase:3 Response Headers
document
ModSecurity Phase:4 Response Body
logging
ModSecurity Phase:5 Logging
cleanup

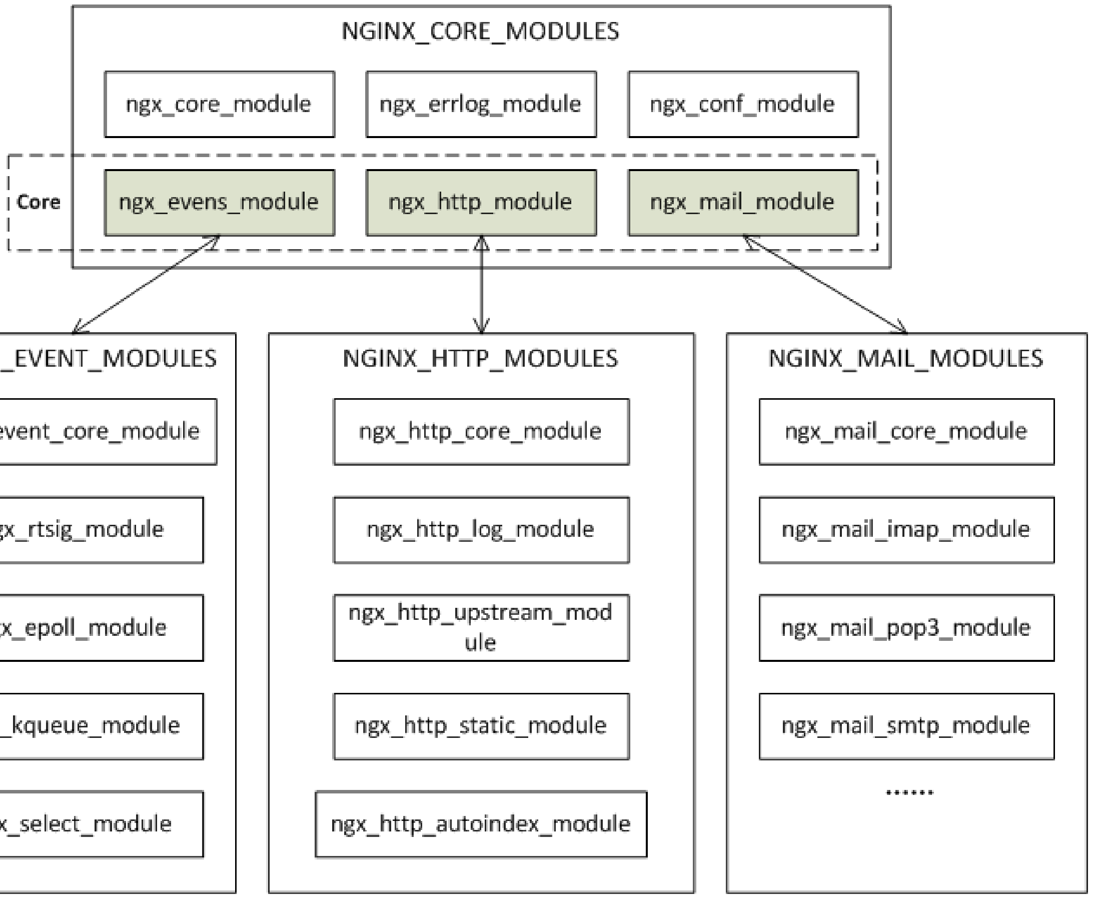
- NGX_HTTP_POST_READ_PHASE (realip)
- NGX_HTTP_SERVER_REWRITE_PHASE (Rewrite)
- NGX_HTTP_FIND_CONFIG_PHASE Can not be extended
- NGX_HTTP_REWRITE_PHASE (Rewite)
- NGX_HTTP_POST_REWRITE_PHASE Can not be extended
- NGX_HTTP_PREACCESS_PHASE (limit_req、limit_zone、degradation、realip)
- NGX_HTTP_ACCESS_PHASE (ACCESS 、auth_basic)
- NGX_HTTP_POST_ACCESS_PHASE Can not be extended
- NGX_HTTP_TRY_FILES_PHASE Can not be extended
- NGX_HTTP_CONTENT_PHASE (Autoindex、static、random_index、gzip_static)
- NGX_HTTP_LOG_PHASE

Nginx phase



- phase checker
- phase handler, array
- sub request

Nginx Modules



lua-nginx-module

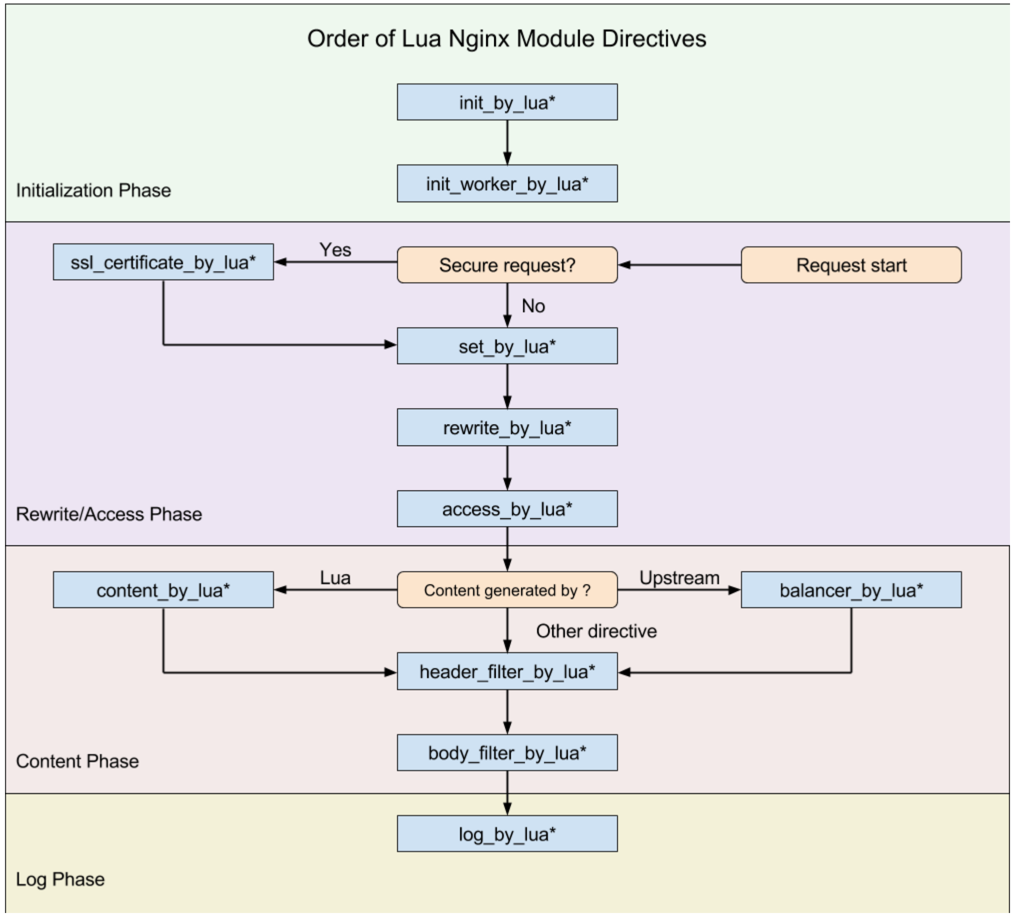
History

- Agentzh(章亦春)
- 阿里巴巴中国雅虎2007年8月至2009年8月
- 阿里巴巴淘宝网2009年8月至2011年7月
- OpenResty Inc

Why We Need It?

lua-nginx-module

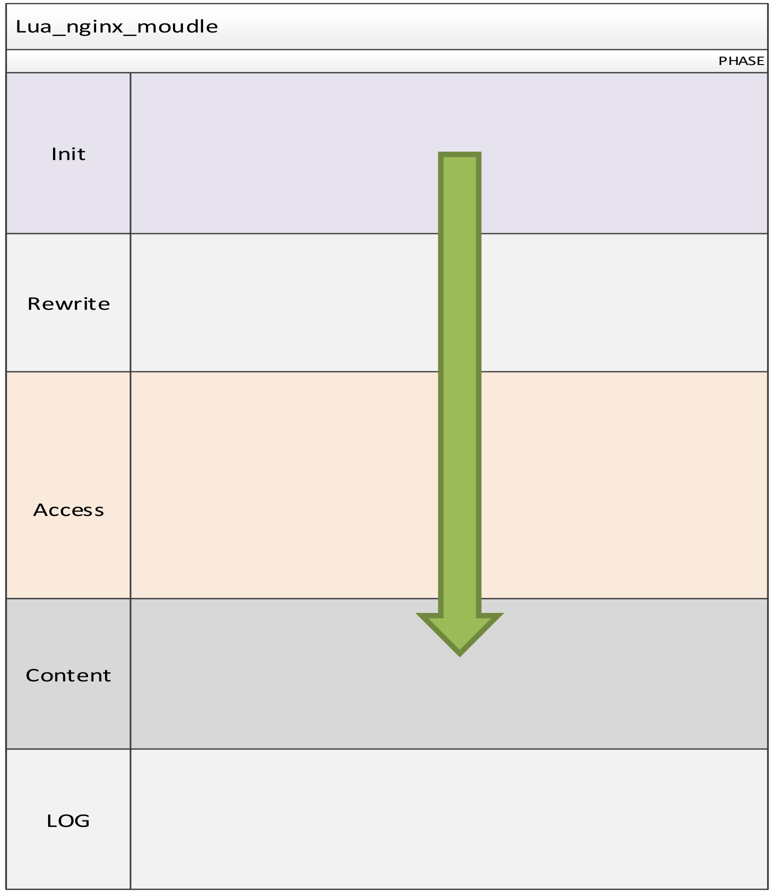
- ngx.shared.DICT
- TCP/UDP
- ngx.worker
- ngx.req
- …



lua-nginx-module

```
ngx.status = ngx.HTTP_GONE
ngx.say("HI")
-- when status >= 200, interrupt current  request
    return status code to nginx.
--When status == 0 ,
     quit the current phase,run next
ngx.exit(ngx.HTTP_OK)
```

```
$ curl -i http://localhost/test
HTTP/1.1 410 Gone
Server: nginx/1.0.6
Date: Thu, 15 Sep 2011 00:51:48 GMT
Content-Type: text/plain
Transfer-Encoding: chunked
Connection: keep-alive

HI
```



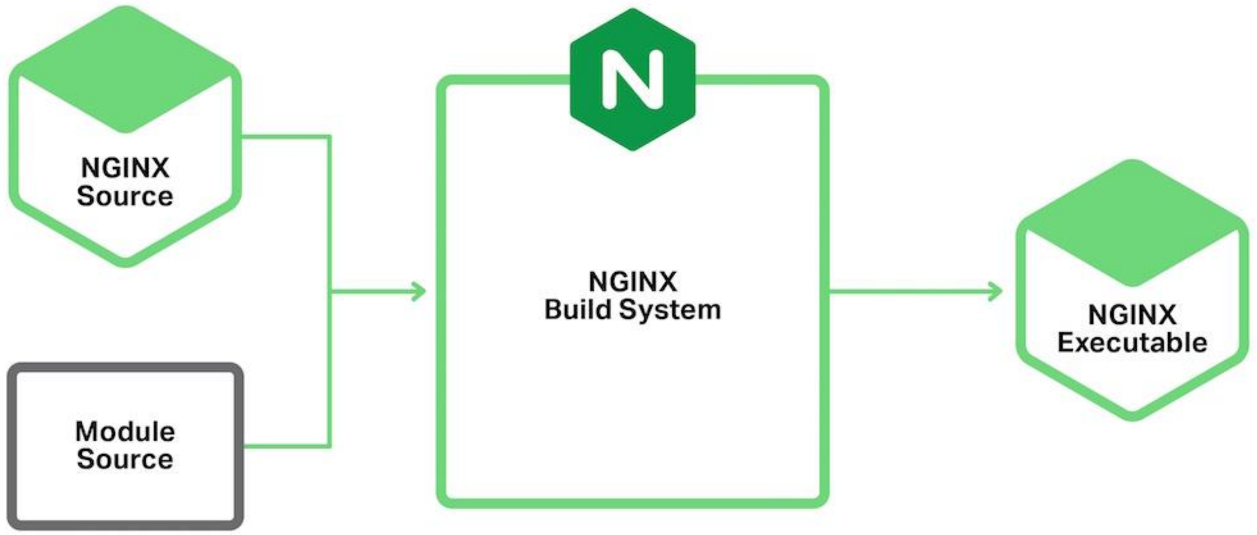
lua-nginx-module

```
local mysql = require "resty.mysql"
local memcached = require "resty.memcached"

 local function query_mysql()
     local db = mysql:new()
     db:connect{…}
     local res, err, errno, sqlstate = db:query(“select * from A limt 10")
     db:set_keepalive(0, 100)
     ngx.say("mysql done: ", cjson.encode(res))
 end

 local function query_memcached()
     local memc = memcached:new()
     memc:connect("127.0.0.1", 11211)
     local res, err = memc:get("some_key")
     ngx.say("memcached done: ", res)
 end

 local function query_http()
     local res = ngx.location.capture("/my-http-proxy")
     ngx.say("http done: ", res.body)
 end
```

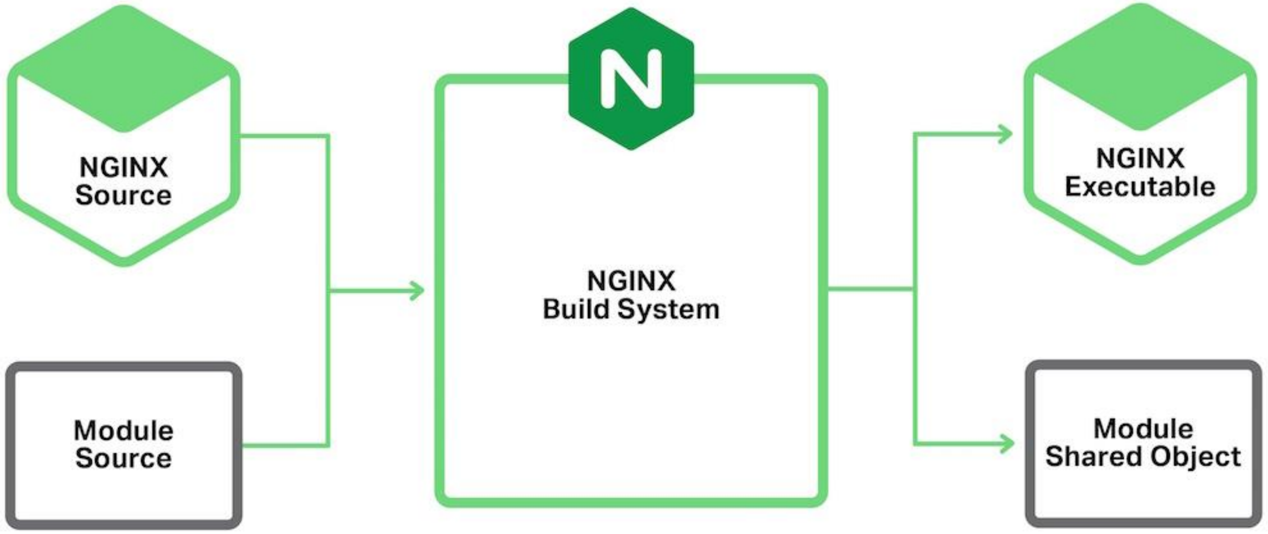
lua-nginx-module

```
wget http://luajit.org/download/LuaJIT-2.0.4.tar.gz
tar -xzf LuaJIT-2.0.4.tar.gz
cd LuaJIT-2.0.4
make install

wget https://github.com/openresty/lua-nginx-module/archive/v0.10.7.zip
wget https://github.com/simpl/ngx_devel_kit/archive/v0.3.0.tar.gz
wget http://nginx.org/download/nginx-1.11.2.tar.gz

 tar -xzf v0.10.7.tar.gz
 tar -xzf v0.3.0rc1.tar.gz
 tar -xzvf nginx-1.11.2.tar.gz
 cd nginx-1.11.2/

 # tell nginx's build system where to find LuaJIT 2.0:
 export LUAJIT_LIB=/usr/local/lib
 export LUAJIT_INC=/usr/local/include/luajit-2.0

 # or tell where to find Lua if using Lua instead:
 #export LUA_LIB=/path/to/lua/lib
 #export LUA_INC=/path/to/lua/include

 # Here we assume Nginx is to be installed under /opt/nginx/.
 ./configure --prefix=/home/work/local \
         --with-ld-opt="-Wl,-rpath,/path/to/luajit-or-lua/LuaJIT-2.0.4" \
         --add-module=/path/to/ngx_devel_kit \
         --add-module=/path/to/lua-nginx-module
 make -j2
 make install
```

Dynamic Modules



Dynamic Modules



Dynamic Modules

```
#Start Nginx Version 1.9.11
# ./configure --with-http_geoip_module=dynamic \
              --with-http_image_filter_module=dynamic \
              --with-mail=dynamic \
              --with-stream=dynamic \
              --with-http_xslt_module=dynamic

#Third-Party Modules
# ./configure --add-dynamic-module=/path/to/module/source
```

```
load_module "modules/ngx_http_geoip_module.so";
load_module "modules/ngx_stream_module.so";
```

- Location match
- Nginx load-balancer
- Nginx http2
- dynamic-modules
- Proxy cache
- **lua-nginx-module**

http://nginx.baidu.com/

THANKS

Bai du 百度