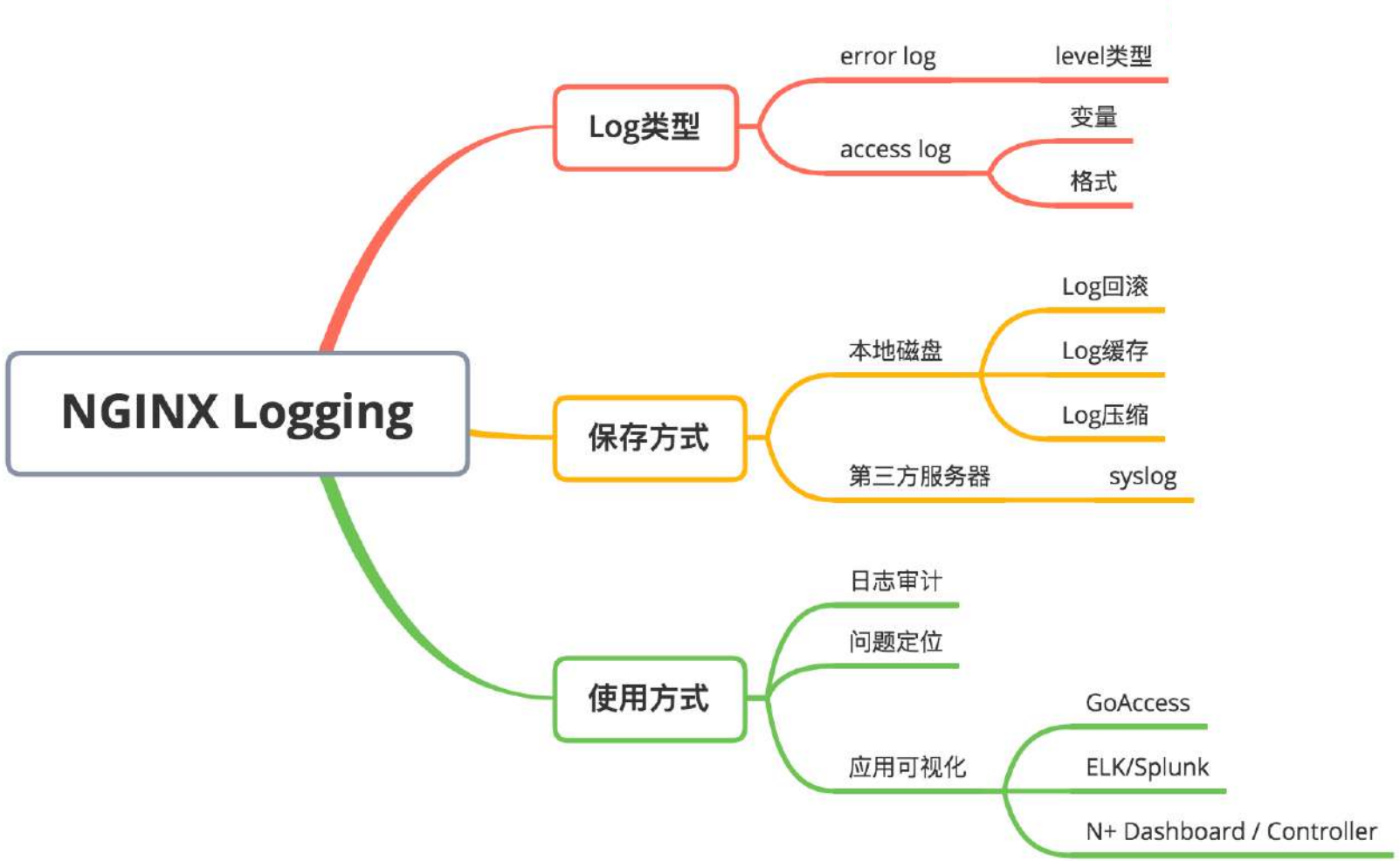
NGINX从入门到精通进阶系列培训

基础篇：个性化NGINX日志处理

廖健雄 j.liao@f5.com

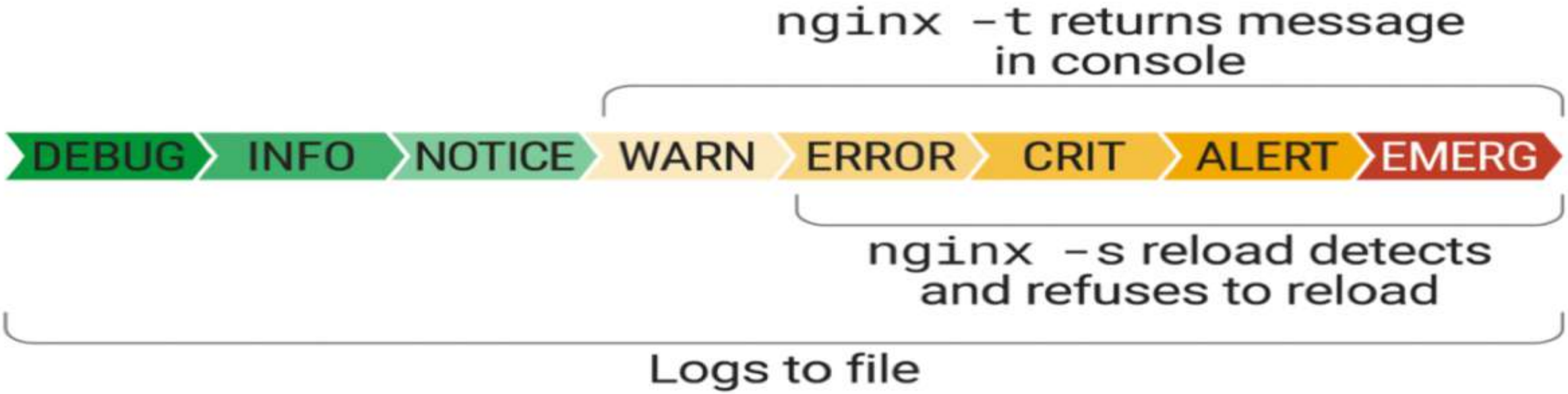
个性化NGINX日志处理



NGINX Logging类型

- error_log - 记录NGINX服务器所有运行信息
  - 记录信息细节取决于配置的level参数级别
- access_log - 记录访问NGINX服务器的流量细节
  - 流量记录细节取决于log_format指令配置

Error Log配置



error_log 指令

- 定义log格式
- 语法： **error_log** *file* [*level*];
- 默认： error_log logs/error.log error;
- 上下文： main, http, mail, stream, server, location

Error log 配置举例

```
server {
   listen 80;
   server_name server1.com;
   error_log /var/log/nginx/server1_info.log info;

   location /error {
      error_log /var/log/nginx/server1_error.log error;
   }
}
```

Error Log分类

- 类型1: upstream/client timed out
- 类型2: connect() failed
- 类型3: no live upstream
- 类型4: upstream/client prematurely closed connection
- 类型5: 104: Connection reset by peer
- 类型6: client intended to send too large body
- 类型7: upstream sent invalid HTTP header
- 类型8：SSL handshake mistake
- 类型9：others

类型	错误日志	原因	解决办法
1	upstream timed out (110: Connection timed out) while connecting to upstream	nginx与upstream建立tcp连接超时，nginx默认连接建立超时为200ms	排查upstream是否能正常建立tcp连接
1	upstream timed out (110: Connection timed out) while reading response header from upstream	nginx从upstream读取响应时超时，nginx默认的读超时为20s，读超时不是整体读的时间超时，而是指两次读操作之间的超时，整体读耗时有可能超过20s	排查upstream响应请求为什么过于缓慢
1	Client timed out (110: Connection timed out) while SSL handshaking	nginx与客户端建立SSL连接时，client 端超时	排查报错的流量占比，如较高可适当调高相应的time out参数。
1	client timed out (110: Connection timed out) while waiting for request	Nginx与客户端建立tcp连接后，等待客户端发送request的过程中，客户端发生超时	排查报错的流量占比，如较高可适当调高相应的time out参数。
2	connect() failed (104: Connection reset by peer) while connecting to upstream	nginx与upstream建立tcp连接时被reset	排查upstream是否能正常建立tcp连接
2	connect() failed (111: Connection refused) while connecting to upstream	nginx与upstream建立tcp连接时被拒	排查upstream是否能正常建立tcp连接
2	(111: Connection refused) while sending request to upstream	Nginx和upstream连接成功后发送request时，若遇到后端upstream挂掉或者不通，会收到该错误	排查upstream server的状态
3	no live upstreams while connecting to upstream	nginx向upstream转发请求时发现upstream状态全都为down	排查nginx的upstream的健康检查为什么失败
4	upstream prematurely closed connection	nginx在与upstream建立完tcp连接之后，试图发送请求或者读取响应时，连接被upstream强制关闭	排查upstream程序是否异常，是否能正常处理http请求
4	Client prematurely close connection (104: Connection reset by peer) while sending to client	Nginx在与client建立完tcp连接之后，试图发送响应g给客户端时，连接被客户端强制关闭	一般是正常现象，比如客户端异常关闭。排查该异常的流量占比。
5	recv() failed (104: Connection reset by peer) while reading response header from upstream	nginx从upstream读取响应时连接被对方reset	排查upstream应用tcp连接状态是否异常
5	peer closed connection in SSL handshake (104: Connection reset by peer) while SSL handshaking	nginx与client在进行ssl 连接握手过程中，客户端reset了连接	一般是正常现象，比如客户端异常关闭。排查该异常的流量占比。
5	writev() failed (104: Connection reset by peer) while sending to client	nginx在生成响应返回客户端时连接被对方reset	一般是正常现象，比如客户端异常关闭。排查该异常的流量占比。
6	client intended to send too large body	客户端试图发送过大的请求body，nginx默认最大允许的大小为1m，超过此大小，客户端会收到http 413错误码	1.调整请求客户端的请求body大小； 2.调大相关域名的nginx配置：client_max_body_size；
7	upstream sent invalid header while reading response header from upstream	nginx不能正常解析从upstream返回来的响应头	排查upstream应用配置
8	SSL_do_handshake() failed	SSL握手失败	排查nginx ssl相关配置
8	could not add new SSL session to the session cache while SSL handshaking	ssl_session_cache配置参数过小不满足需求	增大ssl_session_cache配置参数
8	ngx_slab_alloc() failed: no memory in SSL session shared cache	ssl_session_cache配置参数过小不满足需求	增大ssl_session_cache配置参数
9	client closed keepalive connection	客户端正常关闭与nginx的连接	正常现象，如不希望见到此类告警，调整error log到notice级别及以上。

access log配置

```
Syntax:   access_log path [format [buffer=size] [gzip[=level]] [flush=time] [if=condition]];
          access_log off;
Default:  access_log logs/access.log combined;
Context:  http, server, location, if in location, limit_except
```

- path：本地磁盘文件路径/ syslog服务器地址
- format：log日志格式，由log_format指令进行配置
- buffer：批量将内存中的日志写入磁盘
- gzip：批量压缩内存中的日志，再写入磁盘
- flush：指定时间周期写入磁盘
- if：符合条件才记录日志

举例：
access_log /path/to/log.gz combined gzip flush=5m;

access_log syslog:server=192.168.1.1 debug;

log_format指令

Syntax: **log_format** *name* [escape=default|json|none] *string* ...;
Default: log_format combined "...";
Context: http

默认的combined日志格式：

'$remote_addr - $remote_user [$time_local] ' '"$request" $status $body_bytes_sent ' '"$http_referer" "$http_user_agent"';

10.1.10.1 - - [18/Feb/2020:17:20:26 +0800] "GET / HTTP/1.1" 200 612 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.106 Safari/537.36"

10.1.10.1 - - [18/Feb/2020:17:20:46 +0800] "GET /index.html HTTP/1.1" 200 612 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/13.0.5 Safari/605.1.15"

10.1.10.1 - - [18/Feb/2020:17:20:46 +0800] "GET /favicon.ico HTTP/1.1" 404 153 "http://graydemo/index.html" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/13.0.5 Safari/605.1.15"

日志回滚

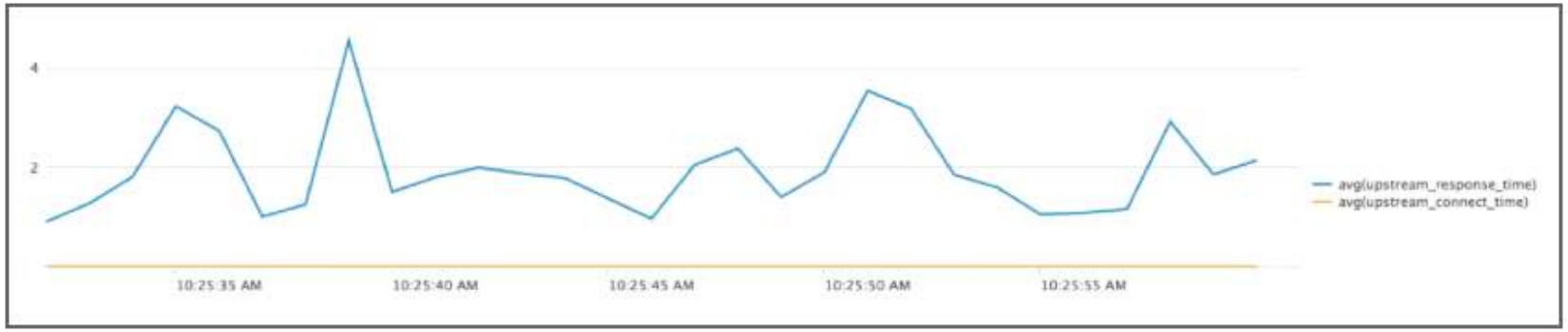
通过logrotate管理NGINX日志文件

```
/var/log/nginx/*.log {
        daily
        missingok
        rotate 52
        compress
        delaycompress
        notifempty
        create 640 nginx adm
        sharedscripts
        postrotate
                if [ -f /var/run/nginx.pid ]; then
                        kill -USR1 `cat /var/run/nginx.pid`
                fi
        endscript
}
```

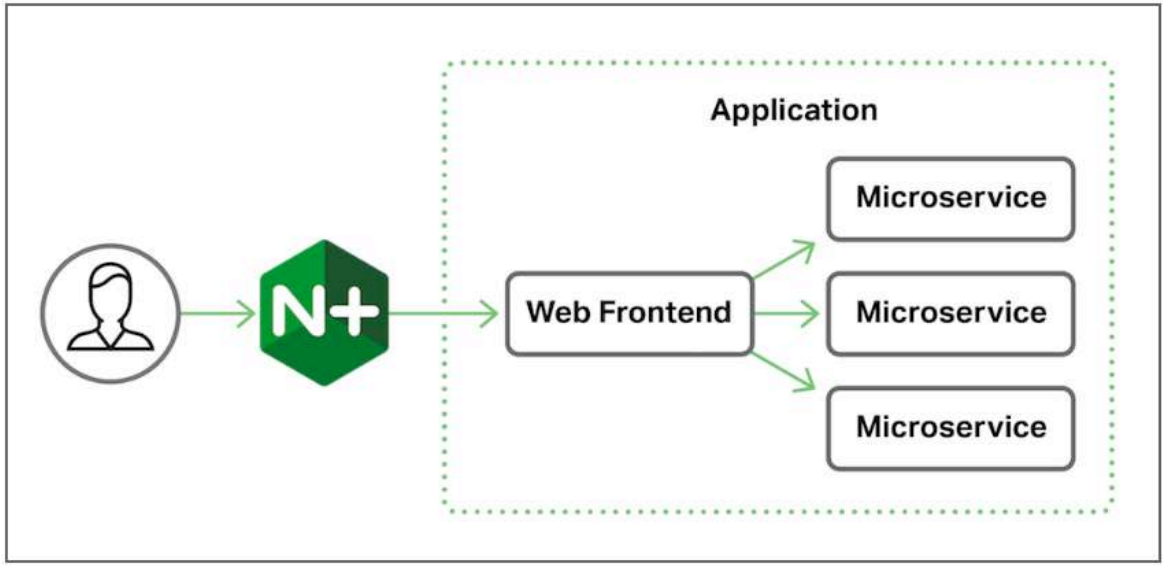
```
[root@rp1 nginx]# ls
access.log                  error.log
access.log-20191216.gz      error.log-20191216.gz
access.log-20191217.gz      error.log-20191217.gz
access.log-20191218.gz      error.log-20191218.gz
access.log-20200105.gz      error.log-20200105.gz
access.log-20200117.gz      error.log-20200117.gz
access.log-20200118.gz      error.log-20200118.gz
access.log-20200119.gz      error.log-20200119.gz
access.log-20200205.gz      error.log-20200120.gz
access.log-20200206.gz      error.log-20200205.gz
access.log-20200207.gz      error.log-20200206.gz
access.log-20200213.gz      error.log-20200213.gz
access.log-20200214.gz      error.log-20200214.gz
```

个性化NGINX日志之应用性能监控（APM）

```
log_format apm ' “$time_local” client=$remote_addr '
        'method=$request_method request="$request” '
        'request_length=$request_length '
        'status=$status bytes_sent=$bytes_sent '
        'body_bytes_sent=$body_bytes_sent '
        'referer=$http_referer '
        'user_agent="$http_user_agent" '
        'upstream_addr=$upstream_addr '
        'upstream_status=$upstream_status '
        'request_time=$request_time '
        'upstream_response_time=$upstream_response_time '
        'upstream_connect_time=$upstream_connect_time '
        'upstream_header_time=$upstream_header_time';
```



个性化NGINX日志之请求追踪



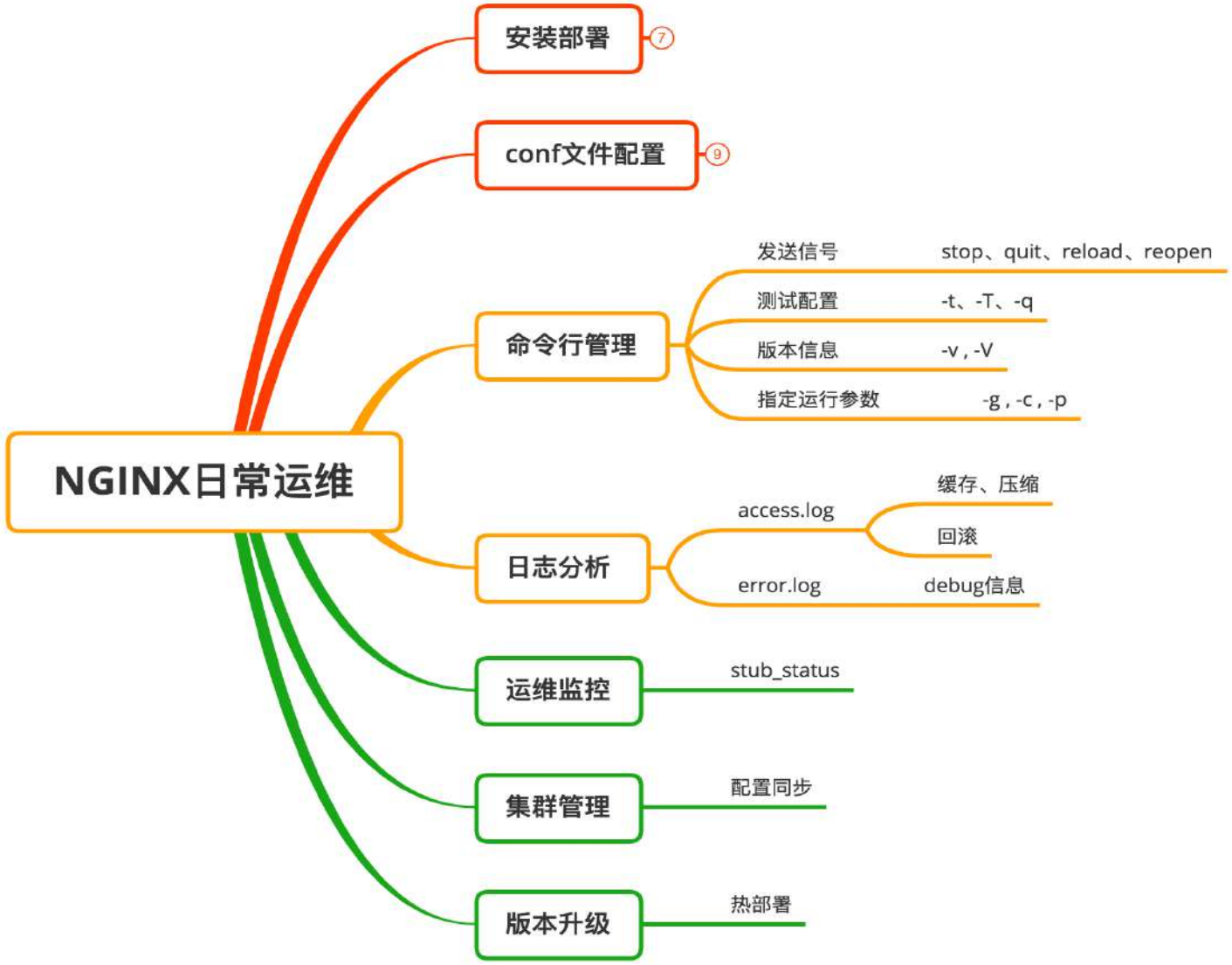

```
log_format trace '$remote_addr - $remote_user [$time_local] "$request" ' '$status
$body_bytes_sent "$http_referer" "$http_user_agent" ' '"$http_x_forwarded_for" $request_id' ;

server {
    listen 80;
    location / {
        proxy_pass http://app_server;
        proxy_set_header X-Request-ID $request_id; # Pass to app
        access_log /var/log/nginx/access_trace.log trace;
    }
}
```

NGINX从入门到精通进阶系列培训

基础篇：轻松掌握NGINX日常运维

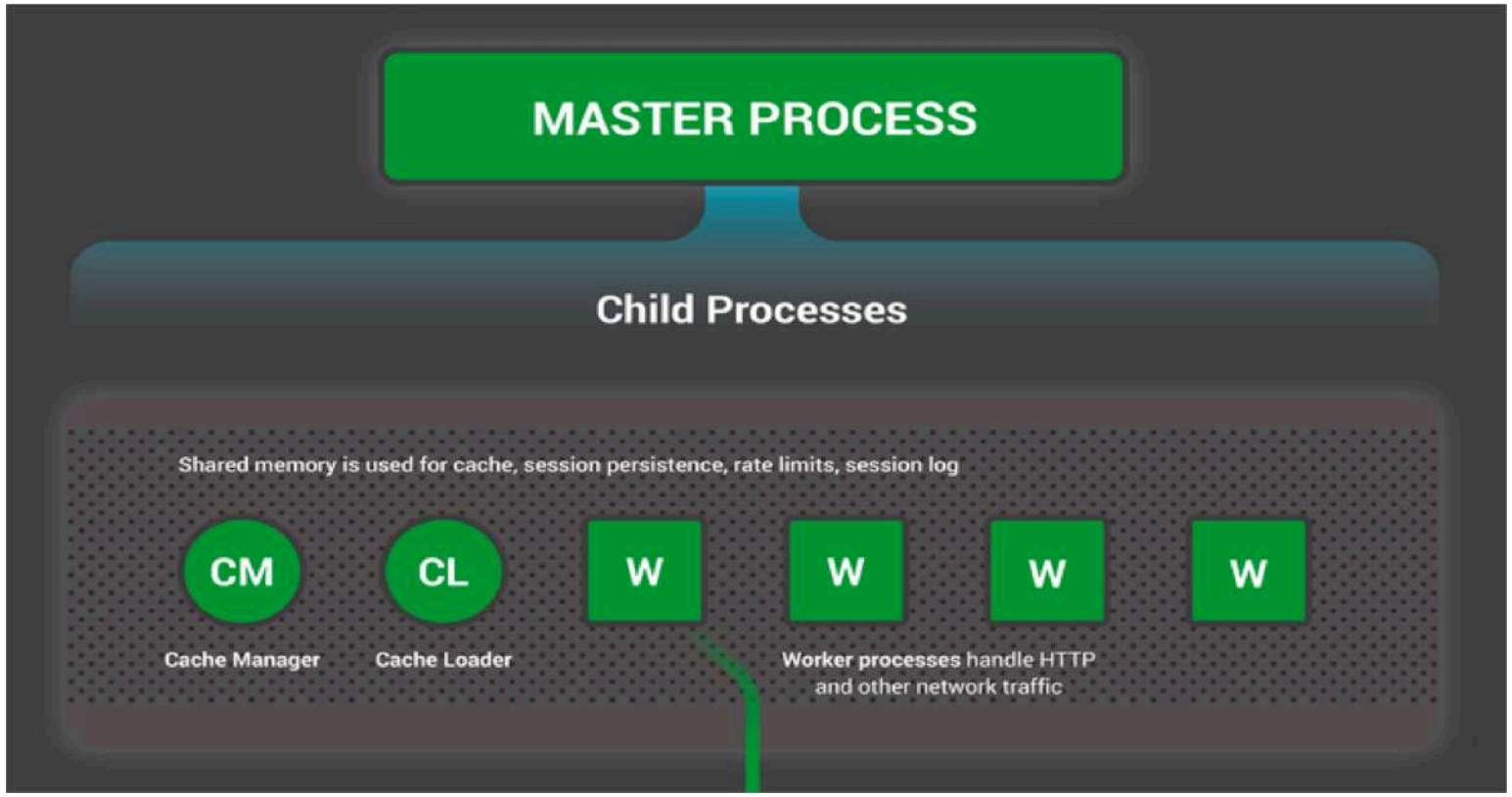
廖健雄 j.liao@f5.com

轻松掌握NGINX日常运维



NGINX进程结构

- Master进程：
  - 初始化工作
  - 接收管理员信号
  - 管理子进程
- Work子进程：
  - 处理请求
- Cache相关进程：
  - Cache Manage负责管理cache
  - Cache Loader负责载入cache



NGINX命令行

- 帮助：-? –h
- 使用指定的配置文件：-c
- 指定配置指令：-g
- 指定运行目录：-p
- 发送信号： -s
- 测试配置文件是否有语法错误：-t –T
- 显示NGINX版本信息、编译模块信息等：-v -V

立刻停止服务：stop

优雅停止服务：quit

重载配置文件：reload

重新开始记录日志文件：reopen

NGINX 进程管理：信号

Master进程

- 监控worker进程
  - CHLD
- 接收信号
  - TERM,INT
  - QUIT
  - HUP
  - USR1
  - USR2
  - WINCH

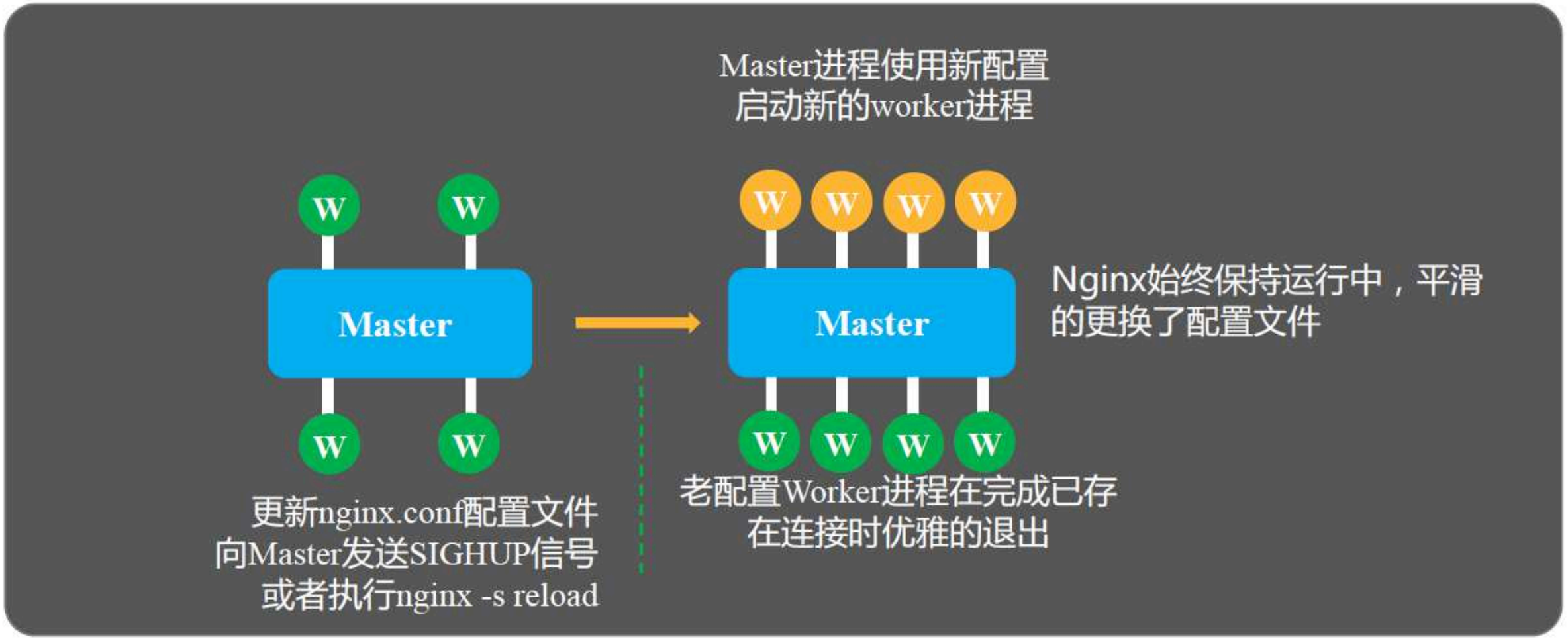
Worker进程

- 接收信号
  - TERM,INT
  - QUIT
  - HUP
  - WINCH

NGINX命令行

- reload: HUP
- reopen: USR1
- stop: TERM
- quit: QUIT

Reload流程


Master进程使用新配置
启动新的worker进程
W
W
W
W
Master
W
W
W
W
Master
W
W
W
W
Nginx始终保持运行中，平滑
的更换了配置文件
更新nginx.conf配置文件
向Master发送SIGHUP信号
或者执行nginx -s reload
老配置Worker进程在完成已存
在连接时优雅的退出

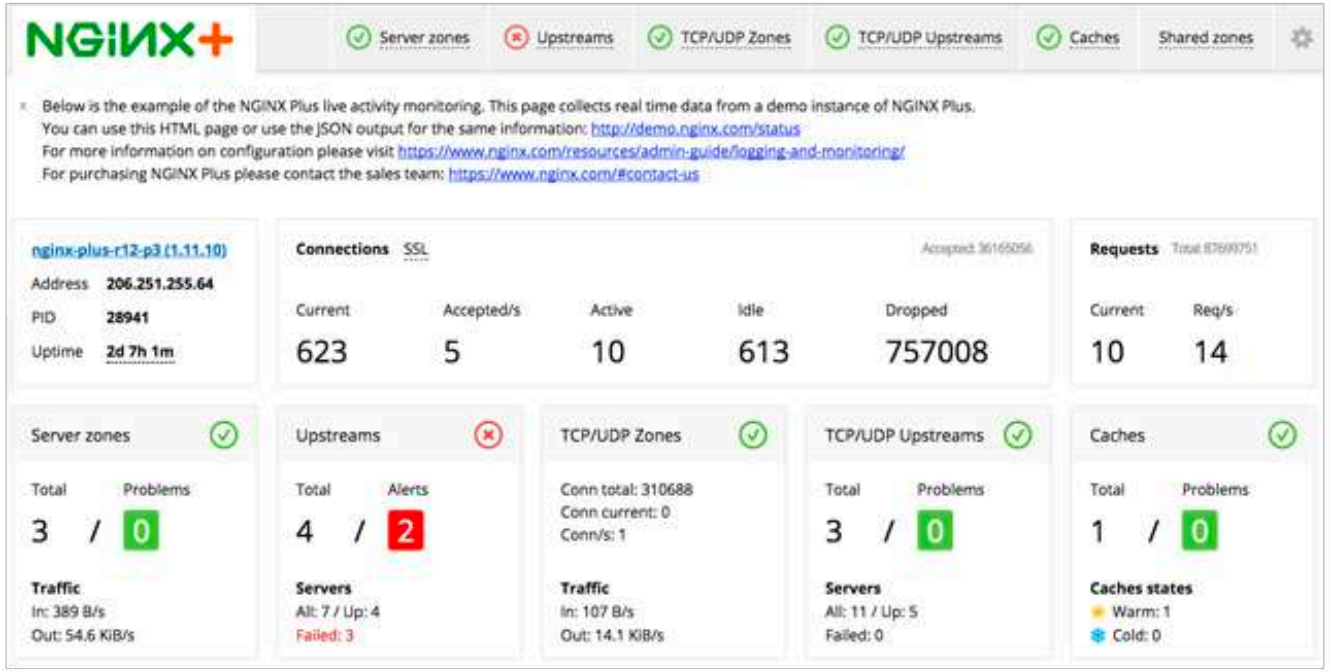
运维监控

不安全 | 10.1.10.143/stub_status

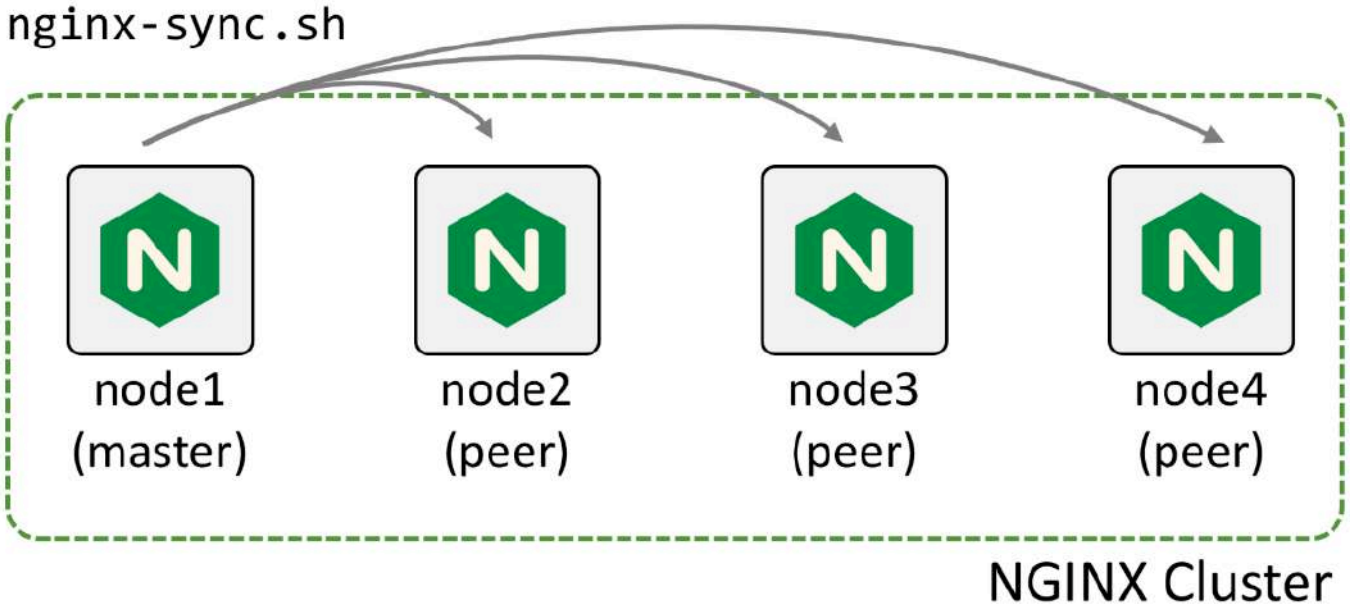
```
Active connections: 2
server accepts handled requests
 23438 23438 82520
Reading: 0 Writing: 1 Waiting: 1
```

- 仅有7个性能参数指标

- 内建实时的图形dashboard
- 高达100多个实时性能参数指标
- 支持以JSON和HTML输出格式集成到客户监控平台



集群管理



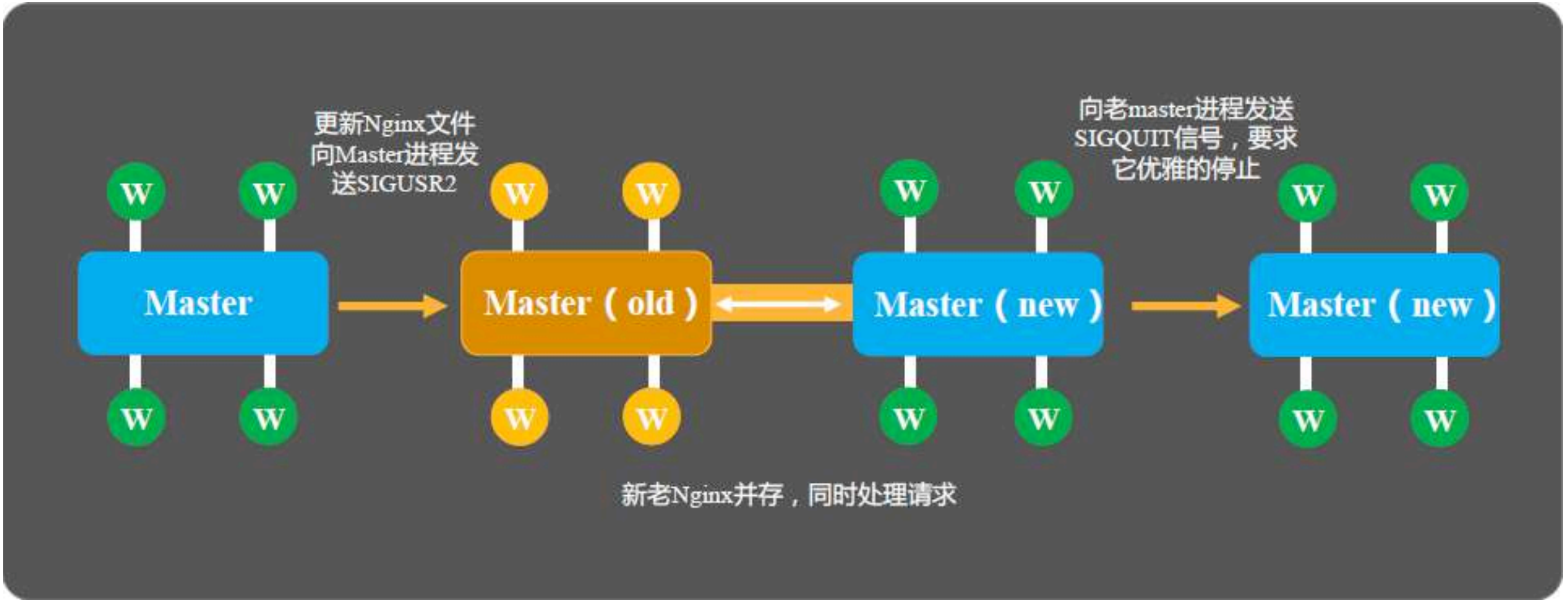
- master节点安装nginx-sync组件
- 配置各peer节点支持master节点无密码ssh登录
- master配置nginxy-sync组件

```
NODES="node2.example.com node3.example.com node4.example.com"
CONFPATHS="/etc/nginx/nginx.conf /etc/nginx/conf.d"
EXCLUDE="default.conf"
```

- master节点通过执行nginx-sync.sh脚本同步配置并且reload 远端NGINX节点

https://docs.nginx.com/nginx/admin-guide/high-availability/configuration-sharing/

热升级流程


更新Nginx文件
向Master进程发
送SIGUSR2
向老master进程发送
SIGQUIT信号，要求
它优雅的停止
Master
Master（old）
Master（new）
Master（new）
W
新老Nginx并存，同时处理请求

关注我们

F5 官方微信
（新闻，技术文章）

F5社区
（答疑，吐槽，分享，互动）

加入F5社区：关注“F5社区”微信公众号，注册成为社员，随时参加meet up活动，代码共享，讨论，答疑等。只要你有想法，有创造，那么就快来大展身手吧，让我们在社区里尽情分享，交流，吐槽和互动，在这个自由的国度里，发现闪亮的自己。让我们一起来见证“一群有才能的人在一起做有梦想的事！

NGINX技术群

操作步骤：

1. 扫描二维码并在“入群信息”栏填写姓名
2. 点击下方“我要入群”
3. 长按识别二维码进入群聊


f5
N
Thank You