NGINX从入门到精通基础篇

presenter :林健 Sr System Engineer

NGINX概要

NGINX 软件概述

Igor Sysoev

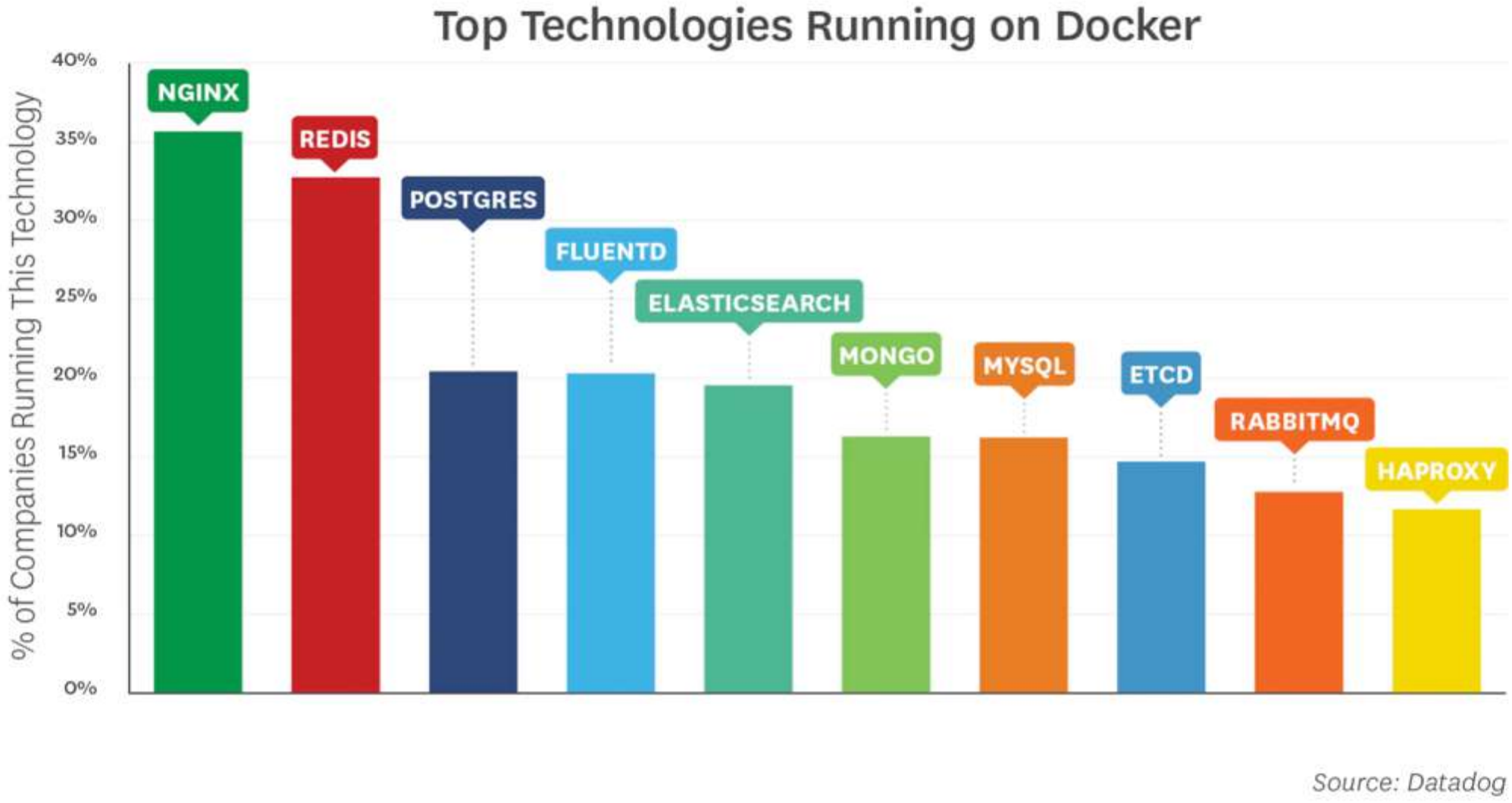
Job title : Senior Architect
Company : F5 Networks, Inc.

Nginx是C语言编写的服务器程序，可以作为WEB服务器，反向代理，负载均衡组件，邮件代理，以及HTTP缓存服务器，初始作者为Igor Sysoev，首次发布版本为2004年10月4日，其开源软件协议为2-clause BSD，可以运行在多种BSD，HP-UX，IBM AIX，Linux MacOS，Solaris，Windows。

Nginx初衷为解决WEB服务器C-10K问题，即并发连接1万，在线程模型下难以实现，Nginx使用Linux 内核的异步事件驱动机制（EPOLL）构建出高性能，轻量化的服务器

负载均衡　API网关　安全　微服务　WEB和移动　云应用

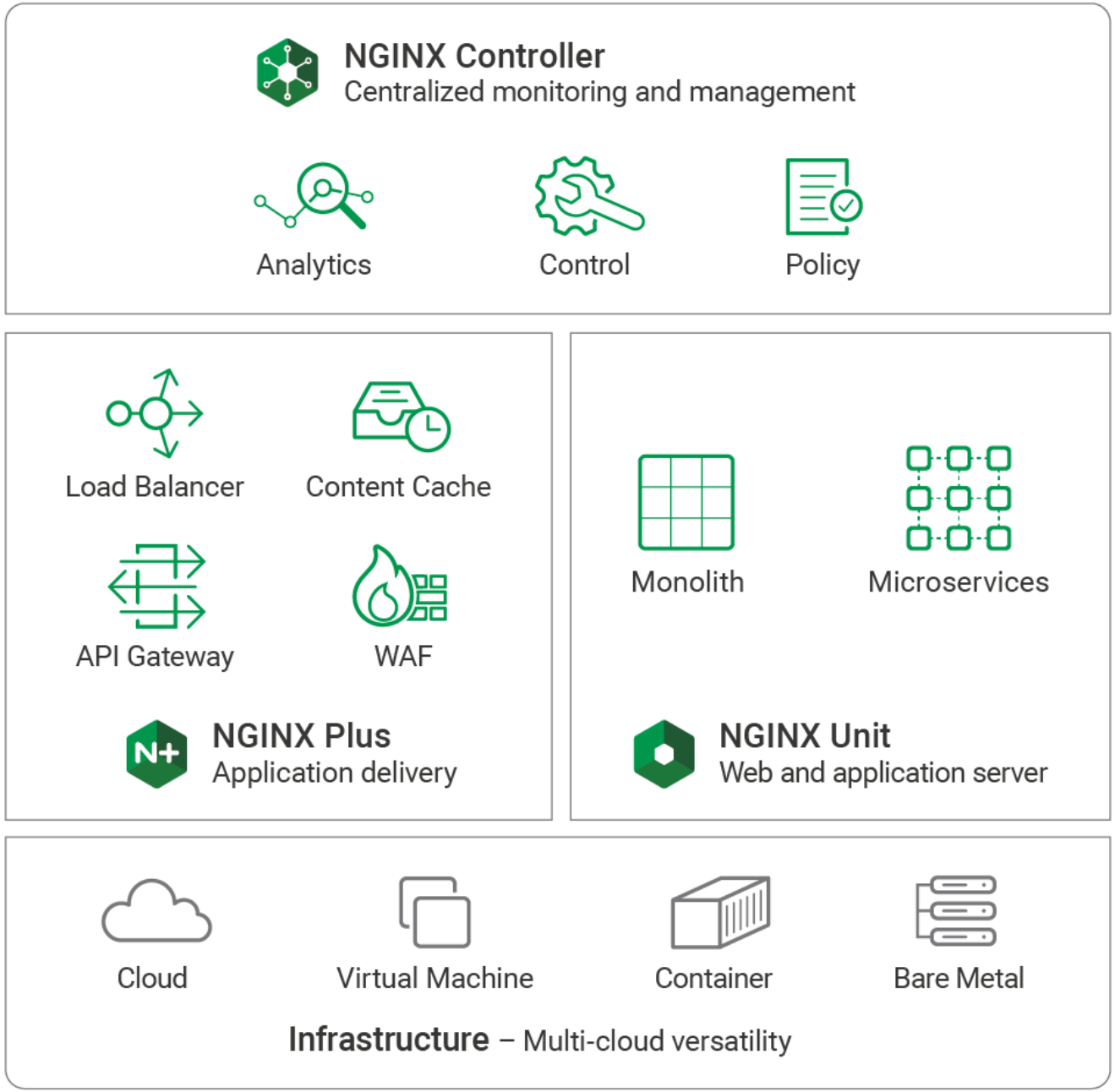
Nginx发展历程



jinx，

NGINX Application Platform

NGINX是一套用于开发和提供数字化体验的技术，范围从传统应用程序到现代的微服务应用程序。



NGINX 2019 highlight

- OSS 11项新功能发布
- 专注在负载均衡，代理以及SSL可扩展
- 新的用户案例（FIPS-140-2，PASV FTP)
- 持续扩展NJS
- Security-HTTP/2 CVEs

NGINX Roadmap

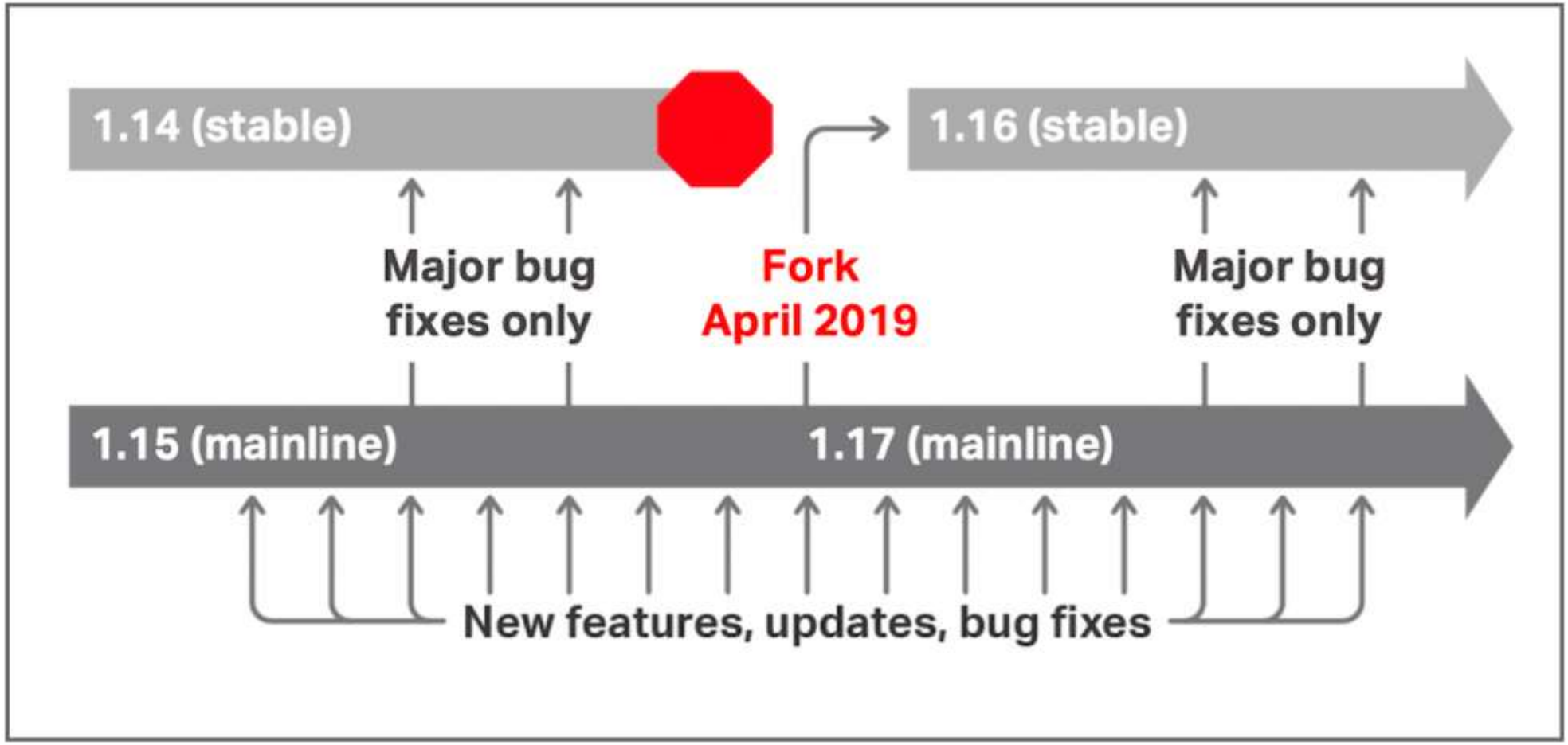
- 支持QUIC/HTTP3
- 将NGINX PLUS更多的统计信息整合到OSS
- 持续增加对nJS的投入

NGINX 2019 Update

	NGINX Plus R17 (1.15.7) Dec 18	NGINX Plus R18 (1.15.10) May 19	NGINX Plus R19 (1.17) Sept 19
OSS	高级Rate limit nJS语言扩展	动态证书加载 支持FTP PASV nJS模块	Rate limits Dry-run测试 动态带宽限制 nJS语言扩展
PLUS	OpenID Connect automation NGINX WAF Module	OpenID Connect-opaque Tokens,session tokens logut Health Checks and TCP connection control 动态KV配置	更多统计信息 (location,DNS,state sharing) Prometheus support KV store: IP/CIDR支持

NGINX安装

NGINX版本说明



- 1.16.1以上/1.17.3以上版本修复HTTP2的漏洞

NGINX获取方式

软件仓库

debian

源码安装

目前Centos/Debian/Ubuntu在默认软件仓库中包含nginx（out of date），Suse和Alpine Linux等无法通过系统默认软件仓库安装，需要添nginx.org/nginx.com的软件仓库

软件仓库安装操作系统支持

Version	Supported Platforms
6.x	x86_64, i386
7.4+	x86_64, ppc64le
8.x	x86_64

Version	Supported Platforms
16.04	x86_64, i386, ppc64el, aarch64/arm64x86_64, i386
18.04	x86_64, i386, ppc64el, aarch64/arm64
19.04/10	x86_64

Version	Supported Platforms
SLES 12	x86_64,
SLES 15	x86_64

Version	Supported Platforms
9.x	x86_64, i386
10.x	x86_64, i386

使用操作系统软件仓库


```
$sudo apt-get update
$sudo apt-get install nginx
```


```
$sudo yum install epel-release
$sudo yum update
$sudo yum install nginx
```

截止2020.2,Yum目前为1.16版本,NGINX仓库为1.17

NGINX OSS仓库：CentOS/RHEL

修改yum配置文件，添加添加软件仓库配置

```
$ sudo vi /etc/yum.repos.d/nginx.repo
```

输入如下内容,<OS>为rhel或centos,<OSRELEASE>为OS发行版本号，如6, 6._x_, 7, 7._x_

```
[nginx]
name=nginx repo
baseurl=https://nginx.org/packages/mainline/<OS>/<OSRELEASE>/$basearch/
gpgcheck=0 enabled=1
```

Update软件仓库，安装NGINX并运行

```
$ sudo yum update
$ sudo yum install nginx
$ sudo nginx
```

NGINX OSS仓库： Debian/Ubuntu

下载NGINX的软件包和软件仓库的签名key

```
$ sudo wget https://nginx.org/keys/nginx_signing.key
$ sudo apt-key add nginx_signing.key
```

修改sources.list文件

```
$ sudo vi /etc/apt/sources.list
```

添加如下内容，其中CODENAME为debian发行版代码，如debina 8.2为jessie，ubuntu 14.04位trusty

```
deb https://nginx.org/packages/mainline/debian/ <CODENAME> nginx deb-src
https://nginx.org/packages/mainline/debian/ <CODENAME> nginx
```

安装并运行

```
$ sudo apt-get remove nginx-common
$ sudo apt-get update
$ sudo apt-get install nginx
$ sudo nginx
```

NGINX OSS仓库： SUSE

增加NGINX stable软件仓库并安装

```
$zypper addrepo -G -t yum -c
'https://nginx.org/packages/sles/12' nginx
$ zypper install nginx
```

增加NGINX mainline软件仓库并安装

```
$zypper addrepo -G -t yum -c
'https://nginx.org/packages/mainline/sles/12' nginx
$ zypper install nginx
```

源码安装-1：安装依赖库

PCRE – Supports regular expressions. Required by the NGINX Core and Rewrite modules.

```
$ wget ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre/pcre-8.43.tar.gz
$ tar -zxf pcre-8.43.tar.gz
$ cd pcre-8.43
$ ./configure
$ make
$ sudo make install
```

zlib – Supports header compression. Required by the NGINX Gzip module.

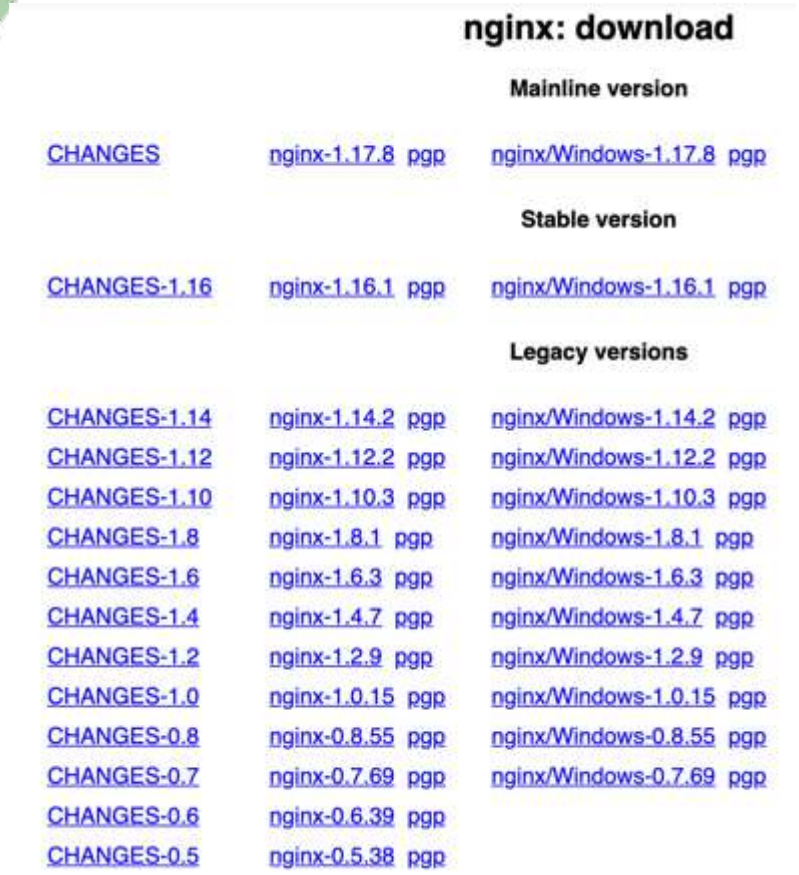
```
$ wget http://zlib.net/zlib-1.2.11.tar.gz
$ tar -zxf zlib-1.2.11.tar.gz
 $ cd zlib-1.2.11
$ ./configure
 $ make $ sudo make install
```

源码安装-1：安装依赖库

OpenSSL – Supports the HTTPS protocol. Required by the NGINX SSL module and others.

```
$ wget http://www.openssl.org/source/openssl-1.1.1c.tar.gz
$ tar -zxf openssl-1.1.1c.tar.gz $ cd openssl-1.1.1c
$ ./Configure darwin64-x86_64-cc --prefix=/usr
$ make $ sudo make install
```

源码安装-2： 获取NGINX源码



nginx: download

Mainline version

CHANGES	nginx-1.17.8 pgp	nginx/Windows-1.17.8 pgp

Stable version

CHANGES-1.16	nginx-1.16.1 pgp	nginx/Windows-1.16.1 pgp

Legacy versions

CHANGES-1.14	nginx-1.14.2 pgp	nginx/Windows-1.14.2 pgp
CHANGES-1.12	nginx-1.12.2 pgp	nginx/Windows-1.12.2 pgp
CHANGES-1.10	nginx-1.10.3 pgp	nginx/Windows-1.10.3 pgp
CHANGES-1.8	nginx-1.8.1 pgp	nginx/Windows-1.8.1 pgp
CHANGES-1.6	nginx-1.6.3 pgp	nginx/Windows-1.6.3 pgp
CHANGES-1.4	nginx-1.4.7 pgp	nginx/Windows-1.4.7 pgp
CHANGES-1.2	nginx-1.2.9 pgp	nginx/Windows-1.2.9 pgp
CHANGES-1.0	nginx-1.0.15 pgp	nginx/Windows-1.0.15 pgp
CHANGES-0.8	nginx-0.8.55 pgp	nginx/Windows-0.8.55 pgp
CHANGES-0.7	nginx-0.7.69 pgp	nginx/Windows-0.7.69 pgp
CHANGES-0.6	nginx-0.6.39 pgp	
CHANGES-0.5	nginx-0.5.38 pgp	

nginx.org/en/download.html

```
$ wget https://nginx.org/download/nginx-
1.17.6.tar.gz
$ tar zxf nginx-1.17.6.tar.gz
$ cd nginx-1.17.6
```

源码安装-3： Config/Make

开源经典三部曲：configure/make/make install/

```
$ ./configure
--sbin-path=/usr/local/nginx/nginx
--conf-path=/usr/local/nginx/nginx.conf
--pid-path=/usr/local/nginx/nginx.pid
--with-pcre=../pcre-8.43
--with-zlib=../zlib-1.2.11
--with-http_ssl_module
--with-stream
--with-mail=dynamic
--add-module=/usr/build/nginx-rtmp-module
--add-dynamic-module=/usr/build/3party_module
```

Configure 最后编译并安装即可

```
$ make
$ sudo make install
$ sudo nginx
```

NGINX模块概念

NGINX =

NGINX由主程序以及模块构成，NGINX的功能通过模块提供，配置文件中的指令和变量由模块提供，A用户的配置文件拷贝到B用户上不一定可以启动NGINX

默认编译模块

NGINX由核心功能以及一系列的模块构成，下面的模块默认会编译在nginx内，
在configure命中添加--without-<MODULE-NAME> 可以取消某个模块的编译

```
http_access_module
http_auth_basic_module
http_autoindex_module
http_browser_module
http_charset_module
http_empty_gif_module
http_fastcgi_module
http_geo_module
http_gzip_module
http_limit_conn_module
http_limit_req_module
http_map_module
http_scgi_module http_ssi_module
http_split_clients_module
http_upstream_hash_module
http_upstream_ip_hash_module
http_upstream_keepalive_module
http_upstream_least_conn_module
http_upstream_zone_modul
http_userid_module
http_uwsgi_module
http_memcached_module
http_proxy_module
http_referer_module
http_rewrite_module
```

重要的非默认预编译模块

Module Name	Description
-- with-http_ssl_module	Enables HTTPS support. Requires an SSL library such as OpenSSL.
-- with-http_stub_status_module	Provides access to basic status information. Note that NGINX Plus customers do not require this module as they are already provided with extended status metrics and interactive dashboard.
-- with-http_v2_module	Enable support for HTTP/2.
-- with-stream	Enables the TCP and UDP proxy functionality. To compile as a separate dynamic module instead, change the option to --with-stream=dynamic.

非默认编译模块

对于不在默认编译中的模块，可以在configure过程中，使用—with-< MODULE-NAME >添加

```
$ ./configure
--sbin-path=/usr/local/nginx/nginx
--conf-path=/usr/local/nginx/nginx.conf
--pid-path=/usr/local/nginx/nginx.pid
--with-pcre=../pcre-8.43
--with-zlib=../zlib-1.2.11
--with-http_ssl_module
–-with-stream
--with-mail
```

其中 mail, stream, geoip, image_filter, perl , xslt 可以编译为动态模块

NGINX Dynamic Modules

- NGINX 1.9 /NGINX PLUS R9开始支持动态模块,可以在nginx.conf中使用指令调用
- NGINX-PLUS用户可以通过Nginx.com的软件仓库获取获取

```
$ apt-get install nginx-plus-module-<name>
```

- OSS社区获取开源的模块

```
$ wget http://nginx.org/download/nginx-1.15.10.tar.gz
$ tar -xzvf nginx-1.15.10.tar.gz
$ cd nginx-1.15.10/
$ ./configure --with-compat –add-dynamic-module=../nginx-foo-modules
$ make modules
```

- 配置文件中模块加载命令

```
$ load_module modules/module-name.so;
```

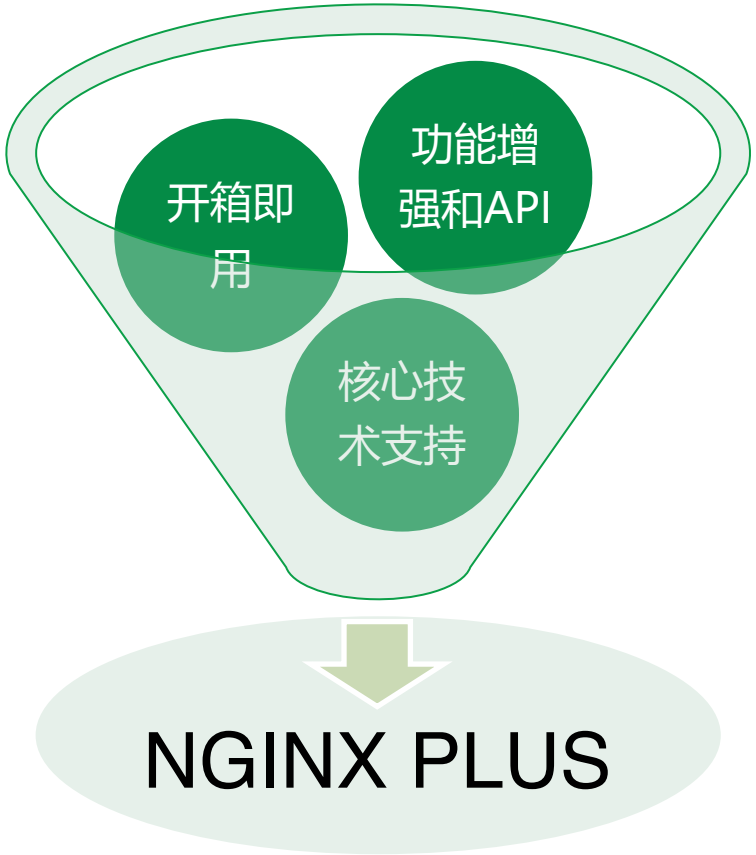
- Centos/ubuntu的官方软件仓库中有部分预编译好的动态模块，可以yum/apt安装

NGINX PLUS

NGINX PLUS是由社区核心开发成员成立的NGINX.INC发布的商业化版本NGINX，并提供技术支持和顾问咨询服务

NGINX PLUS预编译模块

--with-http_realip_module	--with-http_gzip_static_module
--with-http_secure_link_module	--with-http_hls_module
--with-http_session_log_module	--with-http_mp4_module
--with-http_slice_module	--with-http_random_index_module
--with-http_ssl_module	--with-mail_ssl_module
--with-http_stub_status_module	--with-stream
--with-http_sub_module	--with-stream_realip_module
--with-http_v2_module	--with-stream_ssl_module
--with-mail	--with-stream_ssl_preread_module
--with-compat	--with-http_auth_request_module
--with-file-aio	--with-http_dav_module
--with-threads	--with-http_f4f_module
--with-http_addition_module	--with-http_flv_module
--with-http_auth_jwt_module	--with-http_gunzip_module



NGINX PLUS安装

从Nginx.com获取证书以及key，并且安装在yum软件仓库,根据操作系统不同，输入软件仓库路径

```
$ sudo mkdir /etc/ssl/nginx $ cd /etc/ssl/nginx
$ sudo cp nginx-repo.crt /etc/ssl/nginx/ $ sudo cp nginx-repo.key /etc/ssl/nginx/
$ sudo yum install ca-certificates
$ sudo wget -P /etc/yum.repos.d https://cs.nginx.com/static/files/nginx-plus-7.4.repo
```

运行yum install 安装，并启动

```
$ sudo yum install nginx-plus
$ sudo systemctl enable nginx.service
```

SELinux/Firewall

Issue 1: Proxy Connection is Forbidden

Issue 2: File Access is Forbidden

Issue 3: NGINX Cannot Bind to Additional Port

详情可以参考以下链接，修改SELinux参数或者停止SELinux

https://www.nginx.com/blog/using-nginx-plus-with-selinux/

配置文件

Key NGINX Files and Directories

/etc/nginx/ 默认主配置文件

nginx.conf

```
http {

    include conf.d/*.conf;
}
```

全局配置
(tunings, logs, etc)

HTTP block

/etc/nginx/conf.d/

virtualserver1.conf

```
server {
    listen <parameters>;

    location <url> {
        ...
    }
}

upstream { … }
```

请求监听

请求处理规则

后端服务器配置
（可选）

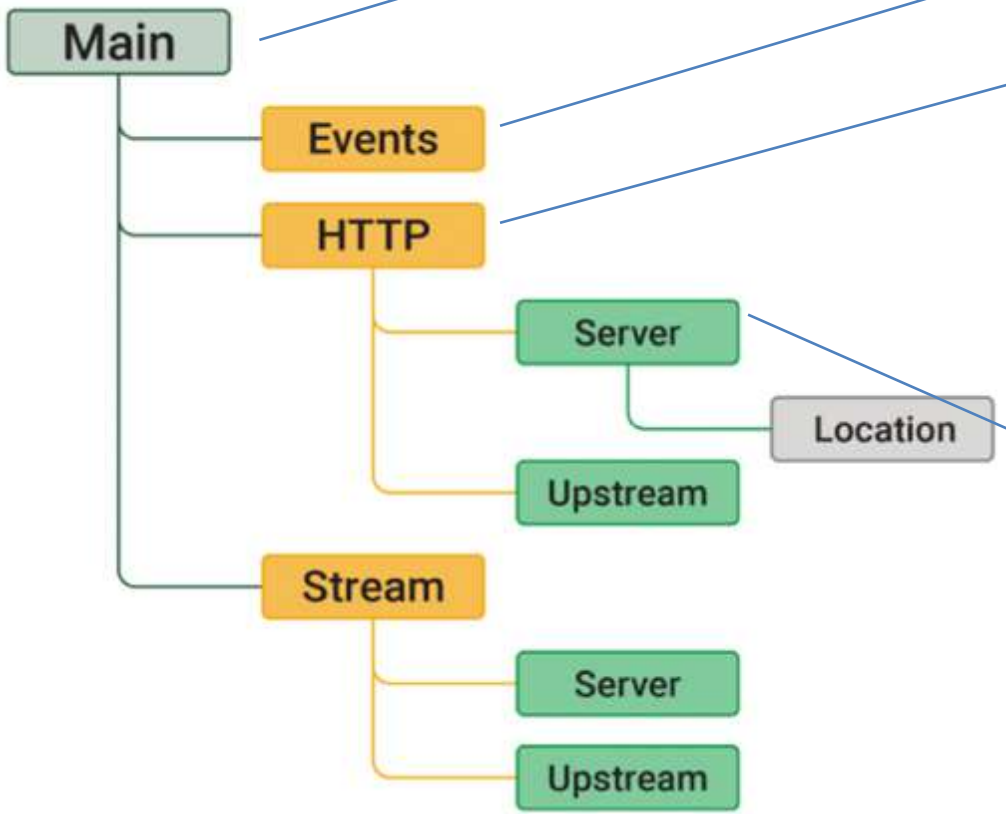
/var/log/nginx/

error.log 错误日志

access.log 每次请求记录，可配置格式

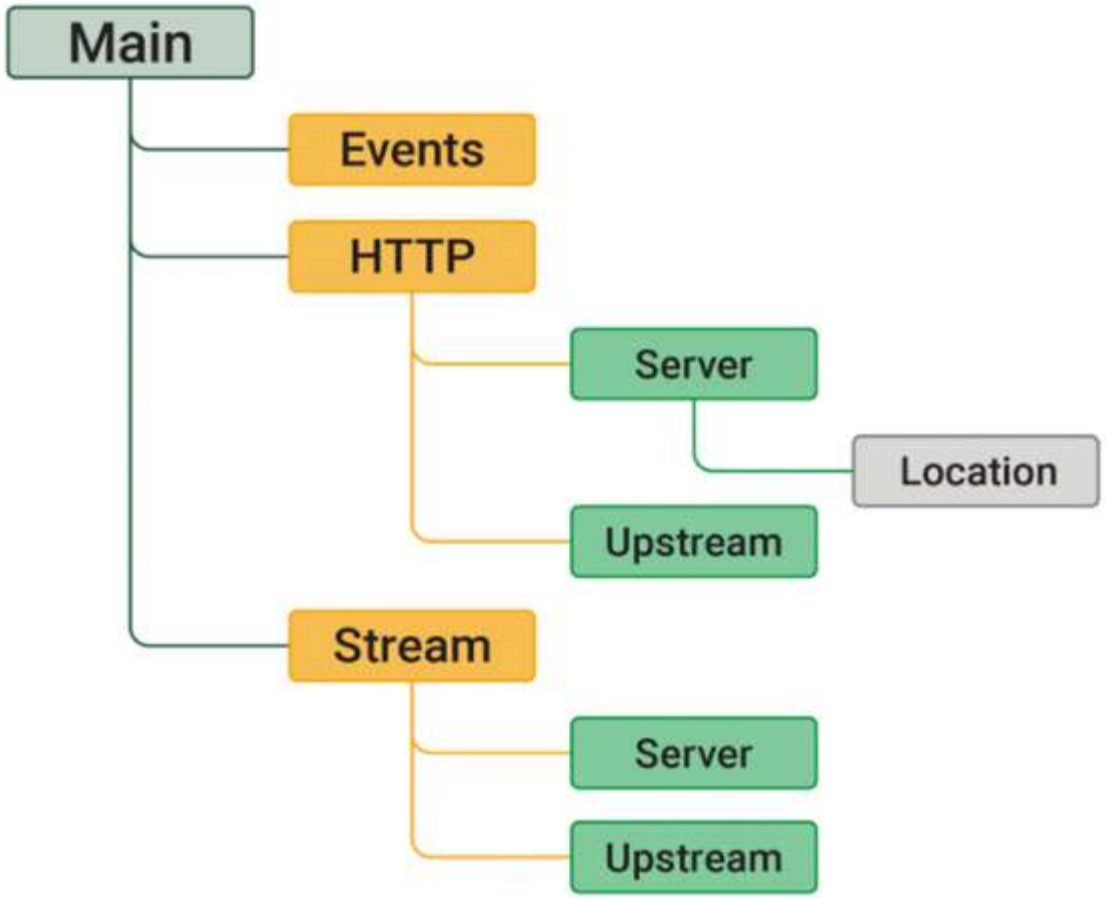
*nginx –c 命令可以指定启动文件的位置

nginx.conf example




```
user       www www;  ## Default: nobody
worker_processes  5;  ## Default: 1
error_log  logs/error.log;
pid        logs/nginx.pid;
worker_rlimit_nofile 8192;

events {
  worker_connections  4096;  ## Default: 1024
}

http {
  include    conf/mime.types;
  include    /etc/nginx/proxy.conf;
  include    /etc/nginx/fastcgi.conf;
  index    index.html index.htm index.php;

  default_type application/octet-stream;
  log_format   main '$remote_addr - $remote_user [$time_local]  $status '
    '"$request" $body_bytes_sent "$http_referer" '
    '"$http_user_agent" "$http_x_forwarded_for"';
  access_log   logs/access.log  main;
  sendfile     on;
  tcp_nopush   on;
  server_names_hash_bucket_size 128; # this seems to be required for some vhosts

  server { # php/fastcgi
    listen       80;
    server_name  domain1.com www.domain1.com;
    access_log   logs/domain1.access.log  main;
    root         html;

    location ~ \.php$ {
      fastcgi_pass   127.0.0.1:1025;
    }
  }
```

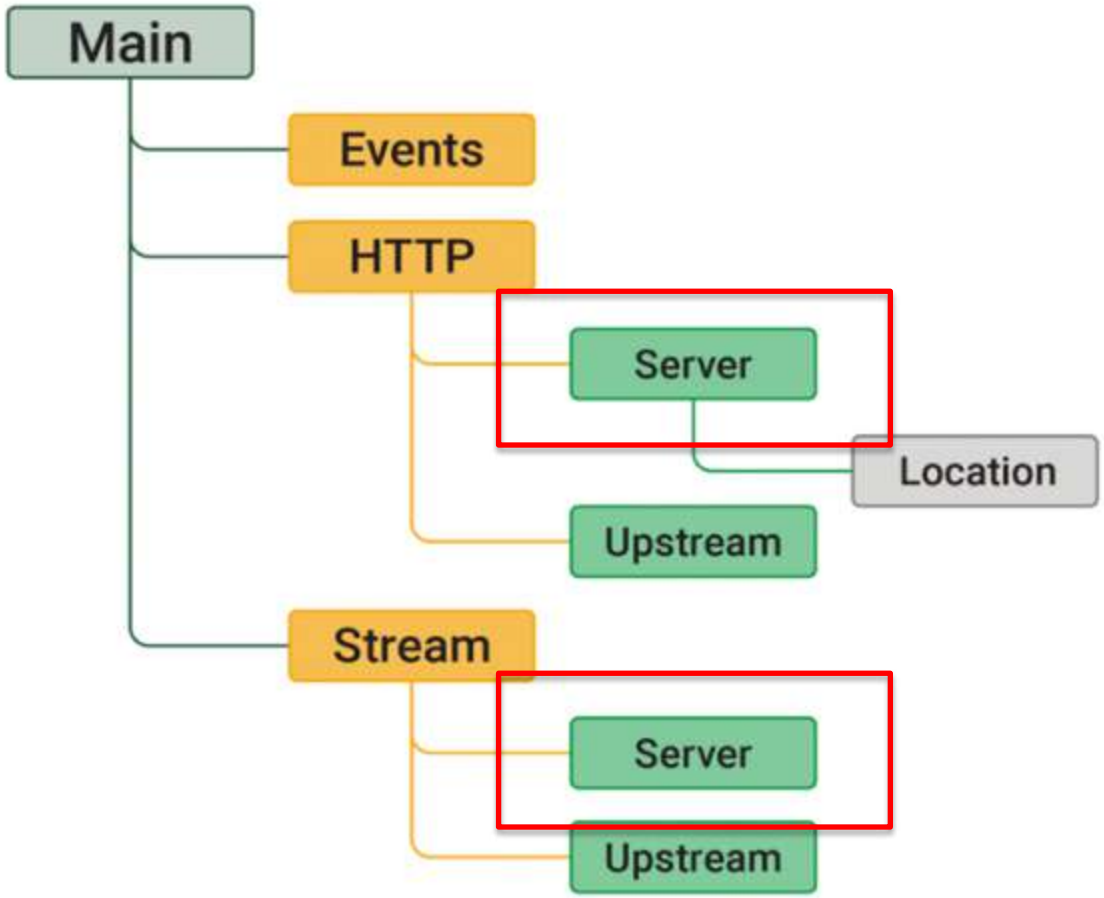
主配置文件nginx.conf



Main	上下文定义最高级别的指令，例如worker的数量，linux用户，PID文件的位置，Log文件的位置等等
Events	用于管理连接处理的指令，比如每worker进程的连接数
HTTP	定义Nginx如何处理HTTP/HTTPS连接，包含后台资源，如Pool 成员，在http 上下文中的其他指令会被其子级上下文继承，如upstream,server,location
server	定义虚拟服务器或虚拟主机，用来处理http请求，定义可以是一个FQDN域名，IP地址，或Unix Socket
location	进一步定义如何处理一个URI
upstream	定义后端服务器组
stream	定义如何处理TCP/UDP流量

*配置文件由若干配置块构成，配置块包含指令指令和模块相关，指令以";"为结束，配置块以{}标示，无需";"结束

Server配置块



NGINX接受到REQUEST后会首先判断使用在nginx.conf文件中的那个server进行处理，选择server通过server中的两个指令进行判断：

- listen
- server_name

listen指令

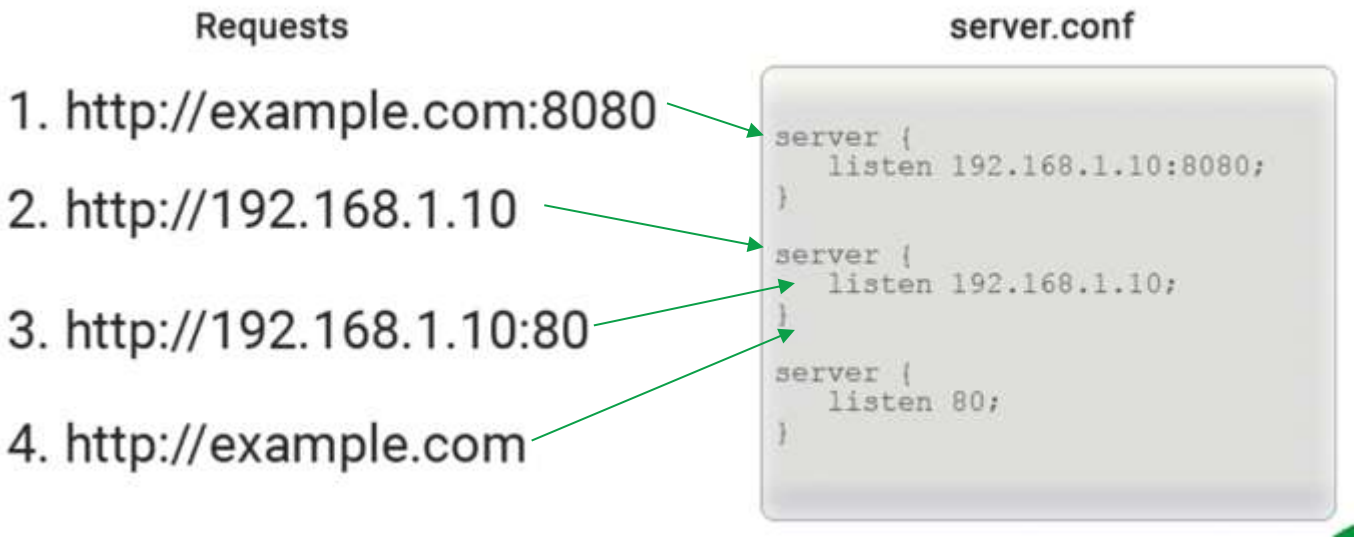
DIRECTIVE	NGINX USES
listen 80;	0.0.0.0:80
listen 127.0.0.1:8080;	127.0.0.1:8080
listen 127.0.0.2;	127.0.0.2:80
listen unix:/var/run/nginx.sock;	Must use absolute path
<no listen directive>	0.0.0.0:80

listen指令定义Nginx监听的IP和端口，如果IP不写就是0.0.0.0，为该服务器上的全部IP地址，如不写端口，此时默认为80,该指令可以配置协议和TCP行为，诸如：

- SSL
- HTTPv2/SPDY
- TCP keepalive

listen指令

假定DNS服务上配置example.com为192.168.1.10



在server 那么一致的情况下无冲突配置：

listen 80 and listen 192.168.1.1

有冲突配置，有一个会被忽略：

listen 192.168.1.1:80 and listen 192.168.1.1

Nginx在接受到请求后，针对不同的请求，会在文件的Server Block进行查找匹配listen IP，然后在同级listen 的server中根据server_name选择。

IP always win

defaut_server参数

```
server {
    listen 192.168.1.1;
    server_name www.example.*;
}

server {
    listen 192.168.1.1;
    server_name *.example.net;
}

server {
    listen 192.168.1.1;
    server_name ~^(www|host1).*\.example\.com$;
}

server {
    listen 192.168.1.1 default_server;
    error_page 404 /40x.html;
}
```

- Request流量进入监听端口，而nginx无法根据listen和server_name参数匹配到任何的server block的时候，会调用包含default_server指令的server block进行处理
- 如果没有default_server指令，而其他所有的server block无法精确匹配上，此时会调用候选第一个server block

server_name指令

同IP端口下的不同服务

```
server {
    listen 192.168.1.10;
    server_name www.example.org;
}

server {
    listen 192.168.1.10;
    server_name www.example.com;
}
```

各server的匹配顺序

1. `server_name` exact matches
2. `server_name` with leading wild card
3. `server_name` with trailing wild card
4. `server_name` regex matches

Server_name指令匹配报文头中的Host字段，默认为“”，通常是一个FQDN域名或IP地址,该指令一般用来做同IP端口下不同服务的区分，server_name可以采用通配符前缀，后缀的方式，也可以用使用正则表达式.

server_name中的正则表达式

Request: http://www.example.com

```
server {
    listen 80;
    server_name ~^(www|host1).*\.example\.com$;
}

server {
    listen 80;
    server_name ~^(subdomain|set|www|host1).*\.example\.com$;
}
```

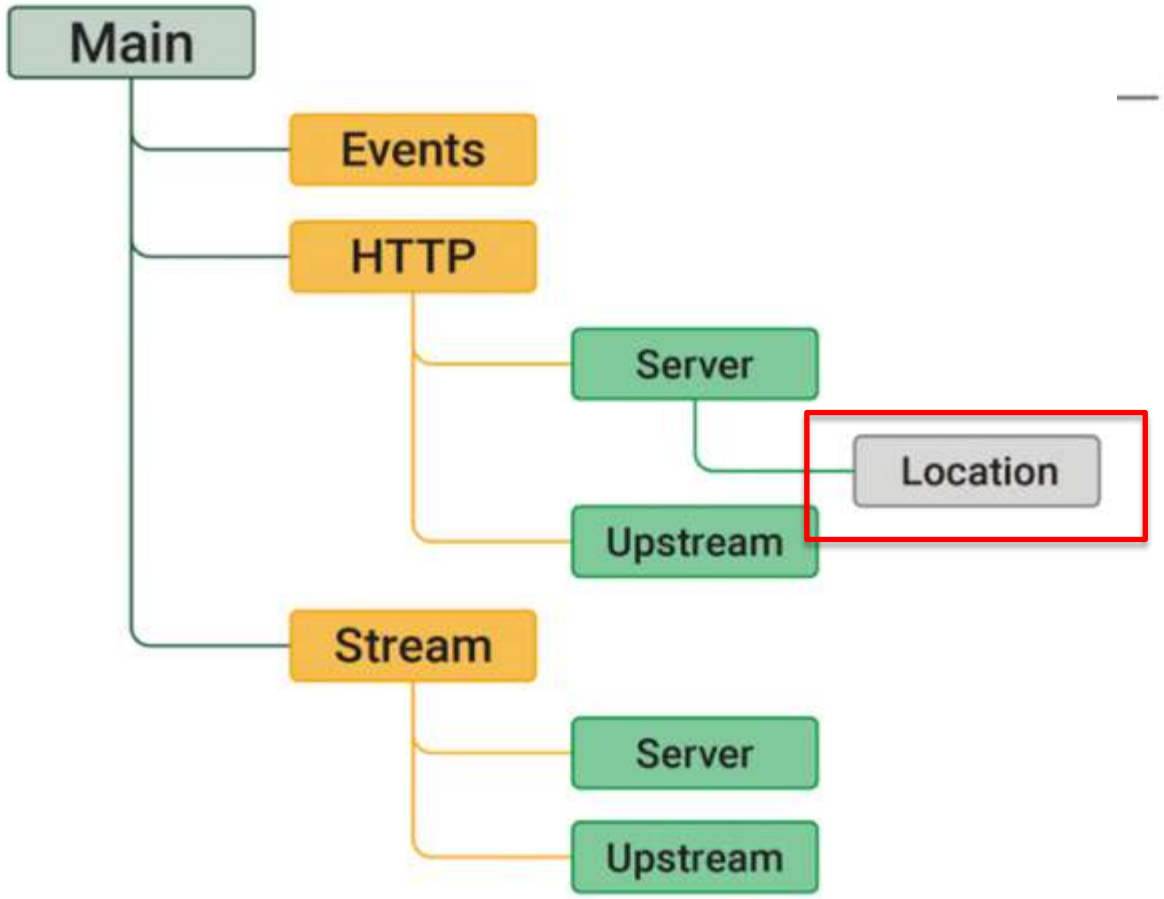
当没有精确匹配和通配符前后缀匹配的时候，Nginx会用perl的正则表达式匹配逻辑进行server block匹配，请注意按照正则表达式进行匹配的时候，没有所谓"best match"，而是按照在配置文件中的顺序进行匹配，如上例，两个server block都可以匹配到www.example.com的请求，第二个server block更精细，但是第一个server block会处理http://www.example.com

question

```
server {
       listen 80;
       server_name www.test.com;
       return 200 "this is listen 80 \n";
   }
server {
       listen 192.168.1.200;
       server_name www.a.com;
       return 200 "this is listen 10.129.4.209 ,www.a.com\n";

   }
```

```
curl http://192.168.1.200 -H "Host:www.test.com" -v
```

Location




```
Syntax: location [ = | ~ | ~* | ^~ ] uri { ... }
location @name { ... }
Context: server, location
```

- Location配置块定义具体的uri应该如何响应
- =,~,~*^~用于uri的匹配
- @用来创建一个内部location标识，可以用在类似try_file的命令里

常用Location修饰器

Module Name	Description
~*	不区分大小写正则表达式
~	区分大小写正则表达式
^~	不使用正则表达式
=	精确匹配

Prefix Location

- Nginx使用最长匹配原则匹配前缀Location
- 在前缀Location中，顺序是无关紧要的

Request： http://www.example.com/gallery/images/cat.png

```
location /gallery {
}

location /gallery/images {
}
```

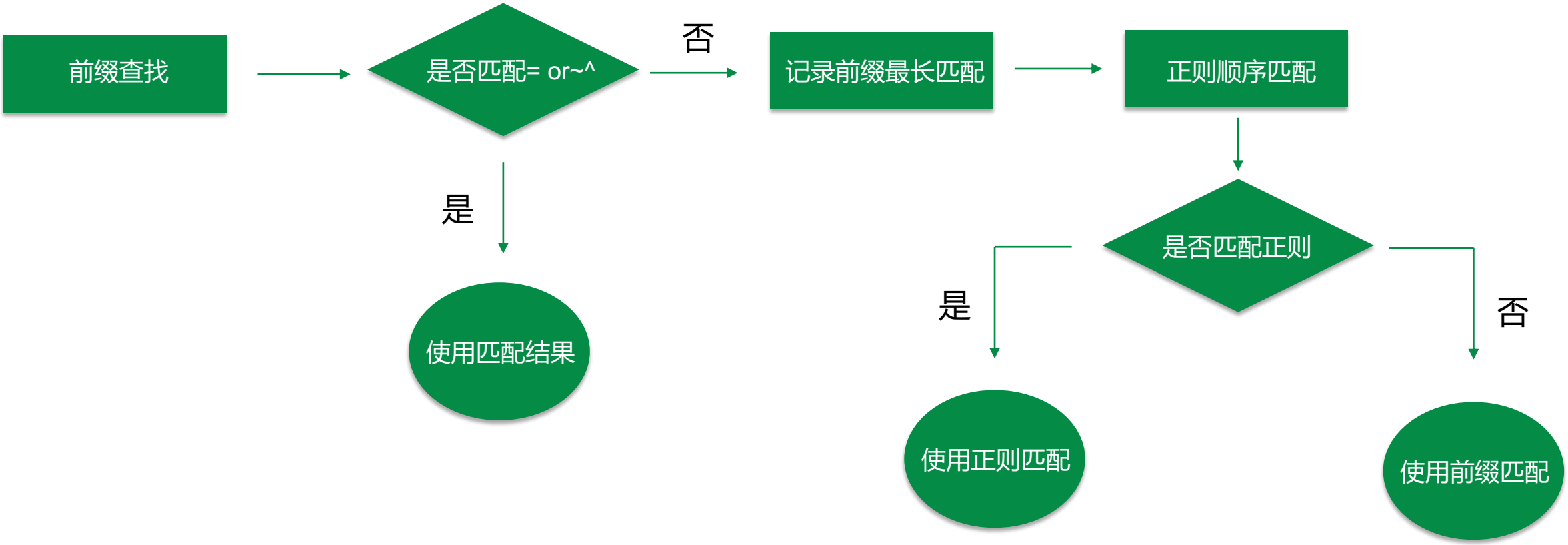
Regex Location

- 在正则表达式匹配中，使用那个location取决于在配置文件中的位置

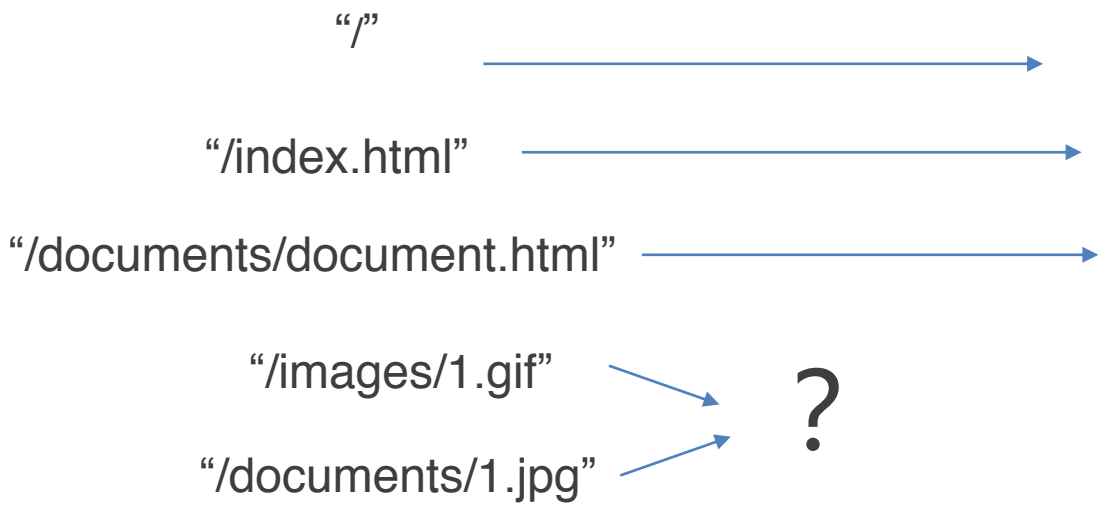
Request： http://www.example.com/gallery/images/cat.png

```
location /gallery {
}

location ~* ^/.(png|jpg)$ {
}

location ^~ /gallery/images {
}
```

Location查找逻辑



Location 范例




```
location = / { [ configuration A ] }

location / { [ configuration B ] }

location /documents/ { [ configuration C ] }

location ^~ /images/ { [ configuration D ] }

location ~* \.(gif|jpg|jpeg)$ { [ configuration E ] }
```

Location Tips

- 如"/"访问比较多，使用=可以加快查找速度

```
location = / { [ configuration A ] }
```

- 如location中，前缀匹配结束包含"/"，在后续配置包含 proxy_pass, fastcgi_pass, uwsgi_pass, scgi_pass, memcached_pass, or grpc_pass, request访问不带"/"结尾，则会返回301，可以考虑分开：

```
location /user/ {
 proxy_pass http://user.example.com; } l
ocation = /user {
proxy_pass http://login.example.com; }
```

进入server/location后

本地静态WEB服务器

```
server {
  listen 80;
  root /home/public_html;

  location /test {
    return http://other-server$uri;
  }
}
```

HTTP/TCP反向代理
和负载均衡

```
location / {
proxy_pass http://192.4.1.4/application1;
}
```

动态服务器和Cache

Directive	Passes to:
fastcgi_pass	FastCGI server
uwsgi_pass	uswgi server
scgi_pass	SCGI server
grpc_pass	gRPC server
memcached_pass	memcached server

Proxy module

Stream

proxy_bind
proxy_buffer_size
proxy_connect_timeout
proxy_download_rate
proxy_next_upstream
proxy_next_upstream_timeout
proxy_next_upstream_tries
proxy_pass
proxy_protocol
proxy_requests
proxy_responses
proxy_session_drop
proxy_socket_keepalive
proxy_ssl
proxy_ssl_certificate
proxy_ssl_certificate_key
proxy_ssl_ciphers
proxy_ssl_crl
proxy_ssl_name
proxy_ssl_password_file
proxy_ssl_protocols
proxy_ssl_server_name
proxy_ssl_session_reuse
proxy_ssl_trusted_certificate
proxy_ssl_verify
proxy_ssl_verify_depth
proxy_timeout
proxy_upload_rate

HTTP

proxy_bind
proxy_buffer_size
proxy_buffering
proxy_buffers
proxy_busy_buffers_size
proxy_cache
proxy_cache_background_update
proxy_cache_bypass
proxy_cache_convert_head
proxy_cache_key
proxy_cache_lock
proxy_cache_lock_age
proxy_cache_lock_timeout
proxy_cache_max_range_offset
proxy_cache_methods
proxy_cache_min_uses
proxy_cache_path
proxy_cache_purge
proxy_cache_revalidate
proxy_cache_use_stale
proxy_cache_valid
proxy_connect_timeout
proxy_cookie_domain
proxy_cookie_path
proxy_force_ranges
proxy_headers_hash_bucket_size
proxy_headers_hash_max_size
proxy_hide_header
proxy_http_version

反向代理和负载均衡行为通过下面两个模块实现：

- ngx_http_proxy_module
- ngx_stream_proxy_module

可以在server以及location上下文中对应的指令实现
如下列的功能：

- Proxy到哪里？
- 对request/responseHTTP报文头的操作
- 指定反向代理使用的源IP地址或保留客户端IP地址
- 客户端断开TCP连接，nginx是否继续等待服务器
- 代理行为中是否忽略某些报文头指定的行为

proxy_pass

该指令用来实现反向代理和负载均衡,其语法为

```
Syntax:proxy_pass URL;
Default:—
Context:location, if in location, limit_except (server)
```

对于HTTP类型需要写在location上下文，对tcp流量可以直接写在server block

```
location /name/ {
proxy_pass http://1.1.1.1/remote/;
}
```

```
server {
     listen 12345;
     proxy_connect_timeout 1s;
     proxy_timeout 3s;
     proxy_pass backend;
}
```

Proxy_pass后续可以用一个url（代理）或者upstream（负载均衡）

HTTP Proxy行为

```
GET /test HTTP/1.1
Host: 10.128.4.209
User-Agent: curl/7.54.0
Accept: */*
```

```
GET / HTTP/1.0
Host: 10.128.4.129:8080
Connection: close
User-Agent: curl/7.54.0
Accept: */*
```


```
HTTP/1.1 200 OK
Server: nginx/1.15.10
Date: Thu, 20 Feb 2020
06:47:29 GMT
Content-Type: text/plain
Transfer-Encoding: chunked
Connection: keep-alive
```

```
HTTP/1.1 200 OK
Content-Type: text/plain
Date: Thu, 20 Feb 2020
06:47:29 GMT
Connection: close
```

Nginx HTTP转发流量是proxy行为，不同于网络意义上的路由转发，防火墙的NAT，硬件负载均衡的LB，默认请求流量会被拆分到多连接，响应流量可能会被缓存。

在http, server, location上下文使用server_tokens off消除nginx版本信息，nginx plus中使用参数""可以彻底删除server报文头，也可以用add_header

proxy_http_version可以指定http 版本号，推荐在keepalive连接和NTLM authentication

后端响应中的"Date", "Server", "X-Pad", and "X-Accel-..."报文头默认不发送给客户端

TCP行为描述

配置	Proxy的TCP行为
Proxy_pass http://backend	客户端用一个TCP连接发起多request，Nginx将针每个request发起一个后端TCP连接，每个TCP连接发送一个request
Proxy_http_version 1.1 Proxy_pass http://backend	客户端用一个TCP连接发起多request，Nginx将针每个request发起一个后端TCP连接，每个TCP连接发送一个request
Proxy_http_version 1.1 Proxy_pass http://backend upstream keepalive	客户端多个个TCP发起多request，Nginx创建少量连接，复用已经打开的TCP连接，发送多个requst
Stream下配置proxy_pass	客户端每建立一个TCP连接，在三次握手完成后，nginx就和后端建立一个TCP连接

HTTP带路径proxy

Request：http://www.example.com/application1/user1/index.php

```
server {
   listen 80 default_server; server_name
www.example.com;
   root /usr/share/nginx/html;

   location /application1/ {
      proxy_pass http://192.168.1.2:8080/otherapp/
   }
}
```

Proxied Request：http://192.168.1.2:8080/otherapp/user1/index.php

如果proxy_pass指令中包含路径，那么request路径会被改写为proxy_pass的路径，行为和root命令保持一致

HTTP不带路径proxy

Client Request：http://www.example.com/application2/user2/index.php

```
server {
   listen 80 default_server;
   server_name www.example.com;
   root /usr/share/nginx/html;

   location /application2 {
      proxy_pass http://192.168.1.2:8080;
   }
}
```

Proxied Request：http://192.168.1.2:8080/application2/user2/index.php

如果proxy_pass指令不包含路径，那么实际request中的路径添加上location中的路径

HTTP proxy_set_header指令

```
Syntax:
proxy_set_header field value;
Default:proxy_set_header Host $proxy_host;proxy_set_header
Connection close;
Context: http, server, location
```

```
server {
   listen 80;
   proxy_set_header Host $host;
   proxy_set_header X-Real-IP $remote_addr;
   proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
   location / {
      proxy_pass http://backend-server.com;
}
```

在代理转发HTTP request中，部分客户端的真实信息会丢失，nginx会替换为自身的信息转发request，此时可以通过：

- proxy_set_header

将指定的信息插入请求中，然后代理发送到外部

HTTP add_header指令

```
Syntax:add_header name value [always];
Default:—
Context:http, server, location, if in location
```

```
server {
  listen 80;
  add_header <custom_header> <value>;
}
```

add_header执行对response中报文头的修改，可以用于添加nginx信息，debug，以及通过修改cache-control重新定义在客户端的缓存行为

只有当response code为200, 201 (1.3.10), 204, 206, 301, 302, 303, 304, 307 (1.1.16, 1.0.13), or 308 (1.13.0)才生效，除非设定为always

该指令可从上下文中继承，如本级配置没有配置，那么上层配置会被继承

客户端 TCP keep alive


```
http {
   keepalive_requests 150;
   keepalive_timeout 30s;
...
}
```

Keepalive指令指定客户侧到NGINX侧连接的最大请求数量和超时时间，默认为每个TCP连接100个request，超时时间为65sec

Course Summary

Task	Task	Task
确定版本	listen配置	Proxy_pass配置
软件仓库/源码安装	server_name配置	Proxy_pass行为描述
加载非默认模块	多server listen的选择	对请求和响应的修改

简单的例子

```
http {
  upstream myproject {
    server 127.0.0.1:8000 weight=3;
    server 127.0.0.1:8001;
    server 127.0.0.1:8002;
    server 127.0.0.1:8003;
  }

  server {
    listen 80;
    server_name www.domain.com;
    location / {
      proxy_pass http://myproject;
    }
  }
}
```

proxy.conf

```
proxy_redirect          off;
proxy_set_header        Host            $host;
proxy_set_header        X-Real-IP       $remote_addr;
proxy_set_header        X-Forwarded-For $proxy_add_x_forwarded_for;
client_max_body_size    10m;
client_body_buffer_size 128k;
proxy_connect_timeout   90;
proxy_send_timeout      90;
proxy_read_timeout      90;
proxy_buffers           32 4k;
```

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

Thank you