ingress 详解

nginx是什么

Nginx是俄罗斯人Igor Sysoev基于C语言编写的十分轻量级的HTTP服务器，它主要有以下特点：

- 它是一个高性能的HTTP和反向代理服务器，同时也是一个IMAP/POP3/SMTP 代理服务器；
- Nginx使用异步事件驱动的方法来处理请求，Nginx的模块化事件驱动架构可以在高负载下提供更可预测的性能；
- 作为Web服务器，Nginx处理静态文件、索引文件，自动索引的效率非常高
- 作为反向代理服务器，Nginx可以实现反向代理加速，提高网站运行速度
- 作为负载均衡服务器，Nginx既可以在内部直接支持Rails和PHP，也可以支持HTTP代理服务器对外进行服务，同时还支持简单的容错和利用算法进行负载均衡
- Nginx是专门为性能优化而开发的，非常注重效率，Nginx在官方测试的结果中，能够支持五万个并行连接，而在实际的运作中，可以支持二万至四万个并行链接
- 在高可用性方面，Nginx支持热部署，启动速度特别迅速，因此可以在不间断服务的情况下，对软件版本或者配置进行升级

nginx比较有用的关键命令

```
nginx -t 检查配置是否可用
nginx -T 遍历所有配置串接
nginx -s reload 重启nginx
```

nginx系统变量

```
$args                # 这个变量等于请求行中的参数，同$query_string
$content_length      # 请求头中的Content-length字段。
$content_type        # 请求头中的Content-Type字段。
$document_root       # 当前请求在root指令中指定的值。
$host                # 请求主机头字段，如果不存在则为服务器名称。
$http_user_agent     # 客户端agent信息
$http_cookie         # 客户端cookie信息
$limit_rate          # 这个变量可以限制连接速率。
$request_method      # 客户端请求的动作，通常为GET或POST。
$remote_addr         # 客户端的IP地址。
$remote_port         # 客户端的端口。
$remote_user         # 已经经过Auth Basic Module验证的用户名。
$request_filename    # 当前请求的文件路径，由root或alias指令与URI请求生成。
$scheme              # HTTP方法（如http，https）。
$server_protocol     # 请求使用的协议，通常是HTTP/1.0或HTTP/1.1。
$server_addr         # 服务器地址，在完成一次系统调用后可以确定这个值。
$server_name         # 服务器名称。
$server_port         # 请求到达服务器的端口号。
$request_uri         # 包含请求参数的原始URI，不包含主机名，如：”/foo/bar.php?
arg=baz”。
```

```
$uri                # 不带请求参数的当前URI，$uri不包含主机名，如”/foo/bar.html”。
$document_uri       # 与$uri相同。
```

nginx变量语法

```
# 设置变量$a = "helloworld";
set $a hello world;

# 设置变量$b = "helloworld, helloworld";
set $b "$a, $a";
```

nginx map语法 (支持正则语法)

```
map $http_user_agent $status {
    default 0; #$http_user_agent默认值 $status为0
    ~curl  -1; #$http_user_agent匹配到curl $status设为-1
    ~*chrome 1; #$http_user_agent匹配到curl $status设为1
}

http {
    server {
        listen 80;
        server_name ops.com;
        location /test {
            default_type text/plain;
            echo http_user_agent: $http_user_agent;
            echo status: $status;
        }
    }
}
```

nginx主体配置

- main（全局设置）：主要是包括Nginx工作进程，日志的配置以及server，location中一些共用的配置
- events（连接设置）：主要包括Nginx连接信息的配置
- server（主机设置）：主要是包括主机名称，Ip，路径解析，http请求头设置，反向代理等配置
- upstream（上游服务器设置）：主要为反向代理服务器信息、负载均衡等相关配置
- location（URL匹配）：特定URL的匹配设置

> 以上每部分包含若干个条指令，他们之间的关系是：server继承main，location继承server，main部分设置的指令将影响其它所有部分的设置，server部分的设置将影响到location部分的设置,upstream既不会继承指令也不会被继承，它有自己的特殊指令，不需要在其他地方的应用。

```
############################################################################
```

```
#######################################
# main全局配置
#
##########################################################################
#######################################

user  www www;  #默认为nobody
# 设置nginx工作进程的用户

worker_processes  2; # 默认为1
# 设置worker角色的工作进程的个数，正常情况下可以设置成cpu的内核数，最多设置为8个；
# 也可以将worker_processes的值设为auto，这样nginx会自动检测CPU核数并打开相同数量的
worker进程；
# 当nginx添加了SSL证书时，最好要打开多个worker进程。SSL握手会进行硬盘I/O操作。所以打
开多个worker进程有利于性能的提升；

worker_cpu_affinity 01 10;
# 通过设置cpu粘性来降低由于多CPU核切换造成的寄存器等现场重建带来的性能损耗，上述设置表示
第一个worker进程用第一个cpu，第二个worker进程进程使用第二个cpu

worker_rlimit_nofile; # 默认为操作系统的限制（65535）
# 设置每个worker进程的最大打开文件数（句柄）限制

error_log  logs/error.log error;
# 配置错误日志路径以及打印级别（debug | info | notice | warn | error | crit |
alert | emerg）
# 生产场景一般是warn | error | crit 这三个级别之一，级别太低会有太多IO消耗，影响效率

pid logs/nginx.pid;
# pid文件为文本文件，内容只有一行, 记录了该进程的ID，根据PID文件的内容，准确判断进程是
否正在运行，防止意外启动多个进程实例。
# 只有获得pid文件(固定路径固定文件名)写入权限(F_WRLCK)的进程才能正常启动并把自身的PID
写入该文件中，其它同一个程序的多余进程则自动退出。

##########################################################################
#######################################
# events模块中包含nginx中所有处理连接的设置
#
##########################################################################
#######################################

events {
    use epoll;
    # 用于设置处理客户端请求的轮询方法
    # 在Linux操作系统下，nginx默认使用epoll事件模型
    # 同时Nginx在OpenBSD或FreeBSD操作系统上采用类似于epoll的高效事件模型kqueue
    # 在操作系统不支持这些高效模型时才使用select

    worker_connections  2048;  #默认为512
    # 设置可由一个worker进程同时打开的最大连接数。但不能超过worker_rlimit_nofile的
设置

    accept_mutex on;  # 默认为on
    # 当一个新连接到达时，如果激活了accept_mutex，那么多个Worker将以串行方式来处理，
```

```
其中有一个Worker会被唤醒，其他的Worker继续保持休眠状态；
    # 如果没有激活accept_mutex，那么所有的Worker都会被唤醒，不过只有一个Worker能获取新连接，其它的Worker会重新进入休眠状态，[thundering herd现象](https://en.wikipedia.org/wiki/Thundering_herd_problem)

    accept_mutex_delay 500ms; # 默认为500ms
    # 当accept_mutex功能启用后，只有一个持有mutex锁的worker进程会接受并处理请求，其他worker进程等待。accept_mutex_delay指定的时间就是这些worker进程的等待时间，过了等待时间下一个worker进程便取得mutex锁，处理请求。

    multi_accept on # 默认为off
    # multi_accept可以让nginx worker进程尽可能多地接受请求，提高性能
    # 如果设置为on，可以让worker进程一次性地接受监听队列里的所有请求，然后处理
    # 如果multi_accept的值设为off，那么worker进程必须一个一个地接受监听队列里的请求
    # 如果web服务器面对的是一个持续的请求流，那么启用multi_accept可能会造成worker进程一次接受的请求大于worker_connections指定可以接受的请求数。这就是overflow，这个overflow会造成性能损失，overflow这部分的请求不会受到处理
}

#########################################################################################################################
# 提供http服务相关的一些配置参数
#
#########################################################################################################################

http {


    ##########################################################################################
    # 基本配置
    #
    ##########################################################################################

    include  mime.types;
    # include可以包含若干子配置文件，实现不同需求配置独立，可以将不同的server配置在不同的conf文件里
    # mime.types文件列出针对不同的请求文件返回的HTTP response的Content-Type的Accept值
    # 除非服务端Web程序手动设置了Content-Type，如果Web程序没设置，则会从mime.types中匹配返回
    # 如果mime.types中也没找到对应文件的扩展名的话，就使用默认的default_type

    default_type  application/octet-stream;
    # 如果在mime.types的配置中没有找到响应请求文件的格式，则走default_type

    types_hash_max_size 2048;
    # 设置散列表的冲突率。
    # types_hash_max_size越大，就会消耗更多的内存，但散列key的冲突率会降低，检索速度就更快。
    # types_hash_max_size越小，消耗的内存就越小，但散列key的冲突率可能上升。
    server_tokens off;
    # 返回错误页面时是否在Server中注明Nginx版本

    server_names_hash_bucket_size 64;
    # 为了提高快速寻找到相应server_name的能力,Nginx 使用散列表来存储
server_name,server_names_hash_bucket_size设置了每个散列桶占用的内存大小。

    server_name_in_redirect off;
    # 重定向主机名称的处理.该配置需要配合server_name使用.
    # 设置为on时,表示在重定向请求时会使用stream里配置的第一个主机名代替原先请求中的
Host头部,而当关闭时,表示在重定向请求时使用请求本身的Host头部。




##########################################################################
#########
    # 日志配置
#

##########################################################################
#########

    log_format  main  '$remote_addr - $remote_user [$time_local] "$request"
'
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';
    # 定义日志nginx日志文件的打印格式并命名为变量main

    access_log  logs/access.log  main;
    # access_log指定nginx的访问日志的存放路径文件以及使用的日志文件格式

    #access_log off;
    # 为了提高效率，可以将访问日志关掉

    access_log /data/logs/nginx-access.log buffer=32k flush=5s;
    # buffer和flush可以设置缓存日志的刷盘策略，日志超过32k才刷盘，如果没满32k，但是超
过5s也自动刷盘

    rewrite_log on; # 默认是off
    # 开启或者关闭rewrite模块指令执行的日志，如果开启，则重写将记录下notice等级的日志
到nginx 的error_log中


##########################################################################
#########
    # 高效文件传输
#

##########################################################################
#########

    sendfile on;
    # 当一个程序需要传输文件时，Linux内核首先将文件数据缓冲，然后将文件数据传送给程序缓冲，最后程序将文件数据传输到目的地。
    # Sendfile方法是一种数据传输的更高效的方法，数据在内核中的文件描述符之间传输
    # 这种方法的结果是改善了对操作系统资源的利用，提高Nginx静态资源托管效率，直接在内核空间完成文件发送，不需要先read再write，没有上下文切换开销

    tcp_nopush on;
    # TCP_NOPUSH是FreeBSD的一个socket选项，对应Linux的TCP_CORK，Nginx里统一用tcp_nopush来控制它，并且只有在启用了sendfile之后才生效。
    # 启用它之后，数据包会累计到一定大小之后才会发送，减小了额外开销，提高网络效率

    tcp_nodelay on;
    # TCP_NODELAY也是一个socket选项，启用后会禁用Nagle算法，尽快发送数据，某些情况下可以节约200ms
    # Nagle算法原理是：在发出去的数据还未被确认之前，新生成的小数据先存起来，凑满一个MSS 或者等到收到确认后再发送
    # Nginx 只会针对处于keep-alive状态的TCP连接才会启用tcp_nodelay

    keepalive_timeout  65;

    # 一个keepalive连接在闲置超过一定时间后（默认的是75秒），会主动关闭这个长连接
    # 客户端可以设置http服务要不要走长连接，通过设置请求头Connection=keep-alive实现的，http1.0默认是关闭的，http1.1默认是打开的
    # 谷歌浏览器同时最多有6个tcp连接
    # keepalive_timeout时间不能设置太长，因为太长会长时间占用tcp连接不释放，导致服务器的tcp连接不够用；也不能太短，如果太短会导致一些大文件上传接口因为上传一半而中断；

    keepalive_requests 200;
    # 设置同一个长连接上最多请求次数，超过这个次数，将主动关闭这个长连接


#########################################################################################
    # http_proxy 设置，client相关配置
#

#########################################################################################

    client_max_body_size   10m;
    # 允许客户端请求的最大单文件字节数限制。如果有上传较大文件的需求，尽量设置大一些

    client_body_buffer_size   128k;
    # 缓冲区代理用户端请求的最大字节数

    client_header_timeout 60;
    # 指定等待client发送一个请求头的超时时间，仅当在一次read中，没有收到请求头，才会算成超时。如果在超时时间内，client没发送任何东西，nginx返回HTTP状态码408(“Request timed out”)

    client_body_timeout 60;
    # 该指令设置请求体（request body）的读超时时间。仅当在一次readstep中，没有得到请求体，就会设为超时。超时后，nginx返回HTTP状态码408(“Request timed out”)
```

```
##########################################################################
#########
    # http_proxy 设置，server相关配置
#

##########################################################################
#########

    proxy_connect_timeout  60;
    # 该指令设置与upstream server的连接超时时间，有必要记住，这个超时不能超过75秒

    proxy_send_timeout   75;
    # 这个指定设置了发送请求给upstream服务器的超时时间。超时设置不是为了整个发送期间，
而是在两次write操作期间。如果超时后，upstream没有收到新的数据，nginx会关闭连接

    proxy_read_timeout   75;
    # 该指令设置与代理服务器的读超时时间。它决定了nginx会等待多长时间来获得请求的响应。
这个时间不是获得整个response的时间，而是两次reading操作的时间,默认值60s

    proxy_upstream_fail_timeout 10;
    # Upstream模块下server指令的参数，设置了某一个upstream后端失败了指定次数
（max_fails）后，该后端不可操作的时间，默认为10秒

    proxy_buffer_size   4k;
    # 设置代理服务器（nginx）从后端realserver读取并保存用户头信息的缓冲区大小，默认与
proxy_buffers大小相同，其实可以将这个指令值设的小一点

    proxy_buffers   4 32k;
    # proxy_buffers缓冲区，4个缓存，每个大小限制为32k。

    proxy_busy_buffers_size   64k;
    # 高负荷下缓冲大小（proxy_buffers*2）

    proxy_temp_file_write_size  64k; # 默认为1024M
    # 当proxy_buffers放不下后端服务器的响应内容时，会将一部分保存到硬盘的临时文件中，
这个值用来设置最大临时文件大小，默认1024M
    # 它与proxy_cache没有关系，大于这个值，将从upstream服务器传回。设置为0禁用

    proxy_temp_path   /usr/local/nginx/proxy_temp 1 2;
    # 指定缓存写到那个目录


##########################################################################
#########
    # gzip压缩功能设置
#

##########################################################################
#########

    gzip on;
    # 开启gzip压缩输出，减少网络传输,客户端通过设置请求头Accept-Encoding=gzip,
deflate, br来支持gzip压缩
```

```
    gzip_static on;
    # nginx对于静态文件的处理模块，该模块可以读取预先压缩的gz文件，这样可以减少每次请求
进行gzip压缩的CPU资源消耗。
    # 该模块启用后，nginx首先检查是否存在请求静态文件的gz结尾的文件，如果有则直接返回该
gz文件内容

    gzip_disable "msie[1-6].";
    # IE6的某些版本对gzip的压缩支持很不好，会造成页面的假死，为了确保其它的IE6版本不出
问题，所以建议加上gzip_disable的设置

    gzip_min_length 1k;
    # 设置允许压缩的页面最小字节数，页面字节数从header头得content-length中进行获取。
默认值是20。建议设置成大于1k的字节数，小于1k可能会越压越大

    gzip_buffers    4 16k;
    # 设置系统获取几个单位的缓存用于存储gzip的压缩结果数据流。4 16k代表以16k为单位，安
装原始数据大小以16k为单位的4倍申请内存

    gzip_http_version 1.0;
    # 用于识别http协议的版本，早期的浏览器不支持Gzip压缩，用户就会看到乱码，所以为了支
持前期版本加上了这个选项。
    # 如果你用了Nginx的反向代理并期望也启用Gzip压缩的话，由于末端通信是 http/1.0，故
请设置为 1.0

    gzip_comp_level 6;
    # gzip压缩比，1压缩比最小处理速度最快，9压缩比最大但处理速度最慢(传输快但比较消耗
cpu)

    gzip_types text/html text/plain text/css text/javascript
application/json application/javascript application/x-javascript
    application/xml;
    # 匹配mime类型进行压缩，无论是否指定,"text/html"类型总是会被压缩的

    gzip_vary on;
    # 和http头有关系，会在响应头加个Vary:Accept-Encoding，可以让前端的缓存服务器缓存
经过gzip压缩的页面，例如，用Squid缓存经过Nginx压缩的数据

    gzip_proxied any
    # Nginx作为反向代理的时候启用，决定开启或者关闭后端服务器返回的结果是否压缩，匹配的
前提是后端服务器必须要返回包含"Via"的 header头


############################################################################
#######
    # FastCGI 设置，为了保证Nginx下PHP环境的高速稳定运行，需要添加一些FastCGI优化指
令          #

############################################################################
#######

    fastcgi_cache_path /usr/local/nginx/fastcgi_cache levels=1:2
keys_zone=TEST:10m inactive=5m;
    # 为FastCGI缓存指定一个文件路径、目录结构等级、关键字区域存储时间和非活动删除时间
    fastcgi_connect_timeout 300;
    # 指定连接到后端FastCGI的超时时间

    fastcgi_send_timeout 300;
    # 指定向FastCGI传送请求的超时时间，这个值是已经完成两次握手后向FastCGI传送请求的超时时间

    fastcgi_read_timeout 300;
    # 指定接收FastCGI应答的超时时间，这个值是已经完成两次握手后接收FastCGI应答的超时时间

    fastcgi_buffer_size 64k;
    # 用于指定读取FastCGI应答第一部分需要多大的缓冲区，这个值表示将使用1个64KB的缓冲区读取应答的第一部分（应答头），可以设置为fastcgi_buffers选项指定的缓冲区大小

    fastcgi_buffers 4 64k;
    # 指定本地需要用多少和多大的缓冲区来缓冲FastCGI的应答请求。
    # 如果一个PHP脚本所产生的页面大小为256KB，那么会为其分配4个64KB的缓冲区来缓存；如果页面大小大于256KB，那么大于256KB的部分会缓存到fastcgi_temp指定的路径中。
    # 一般这个值应该为站点中PHP脚本所产生的页面大小的中间值，如果站点大部分脚本所产生的页面大小为256KB，那么可以把这个值设置为“16 16k”、“4 64k”等

    fastcgi_temp_file_write_size 128k;
    # 表示在写入缓存文件时使用多大的数据块，默认值是fastcgi_buffers的两倍

    fastcgi_cache TEST;
    # 表示开启FastCGI缓存并为其指定一个名称。开启缓存非常有用，可以有效降低CPU的负载，并且防止502错误的发生

    fastcgi_cache_valid 200 302 1h;
    # 指定code为200，302的响应缓存为一小时

    fastcgi_cache_valid 301 1d;
    # 指定code为301的缓存有效时间为1天

    fastcgi_cache_valid any 1m;
    # 其它缓存有效时间都为1分钟


#############################################################################################
    # 设定负载均衡后台服务器列表
#

#############################################################################################

    upstream  backend  {
        keepalive 30
        # 在开启长连接的情况下，最多保持空闲长连接的个数，如果超过这个数量，最近最少使用的长连接将被关闭

        ip_hash;    # 默认为round-robin
        # 负载均衡处理方式，一共有三种方式：
        # (1)round-robin（轮训请求方式）
        # (2)ip_hash（回话持久化方式，这个方法保证从同一个客户端发起的请求总是定向到同一台服务器）
        # (3)least_conn（最少连接方式，找连接最少的服务进行处理）

        server   192.168.10.100:8080 max_fails=2 fail_timeout=30s weight=2;
        server   192.168.10.101:8080 max_fails=2 fail_timeout=30s weight=3;
        server   192.168.10.101:8080 backup
        server   192.168.10.101:8080 down;
        # weight设置每个服务的命中几率，默认是1；
        # backup表示备份服务，只有所有的非备份不能使用时，会启动该服务，down表示当前服务永远不参与负载；
        # max_fails表示容许请求失败的次数，当超过该次数时将暂停一定时间（fail_timeout）
    }


#######################################################################################
    # server虚拟主机配置
#

#######################################################################################

    server {
        ###########################################
        # 基本配置                                #
        ###########################################

        listen  80 default_server; # 默认为80
        # 监听端口设置，小于1024的要以root启动，
        # default_server表示如果找不到对应端口的server_name,则默认走这个匹配

        server_name  itoatest.example.com;
        # 一个nginx可以配置多个server，nginx通过检查请求header中host来匹配每个server的server_name决定走哪个server，
        # 如果没有任何一个server可以匹配，则会选择第一个server做匹配。默认匹配可以通过listen中添加defalut_server来改变。
        # server_name有四种匹配方式：
        # (1)精确匹配（itoatest.example.com）
        # (2)星号开头的最长的通配符名称（*.example.org）
        # (3)星号结束的最长的通配符名称（mail.*）
        # (4)正则表达式匹配（~^www\d+\.example\.net$，正则表达式必须以~开头）

        root   /apps/oaapp;
        # 见下文location讲解

        allow 223.252.218.196;
        allow 59.111.28.48/32;
        # allow表示允许某个ip或ip段访问

        deny all
        # deny表示禁止某个ip或者ip段访问

        error_page   500 502 503 504  /50x.html;
        error_page   403    http://example.com/forbidden.html;
        # 这个参数可以为错误代码指定相应的错误页面

        charset utf-8;
        # 设置http头信息的charset编码

        if ($request_method = POST) {
            return 405;
        }
        # 关于if的使用请看下文[Nginx中如何使用变量？]

        ############################################
        # location特定的URL对应的一系列配置项          #
        ############################################
        location /i/ {        # 关于location中的路径匹配规则以及匹配优先级请看下文[Nginx中location部分URL如何匹配?]
            root   /apps/oaapp;
            #alias  /apps/oaapp/;
            # root和alias都可以用来指定请求资源的真实路径。
            # 区别是root最后得到的资源地址是root的值加上location的值，而alias正如其名，alias指定的路径是location的别名，不管location的值怎么写，资源的真实路径都是 alias 指定的路径。
            # 比如当访问http://itoatest.example.com/i/hello.gif这个地址时，如果是root，资源的真实路径是/apps/oaapp/i/hello.gif;如果是alias真实路径是/apps/oaapp/hello.gif;
            # alias只能作用在location中，而root可以存在server、http和location中
            # alias 后面必须要用 “/” 结束，否则会找不到文件，而 root 则对 ”/” 可有可无

            index  index.jsp index.html index.htm;
            # 当用户请求的是http://itoatest.example.com/i/这个地址时，就会自动在root配置指令指定的文件系统目录下依次寻找 index.jsp 和 index.html，index.htm这三个文件，直到找到一个并返回

            autoindex on; # 默认为off
            # 当index指定的文件都找不到时，如果开启autoindex，那么则会生成一个root所指目录下的“目录索引”的html并返回，如果没有开启，则会返回forbidden

            autoindex_exact_size off # 默认为on
            # 只有在autoindex开启时才起作用，默认为on，显示出文件的确切大小，单位是bytes。改为off后，显示出文件的大概大小，单位是kB或者MB或者GB

            autoindex_localtime on # 默认为off
            # 只有在autoindex开启时才起作用，默认为off，显示的文件时间为GMT时间。改为on后，显示的文件时间为文件的服务器时间

            proxy_pass  http://backend;
            # 请求转向某个upstream定义的负载均衡的服务器列表，如果只有一台服务器的话，也可以直接写成proxy_pass http://ip:port；

            rewrite ^/i/(.*) /$1 break;
            # rewrite 的作用是修改 $uri，具体细节请看下文[rewrite如何重写url?]

            proxy_redirect off; # 默认是default
            proxy_redirect http://192.168.10.101:8080/i/wuman/
http://itoatest.example.com/i/wuman/
            # 如果需要修改从被代理服务器传来的应答头中的"Location"和"Refresh"字段，
可以用这个指令设置,分为三种情况：
            #（1）proxy_redirect off表示不修改服务端的redirect地址
            #（2）proxy_redirect default 将根据location和proxy_pass的设置来决定
            #（3）可以自己设置不同的替换规则

            proxy_set_header  Host  $host; #默认是$proxy_host
            # 可以通过三个变量对Host进行设置：
            #（1）$proxy_host，表示是反向代理后的host，就是proxy_pass后面跟的host
            # (2) $host首先从请求头中获取Host值，如果没有则选择server_name
            # (3) $http_host是直接从请求头中获取，所以可能为空，如果是非80/443端口
的时候，$http_host = $host:$port

            proxy_set_header  X-Real-IP  $remote_addr;
            # 由于在客户端和web服务器之间增加了中间层，因此web服务器无法直接拿到客户端
的ip，通过$remote_addr变量拿到的将是反向代理服务器的ip地址；
            # 所以我们可以设置一个请求头X-Real-IP，通过获取这个请求头就可以拿到客户端
的真实ip

            proxy_set_header  X-Forwarded-For  $proxy_add_x_forwarded_for;
            # 上述的意思增加一个$proxy_add_x_forwarded_for到X-Forwarded-For里
去，注意是增加，而不是覆盖
            # 如果每次代理都使用上述配置，那么X-Forwarded-For可以获取到经过多次代理
后的客户多IP以及多层代理nginx的IP：IP0, IP1, IP2...
            # 所以proxy_set_header也是获取真实客户端ip的一种方法

            proxy_set_header X-Forwarded-Proto  https;
            # 请求标头可帮助您识别客户端与您的负载均衡器连接时所用的协议,并随后将标题传
递到您的服务器

            proxy_set_header X-Forwarded-Host $host
            # 可以帮助您识别客户端与您的负载均衡器连接时所用的host,并随后将标题传递到
您的服务器

            proxy_set_header X-Forwarded-Host $port
            # 可以帮助您识别客户端与您的负载均衡器连接时所用的port,并随后将标题传递到
您的服务器

            proxy_next_upstream error timeout invalid_header http_500
http_502 http_503 http_504; # 默认是error timeout
            # 指定在什么情况下将请求应传递到下一个服务器，如果设置为off表示在任何情况
下都不需要传递
            # error表示发生错误时，将请求传递到下一个服务器
            # timeout表示发生请求或者响应超时时，将请求传递给下一个服务器
            # invalid_header表示服务器返回空响应或无效响应时，将请求传递给下一个服务
器
            # http_code表示服务器返回对应的code时将请求传递到下一个服务器

            proxy_next_upstream_timeout 30 # 默认是0
            # 限制请求可以传递到下一个服务器的时间，0表示关闭限制

            proxy_next_upstream_tries 2 # 默认为0
            # 限制将请求传递到下一个服务器的可能尝试次数， 0值关闭此限制

        }

        location ~ .*\.(gif|jpg|jpeg|bmp|png|ico|txt|js|css)$
        {
            root /apps/oaapp;

            expires      7d;
            # 对于站点中不经常修改的静态内容（如图片，JS，CSS），可以在服务器中设置
expires过期时间，控制浏览器缓存，达到有效减小带宽流量，降低服务器压力的目的
        }

        location = /50x.html {
            root   html;
        }

        location = /video {
            directio 4m;  # 该路径下所有大于4M的文件不直接从磁盘读取，不走缓存
            # Direct I/O是文件系统的一个功能，它允许程序绕过内核缓存，直接在硬盘上读
写数据
            # 这可以充分利用CPU频数，改善缓存的有效性,Directo I/O适用于访问率少的数
据。这些数据不需要缓存在任何位置
            # 在http, server和location当中都可以定义
            # 如果某个请求是通过directo I/O,那么这个请求不能使用sendfile功能
        }
    }
}
```

nginx中location部分url如何匹配？

location主要是匹配url中除去server_name（主机名）后的部分,其中关于url的匹配规则有以下几种：

- 精确匹配：以“=”开头表示精确匹配
- 开头匹配：^~ 表示uri以某个常规字符串开头，不是正则匹配
- 区分大小写的正则匹配：~开头表示区分大小写的正则匹配
- 不区分大小写的正则匹配：~* 开头表示不区分大小写的正则匹配
- 通用匹配：匹配url的前面部分

对于上述五类匹配，它们之间的匹配顺序和优先级关系如下：

- 不同类型之间匹配和location的顺序无关，只和优先级有关，各种匹配规则的优先级关系是： [精确匹配] > [开头匹配] > [正则匹配] > [通用匹配];
- 除了通用匹配,开头匹配以外，相同类型的匹配优先级只和顺序有关，排在前面的优先匹配；
- 通用匹配和开头匹配的优先级与通用匹配的最长字符串有关，通用字符串越长，匹配优先级越高；

下面是我设置的几个location，并测试和验证以上匹配规则：

```
server {
    listen       80 default_server;
    server_name  dev.zdp.com;

    # 通用匹配  [匹配规则0]
    location  / {
        return 302 https://dashboard.youdata.com;
    }

    # 通用匹配  [匹配规则1]
    location  /hello {
        return 302 https://dashboard.youdata.com;
    }

    # 通用匹配  [匹配规则2]
    location  /hello/no {
        return 302 https://dev.youdata.com;
    }

    # 不区分大小写的正则匹配  [匹配规则3]
    location ~* /hello/y[a-e][a-z][1-9] {
        return 302 https://test.youdata.com;
    }

    # 区分大小写的正则匹配  [匹配规则4]
    location ~ /hello/y[A-E][E-Z][1-9] {
        return 302 https://pre.youdata.com;
    }

    # 区分大小写的正则匹配  [匹配规则5]
    location ~ /hello/y[a-e][e-z] {
        return 302 https://pre163.youdata.com;
    }

    # 开头匹配  [匹配规则6]
    location ^~ /hello/yes {
        return 302  https://youdata.netease.com;
    }

    # 开头匹配  [匹配规则7]
    location ^~ /hello/yesno {
        return 302  https://youdata.163.com;
    }

    # 精确匹配 [匹配规则8]
    location = /hello {
        return 302  https://www.baidu.com;
    }
}
```

```
location用例测试：
- “http://dev.zdp.com/hello” —-- 精确匹配优先，命中[匹配规则8]
- “http://dev.zdp.com/hello/yesnoOk” —-- 开头匹配优先，开头匹配同时满足条件时，长优先，命中[匹配规则7]
- “http://dev.zdp.com/hello/yesOk” —-- 开头匹配优先，命中[匹配规则6]
- “http://dev.zdp.com/hello/yaz” —-- 正则匹配，命中[匹配规则5]
- “http://dev.zdp.com/hello/yAZ3” —-- 正则匹配，按照location顺序匹配，命中[匹配规则3]
- “http://dev.zdp.com/hello/no” —-- 通用匹配，按照匹配长度优先，命中[匹配规则2]
- “http://dev.zdp.com/hello/Ok” —-- 通用匹配，命中[匹配规则1]
- “http://dev.zdp.com/everyone” —-- 通用匹配，所有其它匹配不满足时，命中[匹配规则0]
```

nginx中rewrite命令如何重写url？

rewrite功能就是，使用nginx提供的全局变量或自己设置的变量，结合正则表达式和标志位实现url重写以及重定向。rewrite只能放在server{},location{}中，并且只能对域名后边的除去传递的参数外的字符串起作用，例如：

```
http://dev.zdp.com/a/we/index.php?id=1&u=str => rewrite只能对/a/we/index.php部分重写
重写用法：
server {
    rewrite 规则 定向路径 重写flag;
}

location {
    rewrite 规则 定向路径 重写flag;
}

rewrite的执行顺序
执行server块的rewrite指令；
执行location匹配；
执行选定的location中的rewrite指令，如果location中rewrite指令没有break的flag，则会根据当前rewrite路径重新匹配location；
如果其中某步URI被重写，则重新循环执行1-3，直到找到真实存在的文件，循环最多不会超过10次；
rewrite的flag标志
last: 停止处理当前location中的ngx_http_rewrite_module指令集（rewrite，return等），并开始重新搜索与更改后的URI相匹配的location
break : 停止处理当前location中的ngx_http_rewrite_module指令集（rewrite，return等），不会重新搜索
redirect : 返回302临时重定向，地址栏会显示跳转后的地址
permanent : 返回301永久重定向，地址栏会显示跳转后的地址
default: 默认标志，继续会处理当前location中的ngx_http_rewrite_module指令集（rewrite，return等），如果没有return，会开始重新搜索与更改后的URI相匹配的location
rewrite的测试用例

server {
    listen       80 default_server;
    server_name  dev.zdp.com;
    set $flag="default";

    # 当我们访问http://dev.zdp.com/test1/helloworld时，对于不同flag变量返回的结果如下：
    # 当$flag="default"时，会执行后续的ngx_http_rewrite_module命令，所以会重定向到https://newke.com;
    # 当$flag="last"时，不会执行后续的ngx_http_rewrite_module命令，但是会重新匹配location，所以重定向到https://www.baidu.com;
    # 当$flag="break"时，不会执行后续的ngx_http_rewrite_module命令，所以没有找到匹配，失败
    location /test1 {
        rewrite ^/test1\/([^\/]+?) /test2/$1 $flag;
        return 302 https://newke.com;
    }

    location /test2 {
        return 302 https://www.baidu.com;
    }

    location / {

        # 访问 /permanent.html 的时候，页面直接302定向到https://www.baidu.com
        rewrite /permanent.html https://www.baidu.com redirect;

        # 把 /html/*.html => /post/*.html ，301定向
        rewrite ^/html/(.+?).html$ /post/$1.html permanent;

        # 把 /search/key => /search.html?keyword=key
        rewrite ^/search\/([^\/]+?)(\/|$) /search.html?keyword=$1 permanent;
    }
}
```

nginx中if判断如何使用?

只是上面的简单重写很多时候满足不了需求，比如需要判断当文件不存在时、当路径包含xx时等条件，则需要用到if

- Nginx中if语法为：if(condition){...}，对给定的条件condition进行判断。如果为真，大括号内命令将被执行
- if判断规则当表达式只是一个变量时，如果值为空或任何以0开头的字符串都会当做false
- 直接比较变量和内容时，使用=或!=
- ~正则表达式匹配，~*不区分大小写的匹配，!~区分大小写的正则表达式不匹配，满足条件返回true
- -f和!-f用来判断是否存在文件
- -d和!-d用来判断是否存在目录
- -e和!-e用来判断是否存在文件或目录
- -x和!-x用来判断文件是否可执行

if使用举例, if条件中一般会使用到一些变量，这些变量有些是用户定义的，有些是系统本身存在的

```
server {
    if ($http_user_agent ~ MSIE) {
        rewrite ^(.*)$ /msie/$1 break;
    }
    # 如果UA包含"MSIE"，rewrite请求到/msid/目录下

    if ($http_cookie ~* "id=([^;]+)(?:;|$)") {
        set $id $1;
    }
    # 如果cookie匹配正则，设置变量$id等于正则引用部分

    if ($request_method = POST) {
        return 405;
    }
    # 如果提交方法为POST，则返回状态405（Method not allowed）。return不能返回
301,302

    if ($slow) {
        limit_rate 10k;
    }
    # 限速，$slow可以通过 set 指令设置

    if (!-f $request_filename){
        break;
        proxy_pass  http://127.0.0.1;
    }
    # 如果请求的文件名不存在，则反向代理到localhost 。这里的break也是停止rewrite检查

    if ($args ~ post=140){
        rewrite ^ http://example.com/ permanent;
    }
    # 如果query string中包含"post=140"，永久重定向到example.com

    location ~* \.(gif|jpg|png|swf|flv)$ {
        valid_referers none blocked www.jefflei.com www.leizhenfang.com;
        if ($invalid_referer) {
            return 404;
        }
        # 防盗链
    }
}
```

ingress简要概括

ingress 其实就是传统的nginx+lua 在k8s里面的作用就是7层反向代理, ingress 的nginx配置是会根据ingress控制器传递很多参数结合下面的模板生产nginx嵌入lua的主配置文件。

以下配置部分涉及lua及golang的一些语法知识，不做过多讲解，如果对上面nginx部分的接受掌握到位用比对法进行相关配置参考可进行梳理。最终的nginx confg在ingress pod里面可见。

golang变量数据结构

```
# 主要参考这里面的变量结构体字段定义，能够针对那些参数做调整优化然后参考官方文档进行范围确认全局或者局部使用
# https://github.com/kubernetes/ingress-nginx/blob/master/internal/ingress/controller/config/config.go

type TemplateConfig struct {
    ProxySetHeaders            map[string]string
    AddHeaders                 map[string]string
    BacklogSize                int
    Backends                   []*ingress.Backend
    PassthroughBackends        []*ingress.SSLPassthroughBackend
    Servers                    []*ingress.Server
    TCPBackends                []ingress.L4Service
    UDPBackends                []ingress.L4Service
    HealthzURI                 string
    Cfg                        Configuration
    IsIPV6Enabled              bool
    IsSSLPassthroughEnabled    bool
    NginxStatusIpv4Whitelist   []string
    NginxStatusIpv6Whitelist   []string
    RedirectServers            interface{}
    ListenPorts                *ListenPorts
    PublishService             *apiv1.Service
    EnableMetrics              bool
    MaxmindEditionFiles        []string
    MonitorMaxBatchSize        int

    PID        string
    StatusPath string
    StatusPort int
    StreamPort int
}
```

下面golang模板配置

```
{{ $all := . }}
{{ $servers := .Servers }}
{{ $cfg := .Cfg }}
{{ $IsIPV6Enabled := .IsIPV6Enabled }}
{{ $healthzURI := .HealthzURI }}
{{ $backends := .Backends }}
{{ $proxyHeaders := .ProxySetHeaders }}
{{ $addHeaders := .AddHeaders }}

# 从这里可以看出配置是基于checksum计算进行的更新判定
# Configuration checksum: {{ $all.Cfg.Checksum }}

# setup custom paths that do not require root access
```

```
pid {{ .PID }};

{{ if $cfg.UseGeoIP2 }}
load_module /etc/nginx/modules/ngx_http_geoip2_module.so;
{{ end }}

{{ if $cfg.EnableBrotli }}
load_module /etc/nginx/modules/ngx_http_brotli_filter_module.so;
load_module /etc/nginx/modules/ngx_http_brotli_static_module.so;
{{ end }}

{{ if (shouldLoadInfluxDBModule $servers) }}
load_module /etc/nginx/modules/ngx_http_influxdb_module.so;
{{ end }}

{{ if (shouldLoadAuthDigestModule $servers) }}
load_module /etc/nginx/modules/ngx_http_auth_digest_module.so;
{{ end }}

{{ if (shouldLoadModSecurityModule $cfg $servers) }}
load_module /etc/nginx/modules/ngx_http_modsecurity_module.so;
{{ end }}

{{ if (shouldLoadOpentracingModule $cfg $servers) }}
load_module /etc/nginx/modules/ngx_http_opentracing_module.so;
{{ end }}

daemon off;

worker_processes {{ $cfg.WorkerProcesses }};
{{ if gt (len $cfg.WorkerCPUAffinity) 0 }}
worker_cpu_affinity {{ $cfg.WorkerCPUAffinity }};
{{ end }}

worker_rlimit_nofile {{ $cfg.MaxWorkerOpenFiles }};

{{/* http://nginx.org/en/docs/ngx_core_module.html#worker_shutdown_timeout
*/}}
{{/* avoid waiting too long during a reload */}}
worker_shutdown_timeout {{ $cfg.WorkerShutdownTimeout }} ;

{{ if not (empty $cfg.MainSnippet) }}
{{ $cfg.MainSnippet }}
{{ end }}

events {
    multi_accept        {{ if $cfg.EnableMultiAccept }}on{{ else }}off{{
end }};
    worker_connections  {{ $cfg.MaxWorkerConnections }};
    use                 epoll;
}

http {
    lua_package_path "/etc/nginx/lua/?.lua;;";
    {{ buildLuaSharedDictionaries $cfg $servers }}

    init_by_lua_block {
        collectgarbage("collect")

        -- init modules
        local ok, res

        ok, res = pcall(require, "lua_ingress")
        if not ok then
          error("require failed: " .. tostring(res))
        else
          lua_ingress = res
          lua_ingress.set_config({{ configForLua $all }})
        end

        ok, res = pcall(require, "configuration")
        if not ok then
          error("require failed: " .. tostring(res))
        else
          configuration = res
        end

        ok, res = pcall(require, "balancer")
        if not ok then
          error("require failed: " .. tostring(res))
        else
          balancer = res
        end

        {{ if $all.EnableMetrics }}
        ok, res = pcall(require, "monitor")
        if not ok then
          error("require failed: " .. tostring(res))
        else
          monitor = res
        end
        {{ end }}

        ok, res = pcall(require, "certificate")
        if not ok then
          error("require failed: " .. tostring(res))
        else
          certificate = res
          certificate.is_ocsp_stapling_enabled = {{ $cfg.EnableOCSP }}
        end

        ok, res = pcall(require, "plugins")
        if not ok then
          error("require failed: " .. tostring(res))
        else
          plugins = res
        end
        -- load all plugins that'll be used here
        plugins.init({ {{ range  $idx, $plugin := $cfg.Plugins }}{{ if $idx
}},{{ end }}{{ $plugin | quote }}{{ end }} })
    }

    init_worker_by_lua_block {
        lua_ingress.init_worker()
        balancer.init_worker()
        {{ if $all.EnableMetrics }}
        monitor.init_worker({{ $all.MonitorMaxBatchSize }})
        {{ end }}

        plugins.run()
    }

    {{/* Enable the real_ip module only if we use either X-Forwarded
headers or Proxy Protocol. */}}
    {{/* we use the value of the real IP for the geo_ip module */}}
    {{ if or (or $cfg.UseForwardedHeaders $cfg.UseProxyProtocol)
$cfg.EnableRealIp }}
    {{ if $cfg.UseProxyProtocol }}
    real_ip_header      proxy_protocol;
    {{ else }}
    real_ip_header      {{ $cfg.ForwardedForHeader }};
    {{ end }}

    real_ip_recursive   on;
    {{ range $trusted_ip := $cfg.ProxyRealIPCIDR }}
    set_real_ip_from    {{ $trusted_ip }};
    {{ end }}
    {{ end }}

    {{ if $all.Cfg.EnableModsecurity }}
    modsecurity on;

    modsecurity_rules_file /etc/nginx/modsecurity/modsecurity.conf;

    {{ if $all.Cfg.EnableOWASPCoreRules }}
    modsecurity_rules_file /etc/nginx/owasp-modsecurity-crs/nginx-
modsecurity.conf;
    {{ else if (not (empty $all.Cfg.ModsecuritySnippet)) }}
    modsecurity_rules '
      {{ $all.Cfg.ModsecuritySnippet }}
    ';
    {{ end }}

    {{ end }}

    {{ if $cfg.UseGeoIP }}
    {{/* databases used to determine the country depending on the client IP
address */}}
    {{/* http://nginx.org/en/docs/http/ngx_http_geoip_module.html */}}
    {{/* this is require to calculate traffic for individual country using
GeoIP in the status page */}}
```

```
    geoip_country       /etc/nginx/geoip/GeoIP.dat;
    geoip_city          /etc/nginx/geoip/GeoLiteCity.dat;
    geoip_org           /etc/nginx/geoip/GeoIPASNum.dat;
    geoip_proxy_recursive on;
    {{ end }}

    {{ if $cfg.UseGeoIP2 }}
    # https://github.com/leev/ngx_http_geoip2_module#example-usage

    {{ range $index, $file := $all.MaxmindEditionFiles }}
    {{ if eq $file "GeoLite2-City.mmdb" }}
    geoip2 /etc/nginx/geoip/GeoLite2-City.mmdb {
        $geoip2_city_country_code source=$remote_addr country iso_code;
        $geoip2_city_country_name source=$remote_addr country names en;
        $geoip2_city source=$remote_addr city names en;
        $geoip2_postal_code source=$remote_addr postal code;
        $geoip2_dma_code source=$remote_addr location metro_code;
        $geoip2_latitude source=$remote_addr location latitude;
        $geoip2_longitude source=$remote_addr location longitude;
        $geoip2_time_zone source=$remote_addr location time_zone;
        $geoip2_region_code source=$remote_addr subdivisions 0 iso_code;
        $geoip2_region_name source=$remote_addr subdivisions 0 names en;
    }
    {{ end }}

    {{ if eq $file "GeoIP2-City.mmdb" }}
    geoip2 /etc/nginx/geoip/GeoIP2-City.mmdb {
        $geoip2_city_country_code source=$remote_addr country iso_code;
        $geoip2_city_country_name source=$remote_addr country names en;
        $geoip2_city source=$remote_addr city names en;
        $geoip2_postal_code source=$remote_addr postal code;
        $geoip2_dma_code source=$remote_addr location metro_code;
        $geoip2_latitude source=$remote_addr location latitude;
        $geoip2_longitude source=$remote_addr location longitude;
        $geoip2_time_zone source=$remote_addr location time_zone;
        $geoip2_region_code source=$remote_addr subdivisions 0 iso_code;
        $geoip2_region_name source=$remote_addr subdivisions 0 names en;
    }
    {{ end }}

    {{ if eq $file "GeoLite2-ASN.mmdb" }}
    geoip2 /etc/nginx/geoip/GeoLite2-ASN.mmdb {
        $geoip2_asn source=$remote_addr autonomous_system_number;
        $geoip2_org source=$remote_addr autonomous_system_organization;
    }
    {{ end }}

    {{ if eq $file "GeoIP2-ASN.mmdb" }}
    geoip2 /etc/nginx/geoip/GeoIP2-ASN.mmdb {
        $geoip2_asn source=$remote_addr autonomous_system_number;
        $geoip2_org source=$remote_addr autonomous_system_organization;
    }
    {{ end }}
    {{ if eq $file "GeoIP2-ISP.mmdb" }}
    geoip2 /etc/nginx/geoip/GeoIP2-ISP.mmdb {
        $geoip2_isp isp;
        $geoip2_isp_org organization;
    }
    {{ end }}

    {{ if eq $file "GeoIP2-Connection-Type.mmdb" }}
    geoip2 /etc/nginx/geoip/GeoIP2-Connection-Type.mmdb {
        $geoip2_connection_type connection_type;
    }
    {{ end }}

    {{ if eq $file "GeoIP2-Anonymous-IP.mmdb" }}
    geoip2 /etc/nginx/geoip/GeoIP2-Anonymous-IP.mmdb {
        $geoip2_is_anon source=$remote_addr is_anonymous;
        $geoip2_is_hosting_provider source=$remote_addr
is_hosting_provider;
        $geoip2_is_public_proxy source=$remote_addr is_public_proxy;
    }
    {{ end }}

    {{ end }}

    {{ end }}

    aio                 threads;
    aio_write           on;

    tcp_nopush          on;
    tcp_nodelay         on;

    log_subrequest      on;

    reset_timedout_connection on;

    keepalive_timeout  {{ $cfg.KeepAlive }}s;
    keepalive_requests {{ $cfg.KeepAliveRequests }};

    client_body_temp_path           /tmp/client-body;
    fastcgi_temp_path               /tmp/fastcgi-temp;
    proxy_temp_path                 /tmp/proxy-temp;
    ajp_temp_path                   /tmp/ajp-temp;

    client_header_buffer_size       {{ $cfg.ClientHeaderBufferSize }};
    client_header_timeout           {{ $cfg.ClientHeaderTimeout }}s;
    large_client_header_buffers     {{ $cfg.LargeClientHeaderBuffers }};
    client_body_buffer_size         {{ $cfg.ClientBodyBufferSize }};
    client_body_timeout             {{ $cfg.ClientBodyTimeout }}s;

    http2_max_field_size            {{ $cfg.HTTP2MaxFieldSize }};
    http2_max_header_size           {{ $cfg.HTTP2MaxHeaderSize }};
    http2_max_requests              {{ $cfg.HTTP2MaxRequests }};
    http2_max_concurrent_streams    {{ $cfg.HTTP2MaxConcurrentStreams }};
    types_hash_max_size             2048;
    server_names_hash_max_size      {{ $cfg.ServerNameHashMaxSize }};
    server_names_hash_bucket_size   {{ $cfg.ServerNameHashBucketSize }};
    map_hash_bucket_size            {{ $cfg.MapHashBucketSize }};

    proxy_headers_hash_max_size     {{ $cfg.ProxyHeadersHashMaxSize }};
    proxy_headers_hash_bucket_size  {{ $cfg.ProxyHeadersHashBucketSize }};

    variables_hash_bucket_size      {{ $cfg.VariablesHashBucketSize }};
    variables_hash_max_size         {{ $cfg.VariablesHashMaxSize }};

    underscores_in_headers          {{ if $cfg.EnableUnderscoresInHeaders
}}on{{ else }}off{{ end }};
    ignore_invalid_headers          {{ if $cfg.IgnoreInvalidHeaders }}on{{
else }}off{{ end }};

    limit_req_status                {{ $cfg.LimitReqStatusCode }};
    limit_conn_status               {{ $cfg.LimitConnStatusCode }};

    {{ buildOpentracing $cfg $servers }}

    include /etc/nginx/mime.types;
    default_type {{ $cfg.DefaultType }};

    {{ if $cfg.EnableBrotli }}
    brotli on;
    brotli_comp_level {{ $cfg.BrotliLevel }};
    brotli_types {{ $cfg.BrotliTypes }};
    {{ end }}

    {{ if $cfg.UseGzip }}
    gzip on;
    gzip_comp_level {{ $cfg.GzipLevel }};
    gzip_http_version 1.1;
    gzip_min_length {{ $cfg.GzipMinLength}};
    gzip_types {{ $cfg.GzipTypes }};
    gzip_proxied any;
    gzip_vary on;
    {{ end }}

    # Custom headers for response
    {{ range $k, $v := $addHeaders }}
    more_set_headers {{ printf "%s: %s" $k $v | quote }};
    {{ end }}

    server_tokens {{ if $cfg.ShowServerTokens }}on{{ else }}off{{ end }};
    {{ if not $cfg.ShowServerTokens }}
    more_clear_headers Server;
    {{ end }}

    # disable warnings
    uninitialized_variable_warn off;
    # Additional available variables:
    # $namespace
    # $ingress_name
    # $service_name
    # $service_port
    log_format upstreaminfo {{ if $cfg.LogFormatEscapeJSON }}escape=json {{
end }}'{{ $cfg.LogFormatUpstream }}';

    {{/* map urls that should not appear in access.log */}}
    {{/* http://nginx.org/en/docs/http/ngx_http_log_module.html#access_log
*/}}
    map $request_uri $loggable {
        {{ range $reqUri := $cfg.SkipAccessLogURLs }}
        {{ $reqUri }} 0;{{ end }}
        default 1;
    }

    {{ if or $cfg.DisableAccessLog $cfg.DisableHTTPAccessLog }}
    access_log off;
    {{ else }}
    {{ if $cfg.EnableSyslog }}
    access_log syslog:server={{ $cfg.SyslogHost }}:{{ $cfg.SyslogPort }}
upstreaminfo if=$loggable;
    {{ else }}
    access_log {{ or $cfg.HttpAccessLogPath $cfg.AccessLogPath }}
upstreaminfo {{ $cfg.AccessLogParams }} if=$loggable;
    {{ end }}
    {{ end }}

    {{ if $cfg.EnableSyslog }}
    error_log syslog:server={{ $cfg.SyslogHost }}:{{ $cfg.SyslogPort }} {{
$cfg.ErrorLogLevel }};
    {{ else }}
    error_log  {{ $cfg.ErrorLogPath }} {{ $cfg.ErrorLogLevel }};
    {{ end }}

    {{ buildResolvers $cfg.Resolver $cfg.DisableIpv6DNS }}

    # See https://www.nginx.com/blog/websocket-nginx
    map $http_upgrade $connection_upgrade {
        default          upgrade;
        {{ if (gt $cfg.UpstreamKeepaliveConnections 0) }}
        # See
http://nginx.org/en/docs/http/ngx_http_upstream_module.html#keepalive
        ''               '';
        {{ else }}
        ''               close;
        {{ end }}
    }

    # Reverse proxies can detect if a client provides a X-Request-ID
header, and pass it on to the backend server.
    # If no such header is provided, it can provide a random value.
    map $http_x_request_id $req_id {
        default   $http_x_request_id;
        {{ if $cfg.GenerateRequestID }}
        ""        $request_id;
        {{ end }}
    }

    {{ if and $cfg.UseForwardedHeaders $cfg.ComputeFullForwardedFor }}
    # We can't use $proxy_add_x_forwarded_for because the realip module
    # replaces the remote_addr too soon
    map $http_x_forwarded_for $full_x_forwarded_for {
        {{ if $all.Cfg.UseProxyProtocol }}
        default          "$http_x_forwarded_for, $proxy_protocol_addr";
        ''               "$proxy_protocol_addr";
        {{ else }}
        default          "$http_x_forwarded_for, $realip_remote_addr";
        ''               "$realip_remote_addr";
        {{ end}}
    }

    {{ end }}

    # Create a variable that contains the literal $ character.
    # This works because the geo module will not resolve variables.
    geo $literal_dollar {
        default "$";
    }

    server_name_in_redirect off;
    port_in_redirect        off;

    ssl_protocols {{ $cfg.SSLProtocols }};

    ssl_early_data {{ if $cfg.SSLEarlyData }}on{{ else }}off{{ end }};

    # turn on session caching to drastically improve performance
    {{ if $cfg.SSLSessionCache }}
    ssl_session_cache builtin:1000 shared:SSL:{{ $cfg.SSLSessionCacheSize
}};
    ssl_session_timeout {{ $cfg.SSLSessionTimeout }};
    {{ end }}

    # allow configuring ssl session tickets
    ssl_session_tickets {{ if $cfg.SSLSessionTickets }}on{{ else }}off{{
end }};

    {{ if not (empty $cfg.SSLSessionTicketKey ) }}
    ssl_session_ticket_key /etc/nginx/tickets.key;
    {{ end }}

    # slightly reduce the time-to-first-byte
    ssl_buffer_size {{ $cfg.SSLBufferSize }};

    {{ if not (empty $cfg.SSLCiphers) }}
    # allow configuring custom ssl ciphers
    ssl_ciphers '{{ $cfg.SSLCiphers }}';
    ssl_prefer_server_ciphers on;
    {{ end }}

    {{ if not (empty $cfg.SSLDHParam) }}
    # allow custom DH file
http://nginx.org/en/docs/http/ngx_http_ssl_module.html#ssl_dhparam
    ssl_dhparam {{ $cfg.SSLDHParam }};
    {{ end }}

    ssl_ecdh_curve {{ $cfg.SSLECDHCurve }};

    # PEM sha: {{ $cfg.DefaultSSLCertificate.PemSHA }}
    ssl_certificate     {{ $cfg.DefaultSSLCertificate.PemFileName }};
    ssl_certificate_key {{ $cfg.DefaultSSLCertificate.PemFileName }};

    {{ if gt (len $cfg.CustomHTTPErrors) 0 }}
    proxy_intercept_errors on;
    {{ end }}

    {{ range $errCode := $cfg.CustomHTTPErrors }}
    error_page {{ $errCode }} = @custom_upstream-default-backend_{{
$errCode }};{{ end }}

    proxy_ssl_session_reuse on;

    {{ if $cfg.AllowBackendServerHeader }}
    proxy_pass_header Server;
    {{ end }}

    {{ range $header := $cfg.HideHeaders }}proxy_hide_header {{ $header }};
    {{ end }}

    {{ if not (empty $cfg.HTTPSnippet) }}
    # Custom code snippet configured in the configuration configmap
    {{ $cfg.HTTPSnippet }}
    {{ end }}

    upstream upstream_balancer {
        ### Attention!!!
        #
        # We no longer create "upstream" section for every backend.
        # Backends are handled dynamically using Lua. If you would like to
debug
        # and see what backends ingress-nginx has in its memory you can
        # install our kubectl plugin https://kubernetes.github.io/ingress-
nginx/kubectl-plugin.
        # Once you have the plugin you can use "kubectl ingress-nginx
backends" command to
        # inspect current backends.
        #
        ###

        server 0.0.0.1; # placeholder
        balancer_by_lua_block {
          balancer.balance()
        }

        {{ if (gt $cfg.UpstreamKeepaliveConnections 0) }}
        keepalive {{ $cfg.UpstreamKeepaliveConnections }};

        keepalive_timeout  {{ $cfg.UpstreamKeepaliveTimeout }}s;
        keepalive_requests {{ $cfg.UpstreamKeepaliveRequests }};
        {{ end }}
    }

    {{ range $rl := (filterRateLimits $servers ) }}
    # Ratelimit {{ $rl.Name }}
    geo $remote_addr $whitelist_{{ $rl.ID }} {
        default 0;
        {{ range $ip := $rl.Whitelist }}
        {{ $ip }} 1;{{ end }}
    }

    # Ratelimit {{ $rl.Name }}
    map $whitelist_{{ $rl.ID }} $limit_{{ $rl.ID }} {
        0 {{ $cfg.LimitConnZoneVariable }};
        1 "";
    }
    {{ end }}

    {{/* build all the required rate limit zones. Each annotation requires
a dedicated zone */}}
    {{/* 1MB -> 16 thousand 64-byte states or about 8 thousand 128-byte
states */}}
    {{ range $zone := (buildRateLimitZones $servers) }}
    {{ $zone }}
    {{ end }}

    # Cache for internal auth checks
    proxy_cache_path /tmp/nginx-cache-auth levels=1:2
keys_zone=auth_cache:10m max_size=128m inactive=30m use_temp_path=off;

    # Global filters
    {{ range $ip := $cfg.BlockCIDRs }}deny {{ trimSpace $ip }};
    {{ end }}

    {{ if gt (len $cfg.BlockUserAgents) 0 }}
    map $http_user_agent $block_ua {
        default 0;

        {{ range $ua := $cfg.BlockUserAgents }}{{ trimSpace $ua }} 1;
        {{ end }}
    }
    {{ end }}

    {{ if gt (len $cfg.BlockReferers) 0 }}
    map $http_referer $block_ref {
        default 0;

        {{ range $ref := $cfg.BlockReferers }}{{ trimSpace $ref }} 1;
        {{ end }}
    }
    {{ end }}

    {{/* Build server redirects (from/to www) */}}
    {{ range $redirect := .RedirectServers }}
    ## start server {{ $redirect.From }}
    server {
        server_name {{ $redirect.From }};

        {{ buildHTTPListener  $all $redirect.From }}
        {{ buildHTTPSListener $all $redirect.From }}

        ssl_certificate_by_lua_block {
            certificate.call()
        }

        {{ if gt (len $cfg.BlockUserAgents) 0 }}
        if ($block_ua) {
           return 403;
        }
        {{ end }}
        {{ if gt (len $cfg.BlockReferers) 0 }}
        if ($block_ref) {
           return 403;
        }
        {{ end }}

        set_by_lua_block $redirect_to {
            local request_uri = ngx.var.request_uri
            if string.sub(request_uri, -1) == "/" then
                request_uri = string.sub(request_uri, 1, -2)
            end

            {{ if ne $all.ListenPorts.HTTPS 443 }}
            {{ $redirect_port := (printf ":%v" $all.ListenPorts.HTTPS) }}
            return string.format("%s://%s%s%s", ngx.var.scheme, "{{
$redirect.To }}", "{{ $redirect_port }}", request_uri)
            {{ else }}
            return string.format("%s://%s%s", ngx.var.scheme, "{{
$redirect.To }}", request_uri)
            {{ end }}
        }

        return {{ $all.Cfg.HTTPRedirectCode }} $redirect_to;
    }
    ## end server {{ $redirect.From }}
    {{ end }}

    {{ range $server := $servers }}
    ## start server {{ $server.Hostname }}
    server {
        server_name {{ $server.Hostname }} {{range $server.Aliases }}{{ .
}} {{ end }};

        {{ if gt (len $cfg.BlockUserAgents) 0 }}
        if ($block_ua) {
           return 403;
        }
        {{ end }}
        {{ if gt (len $cfg.BlockReferers) 0 }}
        if ($block_ref) {
           return 403;
        }
        {{ end }}

        {{ template "SERVER" serverConfig $all $server }}

        {{ if not (empty $cfg.ServerSnippet) }}
        # Custom code snippet configured in the configuration configmap
        {{ $cfg.ServerSnippet }}
        {{ end }}

        {{ template "CUSTOM_ERRORS" (buildCustomErrorDeps "upstream-
default-backend" $cfg.CustomHTTPErrors $all.EnableMetrics) }}
    }
    ## end server {{ $server.Hostname }}

    {{ end }}

    # backend for when default-backend-service is not configured or it does
not have endpoints
    server {
        listen {{ $all.ListenPorts.Default }} default_server {{ if
$all.Cfg.ReusePort }}reuseport{{ end }} backlog={{ $all.BacklogSize }};
        {{ if $IsIPV6Enabled }}listen [::]:{{ $all.ListenPorts.Default }}
default_server {{ if $all.Cfg.ReusePort }}reuseport{{ end }} backlog={{
$all.BacklogSize }};{{ end }}
        set $proxy_upstream_name "internal";

        access_log off;

        location / {
          return 404;
        }
    }

    # default server, used for NGINX healthcheck and access to nginx stats
    server {
        listen 127.0.0.1:{{ .StatusPort }};
        set $proxy_upstream_name "internal";

        keepalive_timeout 0;
```

```
            gzip off;

            access_log off;

            {{ if $cfg.EnableOpentracing }}
            opentracing off;
            {{ end }}

            location {{ $healthzURI }} {
                return 200;
            }

            location /is-dynamic-lb-initialized {
                content_by_lua_block {
                    local configuration = require("configuration")
                    local backend_data = configuration.get_backends_data()
                    if not backend_data then
                        ngx.exit(ngx.HTTP_INTERNAL_SERVER_ERROR)
                        return
                    end

                    ngx.say("OK")
                    ngx.exit(ngx.HTTP_OK)
                }
            }

            location {{ .StatusPath }} {
                stub_status on;
            }

            location /configuration {
                client_max_body_size                    {{
luaConfigurationRequestBodySize $cfg }}m;
                client_body_buffer_size                 {{
luaConfigurationRequestBodySize $cfg }}m;
                proxy_buffering                         off;

                content_by_lua_block {
                  configuration.call()
                }
            }

            location / {
                content_by_lua_block {
                    ngx.exit(ngx.HTTP_NOT_FOUND)
                }
            }
        }
}

stream {
    lua_package_path "/etc/nginx/lua/?.lua;/etc/nginx/lua/vendor/?.lua;;";

    lua_shared_dict tcp_udp_configuration_data 5M;
        init_by_lua_block {
            collectgarbage("collect")

            -- init modules
            local ok, res

            ok, res = pcall(require, "configuration")
            if not ok then
              error("require failed: " .. tostring(res))
            else
              configuration = res
            end

            ok, res = pcall(require, "tcp_udp_configuration")
            if not ok then
              error("require failed: " .. tostring(res))
            else
              tcp_udp_configuration = res
            end

            ok, res = pcall(require, "tcp_udp_balancer")
            if not ok then
              error("require failed: " .. tostring(res))
            else
              tcp_udp_balancer = res
            end
        }

        init_worker_by_lua_block {
            tcp_udp_balancer.init_worker()
        }

        lua_add_variable $proxy_upstream_name;

        log_format log_stream '{{ $cfg.LogFormatStream }}';

        {{ if or $cfg.DisableAccessLog $cfg.DisableStreamAccessLog }}
    access_log off;
    {{ else }}
    access_log {{ or $cfg.StreamAccessLogPath $cfg.AccessLogPath }}
log_stream {{ $cfg.AccessLogParams }};
    {{ end }}

    error_log  {{ $cfg.ErrorLogPath }};

    {{ if $cfg.EnableRealIp }}
    {{ range $trusted_ip := $cfg.ProxyRealIPCIDR }}
    set_real_ip_from    {{ $trusted_ip }};
    {{ end }}
    {{ end }}

    upstream upstream_balancer {
            server 0.0.0.1:1234; # placeholder
        balancer_by_lua_block {
          tcp_udp_balancer.balance()
        }
    }

    server {
        listen 127.0.0.1:{{ .StreamPort }};

        access_log off;

        content_by_lua_block {
          tcp_udp_configuration.call()
        }
    }

    # TCP services
    {{ range $tcpServer := .TCPBackends }}
    server {
        preread_by_lua_block {
            ngx.var.proxy_upstream_name="tcp-{{
$tcpServer.Backend.Namespace }}-{{ $tcpServer.Backend.Name }}-{{
$tcpServer.Backend.Port }}";
        }

        {{ range $address := $all.Cfg.BindAddressIpv4 }}
        listen                  {{ $address }}:{{ $tcpServer.Port }}{{ if
$tcpServer.Backend.ProxyProtocol.Decode }} proxy_protocol{{ end }};
        {{ else }}
        listen                  {{ $tcpServer.Port }}{{ if
$tcpServer.Backend.ProxyProtocol.Decode }} proxy_protocol{{ end }};
        {{ end }}
        {{ if $IsIPV6Enabled }}
        {{ range $address := $all.Cfg.BindAddressIpv6 }}
        listen                  {{ $address }}:{{ $tcpServer.Port }}{{ if
$tcpServer.Backend.ProxyProtocol.Decode }} proxy_protocol{{ end }};
        {{ else }}
        listen                  [::]:{{ $tcpServer.Port }}{{ if
$tcpServer.Backend.ProxyProtocol.Decode }} proxy_protocol{{ end }};
        {{ end }}
        {{ end }}
        proxy_timeout           {{ $cfg.ProxyStreamTimeout }};
        proxy_pass              upstream_balancer;
        {{ if $tcpServer.Backend.ProxyProtocol.Encode }}
        proxy_protocol          on;
        {{ end }}
    }
    {{ end }}

    # UDP services
    {{ range $udpServer := .UDPBackends }}
    server {
        preread_by_lua_block {
            ngx.var.proxy_upstream_name="udp-{{
```

```
$udpServer.Backend.Namespace }}-{{ $udpServer.Backend.Name }}-{{
$udpServer.Backend.Port }}";
        }

        {{ range $address := $all.Cfg.BindAddressIpv4 }}
        listen                  {{ $address }}:{{ $udpServer.Port }} udp;
        {{ else }}
        listen                  {{ $udpServer.Port }} udp;
        {{ end }}
        {{ if $IsIPV6Enabled }}
        {{ range $address := $all.Cfg.BindAddressIpv6 }}
        listen                  {{ $address }}:{{ $udpServer.Port }} udp;
        {{ else }}
        listen                  [::]:{{ $udpServer.Port }} udp;
        {{ end }}
        {{ end }}
        proxy_responses         {{ $cfg.ProxyStreamResponses }};
        proxy_timeout           {{ $cfg.ProxyStreamTimeout }};
        proxy_pass              upstream_balancer;
    }
    {{ end }}
}

{{/* definition of templates to avoid repetitions */}}
{{ define "CUSTOM_ERRORS" }}
        {{ $enableMetrics := .EnableMetrics }}
        {{ $upstreamName := .UpstreamName }}
        {{ range $errCode := .ErrorCodes }}
        location @custom_{{ $upstreamName }}_{{ $errCode }} {
            internal;

            proxy_intercept_errors off;

            proxy_set_header       X-Code             {{ $errCode }};
            proxy_set_header       X-Format           $http_accept;
            proxy_set_header       X-Original-URI     $request_uri;
            proxy_set_header       X-Namespace        $namespace;
            proxy_set_header       X-Ingress-Name     $ingress_name;
            proxy_set_header       X-Service-Name     $service_name;
            proxy_set_header       X-Service-Port     $service_port;
            proxy_set_header       X-Request-ID       $req_id;
            proxy_set_header       Host               $best_http_host;

            set $proxy_upstream_name {{ $upstreamName | quote }};

            rewrite                (.*) / break;

            proxy_pass            http://upstream_balancer;
            log_by_lua_block {
                {{ if $enableMetrics }}
                monitor.call()
                {{ end }}
            }
        }
        {{ end }}
{{ end }}

{{/* CORS support from https://michielkalkman.com/snippets/nginx-cors-open-
configuration.html */}}
{{ define "CORS" }}
     {{ $cors := .CorsConfig }}
     # Cors Preflight methods needs additional options and different Return
Code
     if ($request_method = 'OPTIONS') {
        more_set_headers 'Access-Control-Allow-Origin: {{
$cors.CorsAllowOrigin }}';
        {{ if $cors.CorsAllowCredentials }} more_set_headers 'Access-
Control-Allow-Credentials: {{ $cors.CorsAllowCredentials }}'; {{ end }}
        more_set_headers 'Access-Control-Allow-Methods: {{
$cors.CorsAllowMethods }}';
        more_set_headers 'Access-Control-Allow-Headers: {{
$cors.CorsAllowHeaders }}';
        {{ if not (empty $cors.CorsExposeHeaders) }} more_set_headers
'Access-Control-Expose-Headers: {{ $cors.CorsExposeHeaders }}'; {{ end }}
        more_set_headers 'Access-Control-Max-Age: {{ $cors.CorsMaxAge }}';
        more_set_headers 'Content-Type: text/plain charset=UTF-8';
        more_set_headers 'Content-Length: 0';
        return 204;
     }

        more_set_headers 'Access-Control-Allow-Origin: {{
$cors.CorsAllowOrigin }}';
        {{ if $cors.CorsAllowCredentials }} more_set_headers 'Access-
Control-Allow-Credentials: {{ $cors.CorsAllowCredentials }}'; {{ end }}
        {{ if not (empty $cors.CorsExposeHeaders) }} more_set_headers
'Access-Control-Expose-Headers: {{ $cors.CorsExposeHeaders }}'; {{ end }}

{{ end }}

{{/* definition of server-template to avoid repetitions with server-alias
*/}}
{{ define "SERVER" }}
        {{ $all := .First }}
        {{ $server := .Second }}

        {{ buildHTTPListener  $all $server.Hostname }}
        {{ buildHTTPSListener $all $server.Hostname }}

        set $proxy_upstream_name "-";

        ssl_certificate_by_lua_block {
            certificate.call()
        }

        {{ if not (empty $server.AuthTLSError) }}
        # {{ $server.AuthTLSError }}
        return 403;
        {{ else }}
        {{ if not (empty $server.CertificateAuth.CAFileName) }}
        # PEM sha: {{ $server.CertificateAuth.CASHA }}
        ssl_client_certificate                  {{
$server.CertificateAuth.CAFileName }};
        ssl_verify_client                       {{
$server.CertificateAuth.VerifyClient }};
        ssl_verify_depth                        {{
$server.CertificateAuth.ValidationDepth }};

        {{ if not (empty $server.CertificateAuth.CRLFileName) }}
        # PEM sha: {{ $server.CertificateAuth.CRLSHA }}
        ssl_crl                                 {{
$server.CertificateAuth.CRLFileName }};
        {{ end }}

        {{ if not (empty $server.CertificateAuth.ErrorPage)}}
        error_page 495 496 = {{ $server.CertificateAuth.ErrorPage }};
        {{ end }}
        {{ end }}

        {{ if not (empty $server.ProxySSL.CAFileName) }}
        # PEM sha: {{ $server.ProxySSL.CASHA }}
        proxy_ssl_trusted_certificate           {{
$server.ProxySSL.CAFileName }};
        proxy_ssl_ciphers                       {{ $server.ProxySSL.Ciphers
}};
        proxy_ssl_protocols                     {{
$server.ProxySSL.Protocols }};
        proxy_ssl_verify                        {{ $server.ProxySSL.Verify
}};
        proxy_ssl_verify_depth                  {{
$server.ProxySSL.VerifyDepth }};
        {{ if not (empty $server.ProxySSL.ProxySSLName) }}
        proxy_ssl_name                          {{
$server.ProxySSL.ProxySSLName }};
        proxy_ssl_server_name                   {{
$server.ProxySSL.ProxySSLServerName }};
        {{ end }}
        {{ end }}

        {{ if not (empty $server.ProxySSL.PemFileName) }}
        proxy_ssl_certificate                   {{
$server.ProxySSL.PemFileName }};
        proxy_ssl_certificate_key               {{
$server.ProxySSL.PemFileName }};
        {{ end }}

        {{ if not (empty $server.SSLCiphers) }}
        ssl_ciphers                             {{ $server.SSLCiphers }};
        {{ end }}

        {{ if not (empty $server.SSLPreferServerCiphers) }}
        ssl_prefer_server_ciphers               {{
```

```
$server.SSLPreferServerCiphers }};
        {{ end }}

        {{ if not (empty $server.ServerSnippet) }}
        {{ $server.ServerSnippet }}
        {{ end }}

        {{ range $errorLocation := (buildCustomErrorLocationsPerServer
$server) }}
        {{ template "CUSTOM_ERRORS" (buildCustomErrorDeps
$errorLocation.UpstreamName $errorLocation.Codes $all.EnableMetrics) }}
        {{ end }}

        {{ buildMirrorLocations $server.Locations }}

        {{ $enforceRegex := enforceRegexModifier $server.Locations }}
        {{ range $location := $server.Locations }}
        {{ $path := buildLocation $location $enforceRegex }}
        {{ $proxySetHeader := proxySetHeader $location }}
        {{ $authPath := buildAuthLocation $location
$all.Cfg.GlobalExternalAuth.URL }}
        {{ $applyGlobalAuth := shouldApplyGlobalAuth $location
$all.Cfg.GlobalExternalAuth.URL }}

        {{ $externalAuth := $location.ExternalAuth }}
        {{ if eq $applyGlobalAuth true }}
        {{ $externalAuth = $all.Cfg.GlobalExternalAuth }}
        {{ end }}

        {{ if not (empty $location.Rewrite.AppRoot) }}
        if ($uri = /) {
            return 302 $scheme://$http_host{{ $location.Rewrite.AppRoot }};
        }
        {{ end }}

        {{ if $authPath }}
        location = {{ $authPath }} {
            internal;

            {{ if $all.Cfg.EnableOpentracing }}
            opentracing on;
            opentracing_propagate_context;
            {{ end }}

            {{ if $externalAuth.AuthCacheKey }}
            set $tmp_cache_key '{{ $server.Hostname }}{{ $authPath }}{{
$externalAuth.AuthCacheKey }}';
            set $cache_key '';

            rewrite_by_lua_block {
                ngx.var.cache_key =
ngx.encode_base64(ngx.sha1_bin(ngx.var.tmp_cache_key))
            }
            proxy_cache auth_cache;

            {{- range $dur := $externalAuth.AuthCacheDuration }}
            proxy_cache_valid {{ $dur }};
            {{- end }}

            proxy_cache_key "$cache_key";
            {{ end }}

            # ngx_auth_request module overrides variables in the parent 
request,
            # therefore we have to explicitly set this variable again so 
that when the parent request
            # resumes it has the correct value set for this variable so 
that Lua can pick backend correctly
            set $proxy_upstream_name {{ buildUpstreamName $location | quote 
}};

            proxy_pass_request_body     off;
            proxy_set_header            Content-Length          "";
            proxy_set_header            X-Forwarded-Proto       "";
            proxy_set_header            X-Request-ID            $req_id;

            {{ if $externalAuth.Method }}
            proxy_method                {{ $externalAuth.Method }};
            proxy_set_header            X-Original-URI          
$request_uri;
            proxy_set_header            X-Scheme                
$pass_access_scheme;
            {{ end }}

            proxy_set_header            Host                    {{ 
$externalAuth.Host }};
            proxy_set_header            X-Original-URL          
$scheme://$http_host$request_uri;
            proxy_set_header            X-Original-Method       
$request_method;
            proxy_set_header            X-Sent-From             "nginx-
ingress-controller";
            proxy_set_header            X-Real-IP               
$remote_addr;
            {{ if and $all.Cfg.UseForwardedHeaders 
$all.Cfg.ComputeFullForwardedFor }}
            proxy_set_header            X-Forwarded-For         
$full_x_forwarded_for;
            {{ else }}
            proxy_set_header            X-Forwarded-For         
$remote_addr;
            {{ end }}

            {{ if $externalAuth.RequestRedirect }}
            proxy_set_header            X-Auth-Request-Redirect {{ 
$externalAuth.RequestRedirect }};
            {{ else }}
            proxy_set_header            X-Auth-Request-Redirect
$request_uri;
            {{ end }}

            {{ if $externalAuth.AuthCacheKey }}
            proxy_buffering                         "on";
            {{ else }}
            proxy_buffering                         {{
$location.Proxy.ProxyBuffering }};
            {{ end }}
            proxy_buffer_size                       {{
$location.Proxy.BufferSize }};
            proxy_buffers                           {{
$location.Proxy.BuffersNumber }} {{ $location.Proxy.BufferSize }};
            proxy_request_buffering                 {{
$location.Proxy.RequestBuffering }};
            proxy_http_version                      {{
$location.Proxy.ProxyHTTPVersion }};

            proxy_ssl_server_name       on;
            proxy_pass_request_headers  on;
            {{ if isValidByteSize $location.Proxy.BodySize true }}
            client_max_body_size        {{ $location.Proxy.BodySize }};
            {{ end }}
            {{ if isValidByteSize $location.ClientBodyBufferSize false }}
            client_body_buffer_size     {{ $location.ClientBodyBufferSize
}};
            {{ end }}

            # Pass the extracted client certificate to the auth provider
            {{ if not (empty $server.CertificateAuth.CAFileName) }}
            {{ if $server.CertificateAuth.PassCertToUpstream }}
            proxy_set_header ssl-client-cert
$ssl_client_escaped_cert;
            {{ end }}
            proxy_set_header ssl-client-verify      $ssl_client_verify;
            proxy_set_header ssl-client-subject-dn  $ssl_client_s_dn;
            proxy_set_header ssl-client-issuer-dn   $ssl_client_i_dn;
            {{ end }}

            {{- range $line := buildAuthProxySetHeaders
$externalAuth.ProxySetHeaders}}
            {{ $line }}
            {{- end }}

            {{ if not (empty $externalAuth.AuthSnippet) }}
            {{ $externalAuth.AuthSnippet }}
            {{ end }}

            set $target {{ $externalAuth.URL }};
            proxy_pass $target;
        }
        {{ end }}
        {{ if isLocationAllowed $location }}
        {{ if $externalAuth.SigninURL }}
        location {{ buildAuthSignURLLocation $location.Path
$externalAuth.SigninURL }} {
            internal;

            add_header Set-Cookie $auth_cookie;

            return 302 {{ buildAuthSignURL $externalAuth.SigninURL }};
        }
        {{ end }}
        {{ end }}

        location {{ $path }} {
            {{ $ing := (getIngressInformation $location.Ingress
$server.Hostname $location.Path) }}
            set $namespace      {{ $ing.Namespace | quote}};
            set $ingress_name   {{ $ing.Rule | quote }};
            set $service_name   {{ $ing.Service | quote }};
            set $service_port   {{ $ing.ServicePort | quote }};
            set $location_path  {{ $location.Path | escapeLiteralDollar |
quote }};

            {{ buildOpentracingForLocation $all.Cfg.EnableOpentracing
$location }}

            {{ if $location.Mirror.Source }}
            mirror {{ $location.Mirror.Source }};
            mirror_request_body {{ $location.Mirror.RequestBody }};
            {{ end }}

            rewrite_by_lua_block {
                lua_ingress.rewrite({{ locationConfigForLua $location $all
}})
                balancer.rewrite()
                plugins.run()
            }

            # be careful with `access_by_lua_block` and `satisfy any`
directives as satisfy any
            # will always succeed when there's `access_by_lua_block` that
does not have any lua code doing `ngx.exit(ngx.DECLINED)`
            # other authentication method such as basic auth or external
auth useless - all requests will be allowed.
            #access_by_lua_block {
            #}

            header_filter_by_lua_block {
                lua_ingress.header()
                plugins.run()
            }

            body_filter_by_lua_block {
            }
            log_by_lua_block {
                balancer.log()
                {{ if $all.EnableMetrics }}
                monitor.call()
                {{ end }}

                plugins.run()
            }

            {{ if not $location.Logs.Access }}
            access_log off;
            {{ end }}

            {{ if $location.Logs.Rewrite }}
            rewrite_log on;
            {{ end }}

            {{ if $location.HTTP2PushPreload }}
            http2_push_preload on;
            {{ end }}

            port_in_redirect {{ if $location.UsePortInRedirects }}on{{ else
}}off{{ end }};

            set $balancer_ewma_score -1;
            set $proxy_upstream_name {{ buildUpstreamName $location | quote
}};
            set $proxy_host          $proxy_upstream_name;
            set $pass_access_scheme  $scheme;

            {{ if $all.Cfg.UseProxyProtocol }}
            set $pass_server_port    $proxy_protocol_server_port;
            {{ else }}
            set $pass_server_port    $server_port;
            {{ end }}

            set $best_http_host      $http_host;
            set $pass_port           $pass_server_port;

            set $proxy_alternative_upstream_name "";

            {{ buildModSecurityForLocation $all.Cfg $location }}

            {{ if isLocationAllowed $location }}
            {{ if gt (len $location.Whitelist.CIDR) 0 }}
            {{ range $ip := $location.Whitelist.CIDR }}
            allow {{ $ip }};{{ end }}
            deny all;
            {{ end }}

            {{ if not (isLocationInLocationList $location
$all.Cfg.NoAuthLocations) }}
            {{ if $authPath }}
            # this location requires authentication
            auth_request        {{ $authPath }};
            auth_request_set    $auth_cookie $upstream_http_set_cookie;
            add_header          Set-Cookie $auth_cookie;
            {{- range $line := buildAuthResponseHeaders
$externalAuth.ResponseHeaders }}
            {{ $line }}
            {{- end }}
            {{ end }}

            {{ if $externalAuth.SigninURL }}
            set_escape_uri $escaped_request_uri $request_uri;
            error_page 401 = {{ buildAuthSignURLLocation $location.Path
$externalAuth.SigninURL }};
            {{ end }}

            {{ if $location.BasicDigestAuth.Secured }}
            {{ if eq $location.BasicDigestAuth.Type "basic" }}
            auth_basic {{ $location.BasicDigestAuth.Realm | quote }};
            auth_basic_user_file {{ $location.BasicDigestAuth.File }};
            {{ else }}
            auth_digest {{ $location.BasicDigestAuth.Realm | quote }};
            auth_digest_user_file {{ $location.BasicDigestAuth.File }};
            {{ end }}
            proxy_set_header Authorization "";
            {{ end }}
            {{ end }}

            {{/* if the location contains a rate limit annotation, create
one */}}
            {{ $limits := buildRateLimit $location }}
            {{ range $limit := $limits }}
            {{ $limit }}{{ end }}

            {{ if $location.CorsConfig.CorsEnabled }}
            {{ template "CORS" $location }}
            {{ end }}

            {{ buildInfluxDB $location.InfluxDB }}

            {{ if isValidByteSize $location.Proxy.BodySize true }}
            client_max_body_size                    {{
$location.Proxy.BodySize }};
            {{ end }}
            {{ if isValidByteSize $location.ClientBodyBufferSize false }}
            client_body_buffer_size                 {{
$location.ClientBodyBufferSize }};
            {{ end }}

            {{/* By default use vhost as Host to upstream, but allow
overrides */}}
            {{ if not (eq $proxySetHeader "grpc_set_header") }}
            {{ if not (empty $location.UpstreamVhost) }}
            {{ $proxySetHeader }} Host                   {{
```

```
$location.UpstreamVhost | quote }};
            {{ else }}
            {{ $proxySetHeader }} Host                    $best_http_host;
            {{ end }}
            {{ end }}

            # Pass the extracted client certificate to the backend
            {{ if not (empty $server.CertificateAuth.CAFileName) }}
            {{ if $server.CertificateAuth.PassCertToUpstream }}
            {{ $proxySetHeader }} ssl-client-cert
$ssl_client_escaped_cert;
            {{ end }}
            {{ $proxySetHeader }} ssl-client-verify
$ssl_client_verify;
            {{ $proxySetHeader }} ssl-client-subject-dn  $ssl_client_s_dn;
            {{ $proxySetHeader }} ssl-client-issuer-dn   $ssl_client_i_dn;
            {{ end }}

            # Allow websocket connections
            {{ $proxySetHeader }}                        Upgrade
$http_upgrade;
            {{ if $location.Connection.Enabled}}
            {{ $proxySetHeader }}                        Connection
{{ $location.Connection.Header }};
            {{ else }}
            {{ $proxySetHeader }}                        Connection
$connection_upgrade;
            {{ end }}

            {{ $proxySetHeader }} X-Request-ID           $req_id;
            {{ $proxySetHeader }} X-Real-IP              $remote_addr;
            {{ if and $all.Cfg.UseForwardedHeaders
$all.Cfg.ComputeFullForwardedFor }}
            {{ $proxySetHeader }} X-Forwarded-For
$full_x_forwarded_for;
            {{ else }}
            {{ $proxySetHeader }} X-Forwarded-For        $remote_addr;
            {{ end }}
            {{ $proxySetHeader }} X-Forwarded-Host       $best_http_host;
            {{ $proxySetHeader }} X-Forwarded-Port       $pass_port;
            {{ $proxySetHeader }} X-Forwarded-Proto
$pass_access_scheme;
            {{ if $all.Cfg.ProxyAddOriginalURIHeader }}
            {{ $proxySetHeader }} X-Original-URI         $request_uri;
            {{ end }}
            {{ $proxySetHeader }} X-Scheme
$pass_access_scheme;

            # Pass the original X-Forwarded-For
            {{ $proxySetHeader }} X-Original-Forwarded-For {{
buildForwardedFor $all.Cfg.ForwardedForHeader }};

            # mitigate HTTPoxy Vulnerability
            # https://www.nginx.com/blog/mitigating-the-httpoxy-
```

```
vulnerability-with-nginx/
            {{ $proxySetHeader }} Proxy                  "";

            # Custom headers to proxied server
            {{ range $k, $v := $all.ProxySetHeaders }}
            {{ $proxySetHeader }} {{ $k }}                    {{ $v | quote
}};
            {{ end }}

            proxy_connect_timeout                   {{
$location.Proxy.ConnectTimeout }}s;
            proxy_send_timeout                      {{
$location.Proxy.SendTimeout }}s;
            proxy_read_timeout                      {{
$location.Proxy.ReadTimeout }}s;

            proxy_buffering                         {{
$location.Proxy.ProxyBuffering }};
            proxy_buffer_size                       {{
$location.Proxy.BufferSize }};
            proxy_buffers                           {{
$location.Proxy.BuffersNumber }} {{ $location.Proxy.BufferSize }};
            {{ if isValidByteSize $location.Proxy.ProxyMaxTempFileSize true
}}
            proxy_max_temp_file_size                {{
$location.Proxy.ProxyMaxTempFileSize }};
            {{ end }}
            proxy_request_buffering                 {{
$location.Proxy.RequestBuffering }};
            proxy_http_version                      {{
$location.Proxy.ProxyHTTPVersion }};

            proxy_cookie_domain                     {{
$location.Proxy.CookieDomain }};
            proxy_cookie_path                       {{
$location.Proxy.CookiePath }};

            # In case of errors try the next upstream server before
returning an error
            proxy_next_upstream                     {{ buildNextUpstream
$location.Proxy.NextUpstream $all.Cfg.RetryNonIdempotent }};
            proxy_next_upstream_timeout             {{
$location.Proxy.NextUpstreamTimeout }};
            proxy_next_upstream_tries               {{
$location.Proxy.NextUpstreamTries }};

            {{/* Add any additional configuration defined */}}
            {{ $location.ConfigurationSnippet }}

            {{ if not (empty $all.Cfg.LocationSnippet) }}
            # Custom code snippet configured in the configuration configmap
            {{ $all.Cfg.LocationSnippet }}
            {{ end }}
            {{/* if we are sending the request to a custom default backend,
we add the required headers */}}
            {{ if (hasPrefix $location.Backend "custom-default-backend-")
}}
            proxy_set_header       X-Code             503;
            proxy_set_header       X-Format           $http_accept;
            proxy_set_header       X-Namespace        $namespace;
            proxy_set_header       X-Ingress-Name     $ingress_name;
            proxy_set_header       X-Service-Name     $service_name;
            proxy_set_header       X-Service-Port     $service_port;
            proxy_set_header       X-Request-ID       $req_id;
            {{ end }}

            {{ if $location.Satisfy }}
            satisfy {{ $location.Satisfy }};
            {{ end }}

            {{/* if a location-specific error override is set, add the
proxy_intercept here */}}
            {{ if $location.CustomHTTPErrors }}
            # Custom error pages per ingress
            proxy_intercept_errors on;
            {{ end }}

            {{ range $errCode := $location.CustomHTTPErrors }}
            error_page {{ $errCode }} = @custom_{{
$location.DefaultBackendUpstreamName }}_{{ $errCode }};{{ end }}

            {{ if (eq $location.BackendProtocol "FCGI") }}
            include /etc/nginx/fastcgi_params;
            {{ end }}
            {{- if $location.FastCGI.Index -}}
            fastcgi_index {{ $location.FastCGI.Index | quote }};
            {{- end -}}
            {{ range $k, $v := $location.FastCGI.Params }}
            fastcgi_param {{ $k }} {{ $v | quote }};
            {{ end }}

            {{ if not (empty $location.Redirect.URL) }}
            return {{ $location.Redirect.Code }} {{ $location.Redirect.URL
}};
            {{ end }}

            {{ buildProxyPass $server.Hostname $all.Backends $location }}
            {{ if (or (eq $location.Proxy.ProxyRedirectFrom "default") (eq
$location.Proxy.ProxyRedirectFrom "off")) }}
            proxy_redirect                          {{
$location.Proxy.ProxyRedirectFrom }};
            {{ else if not (eq $location.Proxy.ProxyRedirectTo "off") }}
            proxy_redirect                          {{
$location.Proxy.ProxyRedirectFrom }} {{ $location.Proxy.ProxyRedirectTo }};
            {{ end }}
            {{ else }}
            # Location denied. Reason: {{ $location.Denied | quote }}
            return 503;
            {{ end }}
            {{ if not (empty $location.ProxySSL.CAFileName) }}
            # PEM sha: {{ $location.ProxySSL.CASHA }}
            proxy_ssl_trusted_certificate           {{
$location.ProxySSL.CAFileName }};
            proxy_ssl_ciphers                       {{
$location.ProxySSL.Ciphers }};
            proxy_ssl_protocols                     {{
$location.ProxySSL.Protocols }};
            proxy_ssl_verify                        {{
$location.ProxySSL.Verify }};
            proxy_ssl_verify_depth                  {{
$location.ProxySSL.VerifyDepth }};
            {{ end }}

            {{ if not (empty $location.ProxySSL.ProxySSLName) }}
            proxy_ssl_name                          {{
$location.ProxySSL.ProxySSLName }};
            {{ end }}
            {{ if not (empty $location.ProxySSL.ProxySSLServerName) }}
            proxy_ssl_server_name                   {{
$location.ProxySSL.ProxySSLServerName }};
            {{ end }}

            {{ if not (empty $location.ProxySSL.PemFileName) }}
            proxy_ssl_certificate                   {{
$location.ProxySSL.PemFileName }};
            proxy_ssl_certificate_key               {{
$location.ProxySSL.PemFileName }};
            {{ end }}
        }
        {{ end }}
        {{ end }}

        {{ if eq $server.Hostname "_" }}
        # health checks in cloud providers require the use of port {{
$all.ListenPorts.HTTP }}
        location {{ $all.HealthzURI }} {
            {{ if $all.Cfg.EnableOpentracing }}
            opentracing off;
            {{ end }}

            access_log off;
            return 200;
        }

        # this is required to avoid error if nginx is being monitored
        # with an external software (like sysdig)
        location /nginx_status {
            {{ if $all.Cfg.EnableOpentracing }}
            opentracing off;
            {{ end }}
            {{ range $v := $all.NginxStatusIpv4Whitelist }}
            allow {{ $v }};
            {{ end }}
            {{ if $all.IsIPV6Enabled -}}
            {{ range $v := $all.NginxStatusIpv6Whitelist }}
            allow {{ $v }};
            {{ end }}
            {{ end -}}
            deny all;

            access_log off;
            stub_status on;
        }

        {{ end }}

{{ end }}
```

> 涉及的lua部分的一些配置可自行了解 https://github.com/kubernetes/ingress-nginx/blob/master/rootfs/etc/nginx/lua/