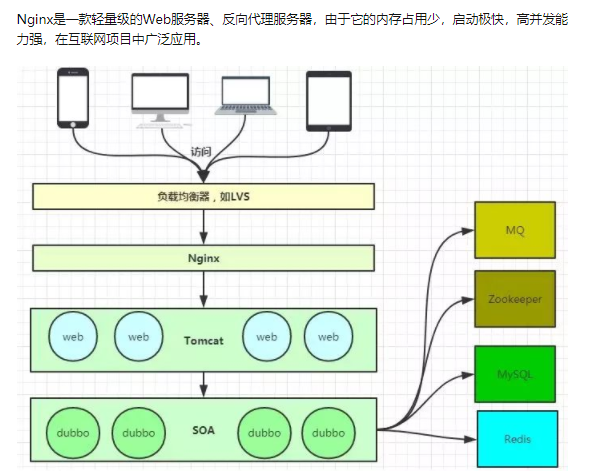
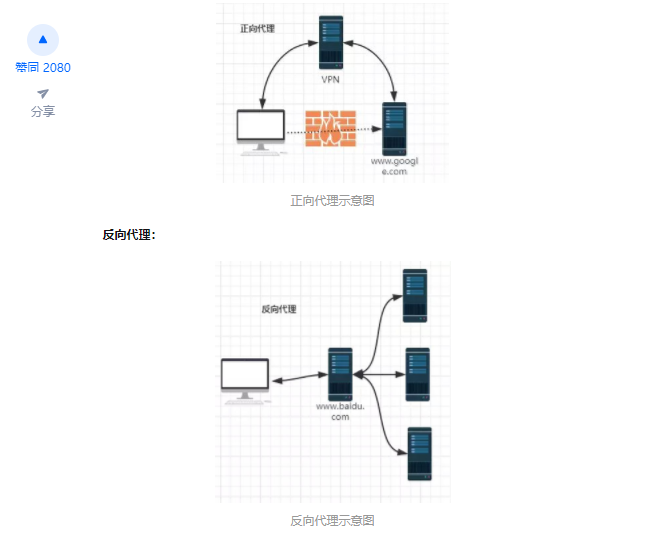
nginx介绍

# 一、nginx简介

## 1、nginx简介



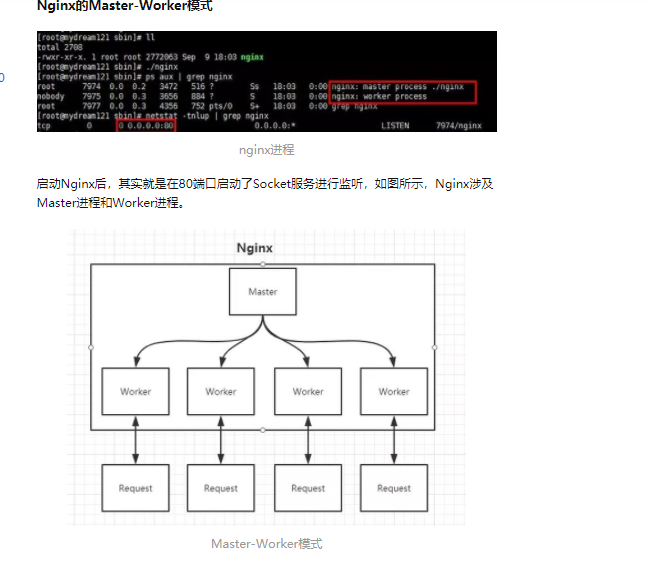
## 2、反向代理：



由于防火墙的原因，我们并不能直接访问谷歌，那么我们可以借助VPN来实现，这就是一个简单的正向代理的例子。这里你能够发现，正向代理“代理”的是客户端，而且客户端是知道目标的，而目标是不知道客户端是通过VPN访问的。

当我们在外网访问百度的时候，其实会进行一个转发，代理到内网去，这就是所谓的反向代理，即反向代理“代理”的是服务器端，而且这一个过程对于客户端而言是透明的。

## 3、nginx的master-worker模式



Master进程的作用是？

**读取并验证配置文件nginx.conf；管理worker进程；**

Worker进程的作用是？

**每一个Worker进程都维护一个线程（避免线程切换），处理连接和请求；注意Worker进程的个数由配置文件决定，一般和CPU个数相关（有利于进程切换），配置几个就有几个Worker进程。**

## 4、思考：Nginx如何做到热部署？

所谓热部署，就是配置文件nginx.conf修改后，不需要stop Nginx，不需要中断请求，就能让配置文件生效！（nginx -s reload 重新加载/nginx -t检查配置/nginx -s stop）

通过上文我们已经知道worker进程负责处理具体的请求，那么如果想达到热部署的效果，可以想象：

方案一：

修改配置文件nginx.conf后，主进程master负责推送给woker进程更新配置信息，woker进程收到信息后，更新进程内部的线程信息。（有点valatile的味道）

方案二：

修改配置文件nginx.conf后，重新生成新的worker进程，当然会以新的配置进行处理请求，而且新的请求必须都交给新的worker进程，至于老的worker进程，等把那些以前的请求处理完毕后，kill掉即可。

Nginx采用的就是方案二来达到热部署的！

## 5、思考：Nginx如何做到高并发下的高效处理？

上文已经提及Nginx的worker进程个数与CPU绑定、worker进程内部包含一个线程高效回环处理请求，这的确有助于效率，但这是不够的。

**作为专业的程序员，我们可以开一下脑洞：BIO/NIO/AIO、异步/同步、阻塞/非阻塞...**

要同时处理那么多的请求，要知道，有的请求需要发生IO，可能需要很长时间，如果等着它，就会拖慢worker的处理速度。

**Nginx采用了Linux的epoll模型，epoll模型基于事件驱动机制，它可以监控多个事件是否准备完毕，如果OK，那么放入epoll队列中，这个过程是异步的。worker只需要从epoll队列循环处理即可。**

## 6、思考：Nginx挂了怎么办？

Nginx既然作为入口网关，很重要，如果出现单点问题，显然是不可接受的。

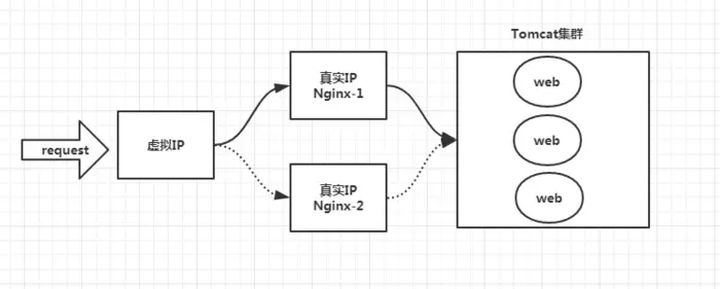
答案是：**Keepalived+Nginx实现高可用**。

Keepalived是一个高可用解决方案，主要是用来防止服务器单点发生故障，可以通过和Nginx配合来实现Web服务的高可用。（其实，Keepalived不仅仅可以和Nginx配合，还可以和很多其他服务配合）

Keepalived+Nginx实现高可用的思路：

第一：请求不要直接打到Nginx上，应该先通过Keepalived（这就是所谓虚拟IP，VIP）

第二：Keepalived应该能监控Nginx的生命状态（提供一个用户自定义的脚本，定期检查Nginx进程状态，进行权重变化,，从而实现Nginx故障切换）

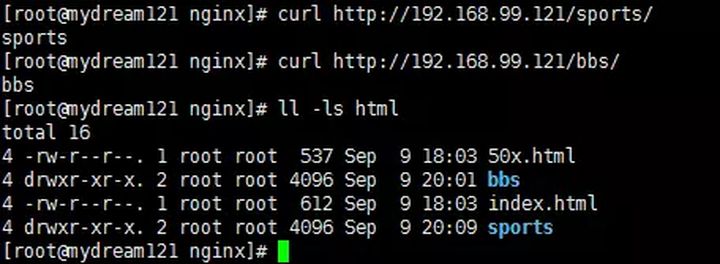
Keepalived+Nginx

## 7、我们的主战场：nginx.conf

很多时候，在开发、测试环境下，我们都得自己去配置Nginx，就是去配置nginx.conf。

nginx.conf是典型的分段配置文件，下面我们来分析下。

### 1）虚拟主机

http的server段访问结果

其实这是把Nginx作为web server来处理静态资源。

第一：location可以进行正则匹配，应该注意正则的几种形式以及优先级。（这里不展开）

第二：Nginx能够提高速度的其中一个特性就是：动静分离，就是把静态资源放到Nginx上，由Nginx管理，动态请求转发给后端。

第三：我们可以在Nginx下把静态资源、日志文件归属到不同域名下（也即是目录），这样方便管理维护。

第四：Nginx可以进行IP访问控制，有些电商平台，就可以在Nginx这一层，做一下处理，内置一个黑名单模块，那么就不必等请求通过Nginx达到后端在进行拦截，而是直接在Nginx这一层就处理掉。

### 2）反向代理【proxy\_pass】

所谓反向代理，很简单，其实就是在location这一段配置中的root替换成**proxy\_pass**即可。root说明是静态资源，可以由Nginx进行返回；而proxy\_pass说明是动态请求，需要进行转发，比如代理到Tomcat上。

反向代理，上面已经说了，过程是透明的，比如说request -> Nginx -> Tomcat，那么对于Tomcat而言，请求的IP地址就是Nginx的地址，而非真实的request地址，这一点需要注意。不过好在Nginx不仅仅可以反向代理请求，还可以由用户**自定义设置HTTP HEADER**。

### 3）负载均衡【upstream】

上面的反向代理中，我们通过proxy\_pass来指定Tomcat的地址，很显然我们只能指定一台Tomcat地址，那么我们如果想指定多台来达到负载均衡呢？

第一，通过**upstream**来定义一组Tomcat，并指定负载策略（IPHASH、加权论调、最少连接），健康检查策略（Nginx可以监控这一组Tomcat的状态）等。

第二，将proxy\_pass替换成upstream指定的值即可。

**负载均衡可能带来的问题？**

负载均衡所带来的明显的问题是，一个请求，可以到A server，也可以到B server，这完全不受我们的控制，当然这也不是什么问题，只是我们得注意的是：**用户状态的保存问题，如Session会话信息，不能在保存到服务器上。**

**负载均衡的方式：**

**轮询：就是将请求挨个发送到每个服务器上。**

**权重：就是按照权重值来向服务器转发请求，不同的权重值分发的请求的数量不同。**

### 4）缓存

缓存，是Nginx提供的，可以加快访问速度的机制，说白了，在配置上就是一个开启，同时指定目录，让缓存可以存储到磁盘上。具体配置，大家可以参考Nginx官方文档，这里就不在展开了。

## 8、[Nginx详解](https://link.zhihu.com/?target=https%3A//mp.weixin.qq.com/s/XoqGvYBabW8YBl9xEeNYZw)：

https://mp.weixin.qq.com/s/XoqGvYBabW8YBl9xEeNYZw

## 9、[一文详解负载均衡和反向代理的真实区别](https://link.zhihu.com/?target=https%3A//mp.weixin.qq.com/s/TYM83F2O-keMvn4ZYa5nqw)

https://mp.weixin.qq.com/s/TYM83F2O-keMvn4ZYa5nqw

# 二、linux安装nginx

## 1、安装gcc--安装在根目录下即可

yum install gcc-c++

如果报错：This system is not registered to Red Hat Subscription Management. You can use subscription-manager to register.

是因为我们没有配置yum仓库。

配置仓库的方法：

1..进入到/etc/yum.repos.d/目录（该目录存放着Yum软件仓库的配置文件）

2.使用vim编辑器创建一个名为rhe17.repo的新配置文件（文件名称可随便，但后缀必须为.repo）

root@2022 ~]# cd /etc/yum.repos.d/

[root@2022 yum.repos.d]# vim rhe17.repo

[rhe17]

name=rhe17

baseurl=file:///media/cdrom

enabled=1

gpgcheck=0

保存退出

3.把光盘设备的系统镜像挂载到/media/cdrom目录  
[root@2022~]*# mkdir -p /media/cdrom*

[root@2022 ~]*# mount /dev/cdrom /media/cdrom*

然后在安装就可以了。

## 2、PCRE pcre-devel 安装

PCRE(Perl Compatible Regular Expressions) 是一个Perl库，包括 perl 兼容的正则表达式库。nginx 的 http 模块使用 pcre 来解析正则表达式，所以需要在 linux 上安装 pcre 库，pcre-devel 是使用 pcre 开发的一个二次开发库。nginx也需要此库。命令：

yum install -y pcre pcre-devel

## 3、zlib 安装

zlib 库提供了很多种压缩和解压缩的方式， nginx 使用 zlib 对 http 包的内容进行 gzip ，所以需要在 Centos 上安装 zlib 库。

yum install -y zlib zlib-devel

## 4、OpenSSL 安装

OpenSSL 是一个强大的安全套接字层密码库，囊括主要的密码算法、常用的密钥和证书封装管理功能及 SSL 协议，并提供丰富的应用程序供测试或其它目的使用。

nginx 不仅支持 http 协议，还支持 https（即在ssl协议上传输http），所以需要在 Centos 安装 OpenSSL 库。

yum install -y openssl openssl-devel

## 5.下载Nginx

wget https://nginx.org/download/nginx-1.19.9.tar.gz

## 6.解压nginx

tar -zxvf nginx-1.19.9.tar.gz

cd nginx-1.19.9

## 7.执行nginx-configure文件

./configure

## 8.make命令编译

执行完后会有一个MakeFile文件夹

make 是一个命令工具，它解释 Makefile 中的指令（应该说是规则）。在 Makefile文件中描述了整个工程所有文件的编译顺序、编译规则

注：需要先安装make包，否则会报没有make命令的错误，可先检查有没有make包 make -v

yum -y install gcc automake autoconf libtool make

make

make install

## 9.查询nginx 安装目录

whereis nginx

## 10.进入安装目录执行nginx

前往安装目录找到sbin 执行nginx

./nginx

# 三、安装启动过程问题及解决方案~

## 1、yum install 404解决方案

这个解决方案是针对centOS的，如果是redHAT8的话可以参考(二、1.)的解决方案。

1）、进入配置文件内，删除所有的.repo文件（也可以备份）

#进入配置文件夹

cd /etc/yum.repos.d/

#删除旧的配置文件

rm \*.repo

#输入“y”回车确认

ls确保该目录下的.repo文件已完全删除

下载可以用的.repo文件

wget -O /etc/yum.repos.d/CentOS-Base.repo https://mirrors.aliyun.com/repo/Centos-vault-8.5.2111.repo

如果你没有安装wget，也可以用下面命令：

curl -o /etc/yum.repos.d/CentOS-Base.repo https://mirrors.aliyun.com/repo/Centos-vault-8.5.2111.rep

## 2、如何修改nginx监听的端口号？

查找nginx.conf文件的位置：whereis nginx.conf

nginx: /usr/local/nginx

进入到对应的目录

cd /usr/local/nginx

cd conf

vim nginx.conf

将server中的 listen后面的参数修改为想要监听的端口

重新加载文件

source nginx.conf

然后重新启动nginx(见上文)

同样在防火墙中需要将3000端口打开，就可以在浏览器中查看nginx的初始界面了。

# 四、nginx常用命令

## 1、配置环境变量

whereis nginx 找到nginx 安装目录

前往根目录 找到etc文件夹

vim profile 打开配置文件

添加环境变量（前半段node忽略）

export PATH=$PATH:/node-v14.19.1-linux-x64/bin:/usr/local/nginx/sbin

重新执行profile文件

## 2.Nginx常用命令介绍

### 1）查看nginx版本号

nginx -v

nginx -V

### 2）启动nginx

直接执行 nginx 即可

### 3）停止nginx

nginx -s stop

Stop 是立即停止

nginx -s quit

Quit 是一个优雅的关闭方式，Nginx在退出前完成已经接受的请求处理

### 4）重载nginx配置文件

nginx -s reload，改了配置文件不需要重新启动nginx服务

### 5）查看nginx进程

ps -ef | grep nginx

ps -ef

输出标准格式的linux进程命令

grep nginx

grep命令 是查找， 是一种强大的文本搜索工具 我们这儿是查找nginx

### 6）检查nginx配置文件是否有语法错误

nginx -t

# 五、nginx配置文件

## 1、配置文件模块

Nginx的主配置文件是nginx.conf，这个配置文件一共由三部分组成，分别为全局块、events块和http块。

在http块中，又包含http全局块、多个server块。

每个server块中，可以包含server全局块和多个location块。在同一配置块中嵌套的配置块，各个之间不存在次序关系

## 2、全局块

全局块是默认配置文件从开始到events块之间的一部分内容，主要设置一些影响Nginx服务器整体运行的配置指令，因此，这些指令的作用域是Nginx服务器全局。

user [user] [group] 指定可以运行nginx服务的用户和用户组，只能在全局块配置 user指令在Windows上不生效，如果你制定具体用户和用户组会报警告

worker\_processes nginx进程数量worker\_processes 比如设置为2，nginx将会开启一个master进程和2两个worker进程

pid logs/nginx.pid 存放pid文件

error\_log logs/error.log; 全局错误日志类型 debug info warn error 存放地址

## 3.events块

events块涉及的指令主要影响Nginx服务器与用户的网络连接。常用到的设置包括是否开启对多worker process下的网络连接进行序列化，是否允许同时接收多个网络连接，选取哪种事件驱动模型处理连接请求，每个worker process可以同时支持的最大连接数等

accept\_mutex 默认开启-开启之后nginx 的多个worker将会以串行的方式来处理，只会有一个worker将会被唤起，其他的worker继续睡眠，如果不开启将会造成惊群效应多个worker全部唤起不过只有一个Worker能获取新连接，其它的Worker会重新进入休眠状态

worker\_connections 单个进程最大连接数（最大连接数=连接数+进程数）

## 4.http块

http块是Nginx服务器配置中的重要部分，代理、缓存和日志定义等绝大多数的功能和第三方模块的配置都可以放在这个模块中。

**include指令**，用于引入其他的配置文件

**default\_type** 如果Web程序没设置，Nginx也没对应文件的扩展名，就用Nginx 里默认的 default\_type定义的处理方式。default\_type application/octet-stream; #nginx默认文件类型

**log\_forma**t指令，用于定义日志格式，此指令只能在http块中进行配置

**sendfile** 简单来说就是启用sendfile()系统调用来替换read()和write()调用，减少系统上下文切换从而提高性能，当 nginx 是静态文件服务器时，能极大提高nginx的性能表现

**keepalive\_timeout** HTTP 有一个 KeepAlive 模式，它告诉 webserver 在处理完一个请求后保持这个 TCP 连接的打开状态。若接收到来自客户端的其它请求，服务端会利用这个未被关闭的连接，而不需要再建立一个连接。

gzip 开启Gzip压缩功能， 可以使网站的css、js 、xml、html 文件在传输时进行压缩，提高访问速度, 进而优化Nginx性能

## 5.server块

每一个http块都可以包含多个server块，而每个server块就相当于一台虚拟主机，它内部可有多台主机联合提供服务，一起对外提供在逻辑上关系密切的一组服务

**listen**指令的配置非常灵活，可以单独制定ip，单独指定端口或者同时指定ip和端口

listen 127.0.0.1:8000; #只监听来自127.0.0.1这个IP，请求8000端口的请求

listen 127.0.0.1; #只监听来自127.0.0.1这个IP，请求80端口的请求（不指定端口，默认80）

listen 9999; #监听来自所有IP，请求9999端口的请求

listen \*:9999; #和上面效果一样

listen localhost:8000; #和第一种效果一致

**server\_name** nginx 允许一个虚拟主机有一个或多个名字，也可以使用通配符"\*"来设置虚拟主机的名字 支持 ip 域名 通配符 正则等

server\_name localhost;

## 6.location块

每个server块中可以包含多个location块。在整个Nginx配置文档中起着重要的作用，而且Nginx服务器在许多功能上的灵活性往往在location指令的配置中体现出来

location 指令可以分为以下 3 类：

前缀字符串匹配

正则表达式匹配

用于内部跳转的命名location

前缀字符串匹配

精确匹配 =

前缀匹配 ^~（立刻停止后续的正则搜索）

按文件中顺序的正则匹配 ~或~\*

匹配不带任何修饰的前缀匹配。

location root

root 指定目录的上级目录，并且该上级目录要含有locatoin指定名称的同名目录。

location /img/ {

root /var/www/image;

}

若按照这种配置的话，则访问/img/目录下的文件时，nginx会去/var/www/image/img/目录下找文件

# 六、nginx反向代理

## 1、Nginx反向代理的配置语法

反向代理中的常用指令：

proxy\_pass

proxy\_set\_header

proxy\_pass

该指令用来设置被代理服务器地址，可以是主机名称、IP地址加端口号形式。

**案例1代理到哔哩哔哩**

location / {

root html;

index index.html index.htm;

proxy\_pass http://bilibili.com;

}

访问/就会被转到哔哩哔哩

案例2 nginx反向代理解决跨域

前端代码

a.onclick = () => {

let xhr = new XMLHttpRequest()

xhr.open('GET','/api/portal/list')

xhr.onreadystatechange = () => {

if(xhr.readyState == 4 && xhr.status == 200){

console.log(xhr.responseText);

}

}

xhr.send(null)

}

express 服务端代码

const express = require('express')

const app = express()

app.get('/portal/list', (req, res) => {

res.json({

code: 200,

message: "搞咩啊"

})

})

app.listen(9000,()=>{

console.log('success');

})

nginx 配置文件

location /api/ {

proxy\_pass http://localhost:9000/;

}

截取到/api/ 将会转发到 http://localhost:9000/

proxy\_set\_header

该指令可以更改Nginx服务器接收到的客户端请求的请求头信息，然后将新的请求头发送给代理的服务器

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Real-Port $remote\_port;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

三个header分别表示：

X-Real-IP 客户端或上一级代理ip

X-Real-Port 客户端或上一级端口

X-Forwarded-For 包含了客户端和各级代理ip的完整ip链路

其中X-Real-IP是必需的，后两项选填。当只存在一级nginx代理的时候X-Real-IP和X-Forwarded-For是一致的，而当存在多级代理的时候，X-Forwarded-For 就变成了如下形式

$remote\_addr是前一节点的IP，并不一定是用户的真实IP。

# 七、Vue histroy 模式 跳转路由404 问题

## 1、解决方法

原因是：vue是单页面应用，刷新界面后，nginx的配置文件中没有配置其他的vue-router配置的路由，解决方式就是将其他的路径重定向到根路径

在nginx.conf文件中。

location / {

root html;

index index.html index.htm;

try\_files $uri $uri/ /index.html;

}

## 2、常见的参数

$uri 当前请求的 URI，但不含“？”后的参数

$args 当前请求的参数，即“？”后的宇符串

$arg\_xxx 当前请求里的某个参数，“arg ”后是参数的名字

$http\_xxx 当前请求里的 xxx 头部对应的值

$sent\_http\_xxx 返回给客户端的响应头部对应的值

$remote\_addr 客户端IP 地址。

$http\_cookie 获取cookie值

$cookie\_xxx 当前请求的cookie xxx对应的值

$request\_uri 浏览器发起的不作任何修改的请求的url中的path 如在www.baidu.com/p1/file?d=111, 其值为/p1/file?d=111

$uri 指当前的请求URI，不包括任何参数，反映任何内部重定向或index模块所做的修改

$request\_method 请求方法

# 八、go-access nginx日志分析器

## 1、简介

GoAccess是一款开源、实时，运行在命令行终端下的web日志分析工具。该工具提供快速、多样的HTTP状态统计，可以令管理员不再纠结于统计各类数据，和繁杂的指令以及一大堆管道/正则表达式说byebye

## 安装

在安装之前需要安装依赖包

[root@ljcccc ~]# yum install ncurses-devel

[root@ljcccc ~]# wget https://github.com/maxmind/geoip-api-c/releases/download/v1.6.11/GeoIP-1.6.11.tar.gz

[root@ljcccc ~]# tar GeoIP-1.6.11.tar.gz

[root@ljcccc ~]# cd GeoIP-1.6.11

[root@ljcccc ~]# ./configure && make && make install

1、wget http://tar.goaccess.io/goaccess-1.2.tar.gz

1. tar -xzvf goaccess-1.2.tar.gz
2. cd goaccess-1.2/
3. ./configure
4. make
5. make install

## 开启实时HTML报告分析（webSocket）

需要先在nginx的html文件夹下创建report.html

whereis ngnix

cd /usr/local/nginx

cd html

echo>report.html

再返回logs文件执行下面的命令

goaccess access.log -a -o ../html/report.html --real-time-html --log-format=COMBINED

## 4、选项

下面的选项可以通过命令行使用，如果是长选项则通过配置文件。

日志/日期/时间 格式

--time-format <timeformat>

参数 time-format 后跟随一个空格符，指定日志的时间格式，包含普通字符与特殊格式说明符的任意组合。他们都由百分号 (%)开始。参考 `man strftime`。 %T 或者 %H:%M:%S.

--date-format <dateformat>

参数 date-format 后跟随一个空格符，指定日志的日期格式，包含普通字符与特殊格式说明符的任意组合。他们都由百分号 (%)开始。参考 `man strftime`。

--log-format <logformat>

参数 log-format 后跟随一个空格符或者制表分隔符(\t)，用于指定日志字符串格式。

另外可以指定原始 日志/日期/时间 格式，简单来说，下表中的预定义日志格式名称可以用作 日志/日期/时间 格式的变量。GoAccess 可以处理在一个变量中处理一个预定义名称，而在另一变量中处理另外一个预定义名称。

COMBINED | 联合日志格式

VCOMBINED | 支持虚拟主机的联合日志格式

COMMON | 通用日志格式

VCOMMON | 支持虚拟主机的通用日志格式

W3C | W3C 扩展日志格式

SQUID | Native Squid 日志格式

CLOUDFRONT | 亚马逊 CloudFront Web 分布式系统

CLOUDSTORAGE | 谷歌云存储

AWSELB | 亚马逊弹性负载均衡

AWSS3 | 亚马逊简单存储服务 (S3)

注意: 一般来说，需要在包含空格、逗号、管道符、引号、/、括号的值的周围引用引号。内部引号必须进行转义处理。

注意: 使用管道传送数据给 GoAccess 时不会提示 日志/日期/时间 配置对话框，你需要在配置文件或者命令行中提前定义。

## 5、用户交互选项

请查阅配置文件中颜色方案示例。

-c --config-dialog

在程序开始运行时显示 日志/日期 配置窗口。

-i --hl-header

颜色高亮活动面板。

-m --with-mouse

在主仪表盘面板使能鼠标支持。

--color=<fg:bg[attrs, PANEL>

使用终端输出时指定自定义颜色。

## 6、颜色语法：

DEFINITION space/tab colorFG#:colorBG# [attributes,PANEL]

FG# = 前景色 [-1...255] (-1 = 默认配色)

BG# = 背景色 [-1...255] (-1 = 默认配色)

使用如下方式应用颜色属性也是允许的(多个属性使用逗号分隔)，例如： bold,underline,normal,reverse,blink

如果喜欢，可以为同一个指标在不同面板设置不同颜色，比如一个指标在页面请求面板使用颜色A，同时在浏览器面板则显示颜色B。

COLOR\_MTRC\_HITS

COLOR\_MTRC\_VISITORS

COLOR\_MTRC\_DATA

COLOR\_MTRC\_BW

COLOR\_MTRC\_AVGTS

COLOR\_MTRC\_CUMTS

COLOR\_MTRC\_MAXTS

COLOR\_MTRC\_PROT

COLOR\_MTRC\_MTHD

COLOR\_MTRC\_HITS\_PERC

COLOR\_MTRC\_HITS\_PERC\_MAX

COLOR\_MTRC\_VISITORS\_PERC

COLOR\_MTRC\_VISITORS\_PERC\_MAX

COLOR\_PANEL\_COLS

COLOR\_BARS

COLOR\_ERROR

COLOR\_SELECTED

COLOR\_PANEL\_ACTIVE

COLOR\_PANEL\_HEADER

COLOR\_PANEL\_DESC

COLOR\_OVERALL\_LBLS

COLOR\_OVERALL\_VALS

COLOR\_OVERALL\_PATH

COLOR\_ACTIVE\_LABEL

COLOR\_BG

COLOR\_DEFAULT

COLOR\_PROGRESS

--color-scheme <1|2|3>

选择终端配色方案。1 为单色方案。2 为绿色方案以及 3 为 Monokai 方案(需终端支持 256 色)。

--crawlers-only

仅解析和显示爬虫(机器人)。

--html-custom-css=<path.css>

在 HTML 报告中按照指定的自定义 CSS 文件路径加载 CSS 样式。

--html-custom-js=<path.js>

在 HTML 报告中按照指定的自定义 JS 文件路径加载 JS 代码。

--html-report-title=<title>

设定 HTML 报告页面的标题和头部内容。

--html-prefs=<JSON>

设定 HTML 报告的默认参数。通过提交一个有效的包含相关参数的 JSON 对象来设置。允许用户为每一个面板单独设置。参考如下示例。

--html-prefs='{"theme":"bright","perPage":5,"layout":"horizontal","showTables":true,"visitors":{"plot":{"chartType":"bar"}}}'

注意: 提交的 JSON 对象必须保存在一行 JSON 字符串中。

--json-pretty-print

使用制表符和新行格式化 JSON 输出。

--max-items=<num>

设置每个面板最多可以显示的单元个数。取值范围是 1 到 n。

注意: 仅 CSV 和 JSON 格式的报告允许大于默认值 366 个单元每面板(或者 50 个在实时 HTML 报告中)。

--no-color

关闭颜色输出。此选项在不支持色彩的终端上为默认选项。

--no-column-names

在终端输出中不显示列名。默认在每一个面板的每一个有效指标都会显示列名。

--no-csv-summary

在 CSV 格式输出中禁止汇总指标。

--no-progress

解析日志时不显示进度指标[总请求数/每秒请求数]。

--no-tab-scroll

禁止通过 TAB 键滚动面板或者使用数字键选择面板。

--no-html-last-updated

在生成的 HTML 报告中不显示“最近更新”。

## 7、服务器选项

--addr=<address>

将服务器绑定到指定 IP 地址。默认绑定到 0.0.0.0 。

通常无需指定，除非您希望将服务器绑定到主机上的其他地址。

--daemonize

使 GoAccess 作为守护程序运行(仅在 --real-time-html 开启下有效)。

--origin=<url>

E在 WebSocket 握手中确保客户端发送指定的源头。且指定的源应与浏览器发送源头字段完全相同。例如：--origin=http://goaccess.io

--port=<port>

指定服务使用的端口。GoAccess 默认使用端口 7890 作为 WebSocket 服务器。请确保此端口可用。

--real-time-html

使能实时 HTML 报告。

--ws-url=<[scheme://]url[:port]>

此 URL 用于 WebSocket 服务器的回应。用于客户端侧的 WebSocket 构建器。

同时可以选择指定 WebSocket 的 URI 协议，比如：ws:// 用于非加密连接, 以及 wss:// 用于加密连接。示例：wss://goaccess.io

如果 GoAccess 运行在代理服务器的后面，您需要通过在主机名后跟随冒号加端口号的方式让客户端连接到另外一个不同的端口。示例：goaccess.io:9999

默认情况下，会尝试去连接生成报告的主机名。如果 GoAccess 运行在一台远程服务器上，则远程主机名也应该在 URL 中指定。当然，必须保证主机是有效的。

--fifo-in=<path/file>

创建一个管道(先入先出)从指定的路径/文件读取数据。

--fifo-out=<path/file>

创建一个管道(先入先出)往指定的路径/文件写入数据。

--ssl-cert=<path/cert.crt>

指定 TLS/SSL 证书的路径。使 GoAccess 支持 TLS/SSL，需要使用参数 --ssl-cert 和 --ssl-key。

仅在使用了参数 --with-openssl 时有效

--ssl-key=<path/priv.key>

指定 TLS/SSL 私钥的路径。使 GoAccess 支持 TLS/SSL，需要使用参数 --ssl-cert 和 --ssl-key。

仅在使用了参数 --with-openssl 时有效

FILE OPTIONS

-f --log-file=<logfile>

指定输入日志文件的路径。如果在配置文件中指定了输入文件，则其优先级要高于在命令行中通过 -f 参数指定。

-l --log-debug=<filename>

发送所有调试信息到指定文件。需要指定配置选项 --enable-debug

-p --config-file=<configfile>

指定使用自定义配置文件。如果设置了此参数，其优先级将高于全局配置文件(如果有)。

--invalid-requests=<filename>

记录无效请求到指定文件。

--no-global-config

禁止加载全局配置文件。可能的目录应该是 /usr/etc/, /etc/ 或者 /usr/local/etc/, 除非在运行 ./configure 时指定了 --sysconfdir=/dir 。

## 8、解析选项

-a --agent-list

开启 UserAgent 列表。开启后会降低解析速度。

-d --with-output-resolver

输出 HTML 或者 JSON 报告时开启 IP 解析。

-e --exclude-ip <IP|IP-range>

排除一个 IPv4 或者 IPv6 地址。 使用连接符表示 IP 段(开始-结束)。

exclude-ip 127.0.0.1

exclude-ip 192.168.0.1-192.168.0.100

exclude-ip ::1

exclude-ip 0:0:0:0:0:ffff:808:804-0:0:0:0:0:ffff:808:808

-H --http-protocol=<yes|no>

HTTP 请求协议开关。将创建一个请求字段包含请求协议+真实请求。

-M --http-method=<yes|no>

HTTP 请求方法开关。将创建一个请求字段包含请求方法+真实请求。

-o --output=<json|csv>

将给定文件重定向到标准输出，通过后缀名决定输出格式：

/path/file.csv - Comma-separated values (CSV)

/path/file.json - JSON (JavaScript Object Notation)

/path/file.html - HTML

-q --no-query-string

忽略请求的查询字符串。即： www.google.com/page.htm?query => www.google.com/page.htm

注意: 去掉查询字符串将极大降低内存消耗，特别对带时间戳的请求。

-r --no-term-resolver

在终端输出时禁止 IP 解析。

--444-as-404

将非标准状态 444 作为 404 处理。

--4xx-to-unique-count

将 4xx 客户端错误数加到独立访客数中。

--all-static-files

统计包含查询字符串的静态文件。

--date-spec=<date|hr>

设置日期的显示格式，一种是标准日期格式(默认)，一种是日期后附加小时的格式。

仅在访客面板有效。对于在小时级别分析访客数据很有帮助。显示格式示例：18/Dec/2010:19

--double-decode

解码双重编码的值。包括 UserAgent，Request 以及 Referer。

--enable-panel=<PANEL>

开启指定面板。面板列表：

VISITORS

REQUESTS

REQUESTS\_STATIC

NOT\_FOUND

HOSTS

OS

BROWSERS

VISIT\_TIMES

VIRTUAL\_HOSTS

REFERRERS

REFERRING\_SITES

KEYPHRASES

STATUS\_CODES

REMOTE\_USER

GEO\_LOCATION

--hour-spec=<hour|min>

设定时间的显示格式，一种是标准时间格式(默认)，一种是时间后附加分钟数(每十分钟)的格式。

用于时间分布面板。对于在特定时间段分析流量峰值很有用处。

--ignore-crawlers

忽略爬虫。

--ignore-panel=<PANEL>

忽略指定面板。面板列表：

VISITORS

REQUESTS

REQUESTS\_STATIC

NOT\_FOUND

HOSTS

OS

BROWSERS

VISIT\_TIMES

VIRTUAL\_HOSTS

REFERRERS

REFERRING\_SITES

KEYPHRASES

STATUS\_CODES

REMOTE\_USER

GEO\_LOCATION

--ignore-referer=<referer>

忽略被统计的来路。支持通配符。例如： \*.domain.com ww?.domain.\*

--ignore-status=<STATUS>

忽略解析或者显示一个或者多个状态码。如果有多个状态码，使用此参数每次指定一个。

--num-tests=<number>

设定测试行数，即使用给定的 日志/日期/时间 格式测试访问日志。默认值为 10 行。如果设置为 0 ，解析器不会做任何测试而是直接解析整个文件。如果在达到 number 之前，有一行匹配上了给定的 日志/日期/时间 格式，则解析器会认为日志文件是有效的，否则 GoAccess 会返回 EXIT\_FAILURE 并显示相关的错误信息。

--process-and-exit

解析日志，且退出时不输出数据。主要用于仅希望往磁盘数据库中添加数据而无需输出报告时使用。

--real-os

显示真实的操作系统名称。例如： Windows XP, Snow Leopard.

--sort-panel=<PANEL,FIELD,ORDER>

S在初始化载入是对面板进行排序。排序选项使用逗号分隔。选项使用这样的格式：PANEL,METRIC,ORDER

## 9、可用指标

BY\_HITS 按访问量

BY\_VISITORS 按独立访客数

BY\_DATA 按数据

BY\_BW 按带宽

BY\_AVGTS 按平均处理时间

BY\_CUMTS 按累积处理时间

BY\_MAXTS 按最大处理时间

BY\_PROT 按 HTTP 协议

BY\_MTHD 按 HTTP 方法

可用排序

ASC

DESC

--static-file <extension>

添加静态文件后缀名。例如：.mp3。 后缀名区分大小写。

## 10、地理位置选项

-g --std-geoip

标准 GeoIP 数据库，低内存占用。

--geoip-database <geocityfile>

设定 GeoIP 数据库路径。例如：GeoLiteCity.dat。需要从 maxmind.com 上下载到本地。IPv4 和 IPv6 均可用支持。注意：--geoip-city-data 是 --geoip-database 的别名。

注意: 如果使用 GeoIP2，您需要从 MaxMind 下载 城市/国家 数据库，并通过 --geoip-database 设定。

其他选项

-h --help

查看帮助信息。

-V --version

显示版本信息并退出。

-s --storage

显示当前存储方法。比如：B+ Tree, Hash。

--dcf

显示默认配置文件的路径，如果没有使用 -p 指定。

## 11、磁盘存储选项

--keep-db-files

在磁盘上保存已解析的数据。如果数据库文件存在，则文件将被覆盖。此参数应用于第一个数据集。如果设置此参数为 false 则在退出程序时将删除所有数据库。示例见下文。

仅在设置了 --enable-tcb=btree 时有效。

--load-from-disk

从磁盘载入之前存储过的数据。如果仅读取已保存的数据，则需要退出数据库文件。参考 keep-db-files 即相关示例见下文。

仅在设置了 --enable-tcb=btree 时有效。

--db-path <dir>

设置磁盘数据库文件的存储路径。默认值为 /tmp 目录。

仅在设置了 --enable-tcb=btree 时有效。

--xmmap <num>

设置附加内存映射的大小，单位为字节。默认值为0。

仅在设置了 --enable-tcb=btree 时有效。

--cache-lcnum <num>

指定被缓存的最大叶子节点数目。如果取值小于 0，则使用默认值。默认值为 1024。设定较大的值以获得较快的处理速度，同时会增加内存消耗。较小的值则会降低内存消耗。

仅在设置了 --enable-tcb=btree 时有效。

--cache-ncnum <num>

指定被缓存的最大非叶子节点数目。如果取值小于 0，则使用默认值。默认值为 512。

仅在设置了 --enable-tcb=btree 时有效。

--tune-lmemb <num>

指定每一个叶子页面的成员数量。如果取值小于 0，则使用默认值。默认值为 128。

仅在设置了 --enable-tcb=btree 时有效。

--tune-nmemb <num>

指定每一个非叶子页面的成员数量。如果取值小于 0，则使用默认值。默认值为 256。

仅在设置了 --enable-tcb=btree 时有效。

--tune-bnum <num>

指定每个 Bucket 组的元素数量。如果取值小于 0，则使用默认值。默认值为 32749。建议 Bucket 组大小的取值为已保存页面数的 1 到 4 倍。

仅在设置了 --enable-tcb=btree 时有效。

--compression <zlib|bz2>

指定页面采用的压缩编码(ZLIB|BZ2)。

仅在设置了 --enable-tcb=btree 时有效。

自定义 日志/日期 格式

GoAccess 可以解析虚拟的任意 Web 日志格式。

预定义的选项包括：通用日志格式，联合日志格式，包含虚拟主机，W3C 格式以及亚马逊 CloudFront(分布式下载)。

GoAccess 允许任意的自定义格式字符串。

有两种方法配置日志格式。最简单的方式是运行 GoAccess 时使用 -c 显示一个配置窗口。但是这种方式不是永久有效的，因此你需要在配置文件中设定格式。

配置文件位于：%sysconfdir%/goaccess.conf 或者 ~/.goaccessrc

注意: %sysconfdir% 可能是 /etc/, /usr/etc/ 或者 /usr/local/etc/

time-format 参数 time-format 后跟随一个空格符，指定日志的时间格式，包含普通字符与特殊格式说明符的任意组合。他们都由百分号 (%)开始。参考 `man strftime`。 %T 或者 %H:%M:%S.

注意: 如果给定的时间戳以微秒计算，则必须在 time-format 中使用参数 %f 。

date-format 参数 date-format 后跟随一个空格符，指定日志的日期格式，包含普通字符与特殊格式说明符的任意组合。他们都由百分号 (%)开始。参考 `man strftime`。

注意: 如果给定的时间戳以微秒计算，则必须在 time-format 中使用参数 %f 。

log-format 参数 log-format 后跟随一个空格符或者制表分隔符(\t)，用于指定日志字符串格式。

## 12、特殊格式说明符

%x 匹配 time-format 和 date-format 变量的日期和时间字段。用于使用时间戳来代替日期和时间两个独立变量的场景。

%t 匹配 time-format 变量的时间字段。

%d 匹配 date-format 变量的日期字段。

%v 根据 canonical 名称设定的服务器名称(服务区或者虚拟主机)。

%e 请求文档时由 HTTP 验证决定的用户 ID。

%h 主机(客户端IP地址，IPv4 或者 IPv6)。

%r 客户端请求的行数。这些请求使用分隔符(单引号，双引号)引用的部分可以被解析。否则，需要使用由特殊格式说明符(例如：%m, %U, %q 和 %H)组合格式去解析独立的字段。

注意: 既可以使用 %r 获取完整的请求，也可以使用 %m, %U, %q and %H 去组合你的请求，但是不能同时使用。

%m 请求的方法。

%U 请求的 URL。

注意: 如果查询字符串在 %U中，则无需使用 %q。但是，如果 URL 路径中没有包含任何查询字符串，则你可以使用 %q 查询字符串将附加在请求后面。

%q 查询字符串。

%H 请求协议。

%s 服务器回传客户端的状态码。

%b 回传客户端的对象的大小。

%R HTTP 请求的 "Referer" 值。

%u HTTP 请求的 "UserAgent" 值。

%D 处理请求的时间消耗，使用微秒计算。

%T 处理请求的时间消耗，使用带秒和毫秒计算。

%L 处理请求的时间消耗，使用十进制数表示的毫秒计算。

%^ 忽略此字段。

%~ 继续解析日志字符串直到找到一个非空字符(!isspace)。

~h 在 X-Forwarded-For (XFF) 字段中的主机(客户端 IP 地址，IPv4 或者 IPv6)。

# 九、将linux服务器设置为中文

以下方式有待研究--redhat8设置后终端打不开

$ localectl  set-locale LANG=zh\_CN.UTF8

# 十、负载均衡

## 1、默认方式--轮询的方式

upstream node { //在http模块下，和server平行

server 127.0.0.1:9001;

server 127.0.0.1:9002;

server 127.0.0.1:9003;

}

location / { //将代理设置为node模块（node的名称可以随便起）

proxy\_pass http://node;

}

利用postman进行模拟接口压测，可以通过pm2 log查看到三个端口均分配

## 2、权重weight

upstream node {

server 127.0.0.1:9001 weight=3;

server 127.0.0.1:9002 weight=2;

server 127.0.0.1:9003 weight=1;

}

权重越大服务器承载的并发就越高

## fail\_timeout backup

fail\_timeout是故障等待超时时间

backup是备用服务器参数，可以为一个upstream设置一个backup的server，在生产server全部都出问题之后，可以自动切换到备用server上，为回复服务争取时间

upstream node {

server 127.0.0.1:9001 fail\_timeout=60;秒

server 127.0.0.1:9002 fail\_timeout=20;

server 127.0.0.1:9003 backup;

}