配置文件：

/etc/nginx/nginx.conf

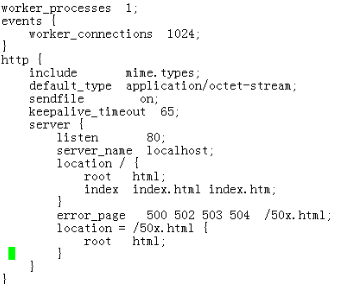
日志：

/var/log/nginx/accecc.log error.log

killall -HUP nginx 重新读取配置文件而不关闭

nginx -t 验证配置文件

vim /nginx.conf



worker\_processer //cpu总核数

//nginx 两个功能：web代理 mail代理

//先是全局的配置

event{

worker\_connections 1024 //每个worker连接个数

//里面还可以用别的指令

}

http{ //所有关于web的内容都在这里

sendfile on;

keepalive\_timeout 65; //非活动进程占用较少

server{ //定义虚拟主机，默认主机

listen 80;

servername www.text.com

location / { //定义路径 融合directory和DocumentRoot

root html; //url的根,相对于软件安装目录

}

error\_page 500 /50x.html

location = /50x.html{ //=精确匹配，只匹配单个文件 root html; //还有~正则表达式的模式匹配 等标示符

}

}

//还有定义php的location 默认没有启用

server { //虚拟主机

listen 172.16.0.2:80; //基于ip的虚拟主机

server\_name [www.test.com](http://www.test.com) // 基于主机时不需要ip

location / {

root /nginx/bbs;

index index.html

}

}

}

**配置选项详解：**

**listen指令：**

syntax: listen address:port [ default [ backlog=num | rcvbuf=size | sndbuf=size | accept\_filter=filter | deferred | bind | ssl ] ]

default: listen 80 listen只能用在server中

backlog：后台记录暂时的缓存，如果超出缓冲区数量的话

rcvbuf：recevie buffer

sndbuf：send buffer

**location 指令**：

syntax: location [=|~|~\*|^~|@] /uri/ { ... } location+操作符+uri+一段指令

default: no

context: server

This directive allows different configurations depending on the URI. It can be configured using both literal strings and regular expressions. To use regular expressions, you must use a prefix:

1. "~" for case sensitive matching 严格区分大小写

2. "~\*" for case insensitive matching uri不区分大小写

= 精确匹配 只匹配路径本身，不匹配之下的文件和子路径

To determine which location directive matches a particular query, the literal strings are checked first. Literal strings match the beginning portion of the query - the most specific match will be used. Afterwards, regular expressions are checked in the order defined in the configuration file. The first regular expression to match the query will stop the search. If no regular expression matches are found, the result from the literal string search is used.

It is possible to disable regular expression checks after literal string matching by using "^~" prefix. If most specific match literal location have this prefix - regular expressions aren't checked.

By using "=" prefix on may define exact match between URI and location. On match search stops immediately as further search has no sense. E.g. if the request "/" occurs frequently, using "location = /" will speed up processing of this request a bit as search will stop after first comparison.

On exact match with literal location without "=" or "^~" prefixes search is also immediately terminated.

To summarize, the order in which directives are checked is as follows:

1. Directives with the "=" prefix that match the query exactly. If found, searching stops.

2. All remaining directives with conventional strings. If this match used the "^~" prefix, searching stops.

3. Regular expressions, in the order they are defined in the configuration file.

4. If #3 yielded a match, that result is used. Otherwise, the match from #2 is used.

It is important to know that nginx does the comparison against decoded URIs. For example, if you wish to match "/images/%20/test", then you must use "/images/ /test" to determine the location.

location / {

//只要通过根能访问都能匹配

}

location = / {

//只匹配根自身 之下的任何文件都不匹配

}

Example:

location = / { //只输入了一个根的时候被这个匹配

# matches the query / only.

[ configuration A ]

}

location / {

# matches any query, since all queries begin with /, but regular

# expressions and any longer conventional blocks will be

# matched first.

[ configuration B ]

}

location ^~ /images/ { //不是做正则表达式匹配，而是做逐字符匹配

# matches any query beginning with /images/ and halts searching,

# so regular expressions will not be checked.

[ configuration C ]

}

location ~\* \.(gif|jpg|jpeg)$ { //

# matches any request ending in gif, jpg, or jpeg. However, all

# requests to the /images/ directory will be handled by

# Configuration C.

[ configuration D ]

}

Example requests:

\* / -> configuration A

\* /a.html->B

\* /documents/document.html -> configuration B

\* /images/1.gif -> configuration C

\* /documents/1.jpg -> configuration D

Note that you could define these 4 configurations in any order and the results would remain the same. While nested locations are allowed by the configuration file parser, their use is discouraged and may produce unexpected results.

The prefix "@" specifies a named location. Such locations are not used during normal processing of requests, they are intended only to process internally redirected requests

//反向代理的负载均衡时用到，先不讲

**root指令：**

syntax: root path

default: root html

context: http, server, location, if in location root specifies the document root for the requests. For example, with this configuration

location /i/ {

root /spool/w3;

}

/i/c/d.html

/spool/w3/i/c/d.html

A request for "/i/top.gif" will return the file "/spool/w3/i/top.gif". You can use variables in the argument.

note: Keep in mind that the root will still append the directory to the request so that a request for "/i/top.gif" will not look in "/spool/w3/top.gif" like might happen in an Apache-like alias configuration where the location match itself is dropped. Use the alias directive to achieve the Apache-like functionality.

**alias指令：**

DocumentRoot /web/htdocs

URI: /a/b.html

URI: /bbs

syntax: alias file-path|directory-path;

default: no

context: location

This directive assigns a path to be used for the indicated location. Note that it may look similar to the root directive, but the document root doesn't change, just the file system path used for the request.

For example:

location / {

root /spool/w3;

}

location /bbs/ {

alias /spool/bbs/;

}

URI: /i/a.html --> /spool/w3/images/a.html

The request "/i/top.gif" will return the file "/spool/w3/images/top.gif".

Alias can also be used in a regex specified location.

For example:

location ~ ^/download/(.\*)$ {

alias /home/website/files/$1;

}

The request "/download/book.pdf" will return the file "/home/website/files/book.pdf"

It is possible to use variables in the replacement path.

**root和alias区别：**

location = /bbs/a.html {

root /web/vhosts;

alias /web/vhosts/bbs/a.html;

}

/web/vhosts/bbs/ 假设在这里

**server指令：**

syntax: server {...}

default: no

context: http

Directive assigns configuration for the virtual server.

There is no separation of IP and name-based (the Host header of the request) servers.

Instead, the directive listen is used to describe all addresses and ports on which incoming connections can occur, and in directive server\_name indicate all names of the server.

**index指令：**指定默认主页

syntax: index file-path [file-path [ ... ] ];

default: no

context: server, location

Sets the default file to serve if no file is specified in the URL. Multiple files can be specified. If the first file isn't found, the second will be used and so on.

**autoindex module:模块**//如果某个目录没有主页是否打开自动索引，如果不打开会报错

This module provides automatic directory listings.

The request only reaches the ngx\_http\_autoindex\_module when the ngx\_http\_index\_module did not find an index file.

Example configuration

location / {

autoindex on; //自动索引开

}

syntax: autoindex [ on|off ]

default: autoindex off

context: http, server, location

Enables or disables the automatic directory listing.

autoindex\_exact\_size

syntax: autoindex\_exact\_size [ on|off ]

default: autoindex\_exact\_size on

context: http, server, location

Defines how to represent file sizes in the directory listing -- either accurately (in bytes), or rounded (KB, MB or GB).

autoindex\_localtime

syntax: autoindex\_localtime [ on|off ]

default: autoindex\_localtime off

context: http, server, location

Enables showing file times as local time. Default is "off" (GMT time).

**AccessModule模块：**

This module provides a simple host-based access control.

Module nginx\_http\_access\_module makes it possible to control access for specific IP-addresses of clients.

Access rules are checked according to the order of their declaration. The first rule that matches a particular address or set of addresses is the one that is obeyed.

Example configuration:

location / {

deny 192.168.1.1;

allow 192.168.1.0/24;

allow 10.1.0.0/16;

deny all;

}

In this example access is granted to networks 10.1.1.0/16 and 192.168.1.0/24 with the exception of address 192.168.1.1, which is denied access together with all other addresses as defined by the deny all rule that is matched last in this location block.

Note that the order of the deny/allow is of the utmost importance.

allow

syntax: allow [ address | CIDR | all ]

default: no

context: http, server, location, limit\_except

Directive grants access for the network or addresses indicated.

deny

syntax: deny [ address | CIDR | all ]

default: no

context: http, server, location, limit\_except

Directive forbids access for the network or addresses indicated.

**Virtual Hosts Examples 虚拟主机样例：**

http {

server {

listen 80;

server\_name www.a.com;

access\_log logs/a.access.log main;

index index.html;

root /var/www/a.com/htdocs;

}

server {

listen 8080;

server\_name www.b.com;

access\_log logs/b.access.log main;

index index.html;

root /var/www/b.com/htdocs;

}

}

www.magedu.com /www/magedu

dz.magedu.com /www/discuz

A Default Catchall Virtual Host

http {

server {

listen 80 default;

server\_name \_;

access\_log logs/default.access.log main;

server\_name\_in\_redirect off;

index index.html;

root /var/www/default/htdocs;

}

}

**开启Nginx状态监控的功能：**

location /nginx\_status { //这个名字可以自己取

stub\_status on;

access\_log off;

}

stub\_status

syntax: stub\_status on

default: None

**context: location**

Enables the status handler in this location.

The stub status module reports status similar to mathopd's status page. It is plain text information like

状态说明：

Active connections: 291

server accepts handled requests

16630948 16630948 31070465

Reading: 6 Writing: 179 Waiting: 106

active connections -- number of all open connections including connections to backends

server accepts handled requests -- nginx accepted 16630948 connections, handled 16630948 connections (no one was closed just it was accepted), and handles 31070465 requests (1.8 requests per connection)

reading -- nginx reads request header

writing -- nginx reads request body, processes request, or writes response to a client

waiting -- keep-alive connections, actually it is active - (reading + writing)

**启用基于用户的认证：**

server {

server\_name www.magedu.com;

. . .

location / {

auth\_basic "Restricted"; //这仅仅是个名称

auth\_basic\_user\_file /etc/nginx/.htpasswd; //这个也是apche生成的

. . .

}

location ~ /\.ht {

deny all;

}

}

