什么是Nginx

`Nginx`是一个高性能的`http`和反向代理服务器，其特点是占用内存小，并发能力强。`Nginx`专为性能优化而开发，性能是其最重要的考量，能经受高负载的考验，有报告表明能支持高达50000个并发连接数。

Nginx可以从事的用途

web服务器

提供Web信息浏览服务。它只需支持HTTP协议、HTML文档格式及URL。

反向代理

正向代理：在浏览器中配置代理服务器，通过代理服务器进行互联网访问。

反向代理：将请求发送到反向代理服务器，由反向代理服务器去选择目标服务器获取数据后，再返回给客户端，此时反向代理服务器和目标服务器对外就是一个服务器，暴漏的是代理服务器地址。

负载均衡

如果请求数过大，单个服务器解决不了，我们增加服务器的数量，然后将请求分发到各个服务器上，将原先请求集中到单个服务器的情况改为请求分发到多个服务器上，就是负载均衡。

Nginx安装

自动安装

CentOS

```
yum install -y nginx
```

Ubuntu

```
apt-get install -y nginx
```

当终端显示出Complete!字样时，则代表我们的Nginx已经安装成功了。

Docker 方式安装（linux通用，需要提前安装docker）：

```
docker container run \
  --rm \
  --name mynginx \
  --volume "$PWD/html":/usr/share/nginx/html \
  --volume "$PWD/conf":/etc/nginx \
  -p 80:80 \
  -d \
  nginx
```

以上参数含义：

```
-d：在后台运行
-p ：容器的80端口映射到本地80端口，格式为本地端口:容器端口
--rm：容器停止运行后，自动删除容器文件
--name：容器的名字为mynginx
--volume "$PWD/conf":/etc/nginx表示把容器的配置目录/etc/nginx，映射到本地的conf子目录。
```

查看Nginx版本：

```
nginx -v
#在这里我安装的是1.16.1版本的nginx
```

其他版本的Linux可以自行查阅资料或者参考手动安装方式

手动安装

`Nginx`需要几个依赖包，分别是`pcre`，`openssl`，`zlib`，在安装`nginx`之前需要先安装这几个依赖。

2.1 安装pcre依赖

1. 使用命令下载`pcre`压缩包

```
wget http://downloads.sourceforge.net/project/pcre/pcre/8.37/pcre-8.37.tar.gz
```

1. 解压压缩文件

```
tar -xvf pcre-8.37.tar.gz
```

1. 进入解压后的名录，执行以下命令

```
./configure
```

1. 使用以下命令进行编译安装

```
make && make install
```

1. 查看安装的`pcre`版本号

```
pcre-config --version
```

2.2 安装openssl，zlib等依赖

```
yum -y install make zlib zlib-devel gcc-c++ libtool openssl openssl-devel
```

2.3 安装nginx

1. `nginx`官网下载`nginx`，官网地址：https://nginx.org/download/；
2. 将压缩包拖到服务器上;
3. 使用命令`tar -xvf nginx-1.12.2.tar.gz`解压压缩包;
4. 使用命令`./configure`检查;
5. 使用命令`make && make isntall`编译安装;

安装成功后，在`usr`会多出来一个文件夹，`local/nginx`，在`nginx`的`sbin`文件夹下有启动脚本。

Nginx自带常用命令

```
nginx -s stop       快速关闭Nginx，可能不保存相关信息，并迅速终止web服务。
nginx -s quit       平稳关闭Nginx，保存相关信息，有安排的结束web服务。
nginx -s reload     因改变了Nginx相关配置，需要重新加载配置而重载。
nginx -s reopen     重新打开日志文件。
nginx -c filename   为 Nginx 指定一个配置文件，来代替缺省的。
nginx -t            不运行，仅仅测试配置文件。nginx 将检查配置文件的语法的正确性，并尝试打开
配置文件中所引用到的文件。
nginx -v            显示 nginx 的版本。
nginx -V            显示 nginx 的版本，编译器版本和配置参数。
```

Nginx启动

```
##在centos7+ 启动nginx服务
systemctl start nginx
#centos6+ 上启动nginx服务
service nginx start
#或，简单粗暴一句，通过这种方式启动nginx 使用systemctl status nginx查看的状态是未运行
nginx
```

Nginx停止

```
##在centos7+ 停止nginx服务
systemctl stop nginx
#centos6+ 上停止nginx服务
service nginx stop
#粗鲁的停止，下班了，不干了，就算请求来了我也不接了。
nginx -s stop
##优雅的停止，Nginx在退出前完成已经接受的连接请求。
nginx -s quit
```

Nginx重启

当我们修改了nginx的某些配置，为了使配置生效，我们往往需要重启nginx，同样的，linux下依然有两种方式来重启我们的nginx服务:

```
##在centos7+ 重启nginx服务
systemctl restart nginx
#centos6+ 上重启nginx服务
service nginx restart
#使用nginx命令停止，推荐这个
nginx -s reload
```

而具体使用nginx原生的nginx -s 操作还是linux提供的systemctl ，这个主要看个人喜好，实际两者的功能是差不多的，并没有什么明显的不同。

四种解决Nginx出现403 forbidden 报错的方法

访问机器所在的IP地址，检查安装是否成功

我是在在本地用虚拟机中通过yum安装nginx的，安装一切正常，但是访问时报403，

于是查看nginx日志，路径为/var/log/nginx/error.log。打开日志发现报错Permission denied，详细报错如下：

```
1.    open() "/data/www/1.txt" failed (13: Permission denied), client:
192.168.1.194, server: www.web1.com, request: "GET /1.txt HTTP/1.1", host:
"www.web1.com"
```

没有权限？于是找了不少资料，可以通过下面四步排查解决此问题。你可能只是其中之前配置有问题，不一定四个步骤都用上。

一、由于启动用户和nginx工作用户不一致所致

1.1查看nginx的启动用户，发现是nobody，而为是用root启动的

```
命令：ps aux | grep "nginx: worker process" | awk '{print $1}'
```

1.2将nginx.config的user改为和启动用户一致，

命令：vi conf/nginx.conf

二、缺少index.html或者index.php文件，就是配置文件中index index.html index.htm这行中的指定的文件。

```
1.    server {
2.      listen       80;
3.      server_name  localhost;
4.      index  index.php index.html;
5.      root  /data/www/;
6.    }
```

如果在/data/www/下面没有index.php,index.html的时候，直接文件，会报403 forbidden。

三、权限问题，如果nginx没有web目录的操作权限，也会出现403错误。

解决办法：修改web目录的读写权限，或者是把nginx的启动用户改成目录的所属用户，重启Nginx即可解决。

```
1.    chmod -R 777 /data
2.    chmod -R 777 /data/www/
```

四、SELinux设置为开启状态（enabled）的原因。

4.1、查看当前selinux的状态。

```
1.    /usr/sbin/sestatus
```

4.2、将SELINUX=enforcing 修改为 SELINUX=disabled 状态。

```
1.    vi /etc/selinux/config
2.
3.    #SELINUX=enforcing
4.    SELINUX=disabled
```

4.3、重启生效。reboot。

```
1.    reboot
```

SELinux作为Linux的安全防护插件，在普通机器和其他生产环境有额外的系统做安全防护，所以这里关闭即可。

Nginx配置概要

nginx本身作为一个完成度非常高的负载均衡框架，和很多成熟的开源框架一样，大多数功能都可以通过修改配置文件来完成，使用者只需要简单修改一下nginx配置文件，便可以非常轻松的实现比如反向代理，负载均衡这些常用的功能，同样的，和其他开源框架比如tomcat一样，nginx配置文件也遵循着相应的格式规范，并不能一顿乱配，在讲解如何使用nginx实现反向代理，负载均衡等这些功能的配置前，我们需要先了解一下nginx配置文件的结构。

既然要了解nginx的配置文件，那我总得知道nginx配置文件在哪啊，nginx配置文件默认都放在nginx安装路径下的conf目录，而主配置文件nginx.conf自然也在这里面，我们下面的操作几乎都是对nginx.conf这个配置文件进行修改。

可是，我怎么知道我nginx装哪了？我要是不知道nginx装哪了咋办？

这个，细心的朋友们可能会发现，运行nginx -t命令，下面除了给出nginx配置文件是否OK外，同时也包括了配置文件的路径。诺，就是这个

```
[root@localhost ~]# nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

使用vim打开该配置文件，我们一探究竟，不同版本的配置文件可能稍有不同，我的配置文件内容如下：

```
# For more information on configuration, see:
#   * Official English Documentation: http://nginx.org/en/docs/
```

```
#   * Official Russian Documentation: http://nginx.org/ru/docs/

user nginx;
worker_processes auto;
error_log /var/log/nginx/error.log;
pid /run/nginx.pid;

# Load dynamic modules. See /usr/share/doc/nginx/README.dynamic.
include /usr/share/nginx/modules/*.conf;

events {
    worker_connections 1024;
}

http {
    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;

    sendfile            on;
    tcp_nopush          on;
    tcp_nodelay         on;
    keepalive_timeout   65;
    types_hash_max_size 2048;

    include             /etc/nginx/mime.types;
    default_type        application/octet-stream;

    # Load modular configuration files from the /etc/nginx/conf.d directory.
    # See http://nginx.org/en/docs/ngx_core_module.html#include
    # for more information.
    include /etc/nginx/conf.d/*.conf;

    server {
        listen       80 default_server;
        listen       [::]:80 default_server;
        server_name  _;
        root         /usr/share/nginx/html;

        # Load configuration files for the default server block.
        include /etc/nginx/default.d/*.conf;

        location / {
        }

        error_page 404 /404.html;
            location = /40x.html {
        }

        error_page 500 502 503 504 /50x.html;
            location = /50x.html {
        }
    }

}
```

?? ?

这一堆都是啥玩意er，完全没有头绪啊

没关系，下面我们就来详细分析一下nginx.conf这个文件中的内容。

按照功能划分，我们通常将nginx配置文件分为三大块，**全局块，events块，http块。**

```
# 全局块
...
# events块
events {
   ...
}
# http块
http
{
    # http全局块
    ...
    # 虚拟主机server块
    server
    {
        # server全局块
        ...
        # location块
        location [PATTERN]
        {
            ...
        }
        location [PATTERN]
        {
            ...
        }
    }
    server
    {
      ...
    }
    # http全局块
    ...
}
```


第一部分：全局块

首先映入眼帘的这一堆：

```
user nginx;
worker_processes auto;
error_log /var/log/nginx/error.log;
pid /run/nginx.pid;

# Load dynamic modules. See /usr/share/doc/nginx/README.dynamic.
include /usr/share/nginx/modules/*.conf;
```

我们称之为**全局块**，知识点呐朋友们，要记住，这里呢，主要会设置一些影响 nginx 服务器整体运行的配置指令，主要包括配 置运行 Nginx 服务器的用户（组）、允许生成的 worker process 数，进程 PID 存放路径、日志存放路径和类型以及配置文件的引入等。

比如 worker_processes auto； 这一行，worker_processes 值越大，我们nginx可支持的并发数量就越多，很多人想这不就爽了吗，我设置成正无穷，无限并发flag达成，秒杀问题轻松解决，这个，受自己服务器硬件限制的，不能乱来。设置auto表示会自动根据当前cpu的核心数来分配worker,规则是1个核心1个worker.

第二部分：events 块：

```
events {
    worker_connections 1024;
}
```

这一堆，就是我们配置文件的第二部分，**events 块**

起名字这么随意的么，那第三部分是不是叫http块?

wc，这你都知道，是的

events 块涉及的指令主要影响 Nginx 服务器与用户的网络连接，常用的设置包括是否开启对多 work 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来处理连接请求，每个 work process 可以同时支持的最大连接数等。

第三部分：http块：

内容太多，略

```
http {
    server {
        }
}
```

> 注意：
>
> http是一个大块，里面也可以包括很多小块，比如http全局块，server块等。

http 全局块配置的指令包括文件引入、MIME-TYPE 定义、日志自定义、连接超时时间、单链接请求数上限等。

而http块中的server块则相当于一个虚拟主机，一个http块可以拥有多个server块。

server块又包括**server全局**块，和**location**块。

全局server块主要包括了本虚拟机主机的监听配置和本虚拟主机的名称或 IP 配置

location块则用来对虚拟主机名称之外的字符串进行匹配，对特定的请求进行处理。地址定向、数据缓存和应答控制等功能，还有许多第三方模块的配置也在这里进行。比如，对/usl相关的请求交给8080来处理，/admin则较给8081处理。

说了这么多，我还是不是特别理解咋办，问题不大，接下来我们通过几个实例来帮助大家更好的理解这些配置在实际中所发挥的作用。

Nginx实战

配置web服务器

将我们上1节课程部署到linux上的静态文件代理给Nginx处理。当我们访问服务器IP时，可以自动帮我们返回静态文件主页。

知识点:

```
location /path/ {
    # 相对匹配，文件路径等于 root + location，如访问 http://ip/path/ 对应的文件目录是/root/software/autotpsite/dist/path/
    root    /root/software/autotpsite/dist;
    # 绝对匹配，文件路径等于alias对应目录与location无关，目录必须以/结尾 与root二选一 ，如访问，如访问 http://ip/path/ 对应的文件目录 /root/software/autotpsite/dist/
    # alias   /root/software/autotpsite/dist/;
}
```

以下配置，放在server块中:

```
location / {
    root    /root/software/autotpsite/dist;    # 相对匹配，文件路径等于 root + location
    # alias   /root/software/autotpsite/dist/;  # 绝对匹配，文件路径等于alias对应目录与location无关，目录必须以/结尾
}
```

> 注意以上目录改成你静态文件目录(dist)的实际路径

部署反向代理

反向代理服务器和目标服务器对外就是一个服务器，暴露的是代理服务器地址，隐藏了真实服务器 IP 地址。

前面我们访问服务器的时候，只能返回静态文件，而静态文件中的Ajax请求全部失效了，原因就是web服务没有代理app服务，app服务是由uWSGI代理的。所以我们要做的就是将发送app服务的请求转给uWSGI就可以了。

核心配置:

```
### 省略了http和server块，这里加到server块里面就可以
location /api/ {
    include        uwsgi_params;
    uwsgi_pass     127.0.0.1:8081;  # 此方式需要 uwsgi采用socket连接方式

}
#location /jira {
#    include        uwsgi_params;
#    uwsgi_pass     127.0.0.1:8081;
#}
location / {
    alias      /root/software/autotpsite/dist/;
}
```

或者使用正则的方式匹配URL

```
location / {
    alias      /root/software/autotpsite/dist/;
}

location ~/(api/|jira/) {
    include        uwsgi_params;
    uwsgi_pass     127.0.0.1:8081;
    # proxy_pass   http://127.0.0.1:8081;   # 通用配置方式
}
```

更新uWSGI服务配置

```
[uwsgi]
# 配置文件处于项目根目录，因此设置为相对路径即可，复用性更高
chdir = ./
module = autotpsite.wsgi:application
# Nginx使用uwsgi_pass做方向代理时 需要设置成socket
socket = 0.0.0.0:8081
#http-socket = 0.0.0.0:8081   #对应nginx的uwsgi模式
#http = 0.0.0.0:8081      #对应nginx的uwsgi模式
master = true
pidfile = uwsgi8081.pid
daemonize = uwsgi_server.log
# 只记录错误信息
disable-logging = true


# 新增配置--允许多线程
enable-threads = true
# 设置请求头最大字节数，用于socket模式
buffer-size = 40960
```


location指令说明:

功能：用于匹配URL

语法如下：

```
1、= ：用于不含正则表达式的 uri 前，要求请求字符串与 uri 严格匹配，如果匹配
成功，就停止继续向下搜索并立即处理该请求。
2、~：用于表示 uri 包含正则表达式，并且区分大小写。
3、~*：用于表示 uri 包含正则表达式，并且不区分大小写。
4、^~：用于不含正则表达式的 uri 前，要求 Nginx 服务器找到标识 uri 和请求字
符串匹配度最高的 location 后，立即使用此 location 处理请求，而不再使用 location
块中的正则 uri 和请求字符串做匹配。
复制代码
```

注意:

> 如果 uri 包含正则表达式，则必须要有 ~ 或者 ~* 标识。

部署负载均衡

在nginx中配置负载均衡也是十分容易的，同时还支持了多种负载均衡策略供我们灵活选择。为了演示逼真，我准备了两台虚拟机，当然你也可以在一台机器上部署两个不同端口的服务，都可以模拟。

然后修改我们的http块如下:

```
http {

###此处省略一大堆没有改的配置

    ##自定义我们的服务列表
    upstream myserver{
       server 127.0.0.1:8081;
       server 192.168.21.142:8081;
    }


   server {
       listen       80 ; ##设置我们nginx监听端口为8888
       server_name  myserve;

       # Load configuration files for the default server block.
       include /etc/nginx/default.d/*.conf;

       location / {
           alias    /root/software/autotpsite/dist/;
       }

       location ~/(api/|jira/) {
           proxy_pass   http://myserver; # 核心配置在这里
           proxy_connect_timeout 10;   # 超时时间，单位秒
       }

       error_page 404 /404.html;
           location = /40x.html {
       }

       error_page 500 502 503 504 /50x.html;
           location = /50x.html {
       }
   }


}
```

uWSGI配置:

```
[uwsgi]
chdir = /root/software/autotpsite/
module = autotpsite.wsgi:application
http-socket = 0.0.0.0:8081
master = true
pidfile = /root/software/autotpsite/uwsgi8081.pid
daemonize = /root/software/autotpsite/uwsgi_server.log
# 只记录错误信息
disable-logging = true
```

这就完了?当然还没有，之前就有说过，nginx提供了三种不同的负载均衡策略供我们灵活选择，分别是：

- **轮询(默认方式):** 每个请求按时间顺序逐一分配到不同的后端服务器，如果后端服务器 down 掉，能自动剔除。

  用法：啥也不加，上文实例就是默认的方式，就是默认的

- **权重(weight):** weight 代表权重,默认为 1,权重越高被分配的客户端越多,权重越大，能力越大，责任越大，处理的请求就越多。

  用法:

  ```
  upstream myserver{
      server 127.0.0.1:8081 weight =1;
      server 192.168.21.142:8081 weight=2;
  }
  ```

- **ip_hash**：每个请求按访问 ip 的 hash 结果分配，这样每个访客固定访问一个后端服务器，可以解决 session 的问题。

  用法：

  ```
   upstream myserver{
         ip_hash;#可与weight配合使用
         server 127.0.0.1:8081 weight =1;
         server 192.168.21.142:8081 weight =2;
       }
  ```