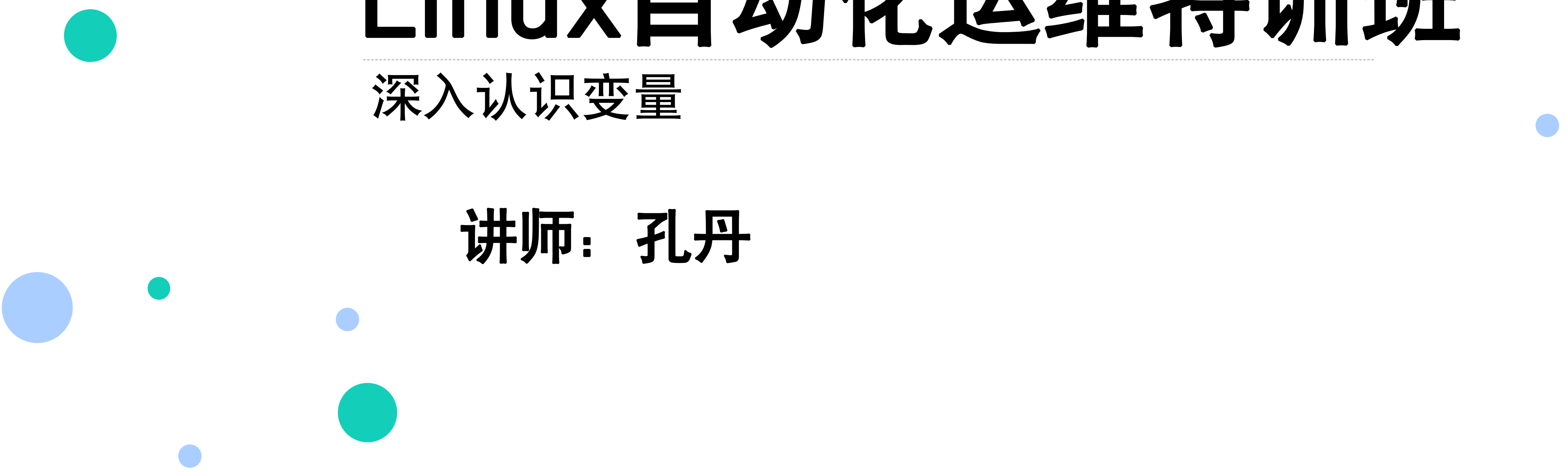




Linux自动化运维特训班

Tengine实战

讲师：孔丹



大纲

- Nginx简介
- Tengine简介
- Tengine配置文件
- Tengine虚拟主机
- Tengine安全优化

Nginx简介

- Web服务器、HTTP反向代理和邮件代理服务器
- • 俄罗斯程序员Igor Sysoev于2002年开始
- • 目前全球使用量排名第一
- • 2011年成立商业公司
- • 特点
 - 性能非常高
 - 资源占用（CPU、内存）非常节省
 - 内存池设计，非常稳定
 - 高度模块化，易于扩展

官方网站：<http://nginx.org/>

Nginx特性与优点

□ nginx的特性

Nginx使用基于事件驱动架构，使得其可以支持数以百万级别的TCP连接

高度的模块化和自由软件许可证是的第三方模块层出不穷

Nginx是一个跨平台服务器，可以运行在

Linux,Windows,FreeBSD,Solaris, AIX,Mac OS等操作系统上

这些优秀的设计带来的极大的稳定性

Nginx特性与优点

□ nginx的优点

1. 高并发连接：官方测试能够支撑5万并发连接，在实际生产环境中跑到2-3万并发连接数
2. 内存消耗少：在3万并发连接下，开启的10个nginx进程才消耗150M内存（ $15M \times 10 = 150M$ ）
3. 配置文件非常简单：风格跟程序一样通俗易懂
4. 成本低廉：nginx为开源软件，可以免费使用。而购买F5 BIG-IP、NetScaler等硬件负载均衡交换机则需要十多万至几十万人民币
5. 支持Rewrite重写规则：能够根据域名、URL的不同，将HTTP请求分到不同的后端服务器群组
6. 内置的健康检查功能：如果Nginx Proxy后端的某台Web服务器宕机了，不会影响前端访问
7. 节省带宽：支持GZIP压缩，可以添加浏览器本地缓存的Header头
8. 稳定性高：用于反向代理，宕机的概率微乎其微
9. 模块化设计：模块可以动态编译
10. 外围支持好：文档全，二次开发和模块较多
11. 支持热部署：可以不停机重载配置文件
12. 支持事件驱动、AIO（AsyncIO，异步IO）、mmap（Memory Map，内存映射）等性能优化

Nginx企业应用

- 作为Web服务软件
- 反向代理或负载均衡服务
- 前端业务数据缓存服务

Tengine简介

- Tengine是由淘宝网发起的Web服务器项目。它在Nginx的基础上，针对大访问量网站的需求，添加了很多高级功能和特性。Tengine的性能和稳定性已经在大型的网站如淘宝网，天猫商城等得到了很好的检验。它的最终目标是打造一个高效、稳定、安全、易用的Web平台。
- 从2011年12月开始，Tengine成为一个开源项目，Tengine团队在积极地开发和维护着它。Tengine团队的核心成员来自于淘宝、搜狗等互联网企业。Tengine是社区合作的成果，我们欢迎大家参与其中，贡献自己的力量。
- 官方站点: <http://tengine.taobao.org/>

The logo for Tengine, featuring the word "Tengine" in a stylized, italicized font with a blue-to-green gradient and a slight shadow effect.

Tengine安装和使用

□ 下载

```
[root@localhost ~]# wget -c  
http://tengine.taobao.org/download/tengine-2.3.2.tar.gz
```

□ 安装依赖

```
yum install -y pcre-devel openssl-devel  
yum install gcc gcc-c++ make -y
```

□ 创建用户

```
useradd -M -r -u 995 -s /sbin/nologin nginx
```

□ 解压

```
tar xf tengine-2.3.2.tar.gz -C /usr/src/
```

□ 配置

```
cd /usr/src/tengine-2.3.2/  
./configure --prefix=/usr/local/tengine --user=nginx --  
group=nginx --with-http_stub_status_module --with-  
http_ssl_module
```


Tengine安装和使用

- 编译

make

- 安装

make install

- 创建链接

ln -sv /usr/local/tengine/sbin/nginx /usr/sbin/nginx

Tengine安装和使用

□ 服务脚本

```
vim /usr/lib/systemd/system/nginx.service
```

```
[Unit]
```

```
Description=tengine - high performance web server
```

```
Documentation=http://tengine.taobao.org/documentation.html
```

```
After=network.target remote-fs.target nss-lookup.target
```

```
[Service]
```

```
Type=forking
```

```
PIDFile=/usr/local/tengine/logs/nginx.pid
```

```
ExecStartPre=/usr/sbin/nginx -t -c /usr/local/tengine/conf/nginx.conf
```

```
ExecStart=/usr/sbin/nginx -c /usr/local/tengine/conf/nginx.conf
```

```
ExecReload=/bin/kill -s HUP $MAINPID
```

```
ExecStop=/bin/kill -s QUIT $MAINPID
```

```
PrivateTmp=true
```

```
[Install]
```

```
WantedBy=multi-user.target
```

Tengine安装和使用

□ 服务脚本

```
mkdir -p /etc/systemd/system/nginx.service.d
```

```
# vim /etc/systemd/system/nginx.service.d/override.conf
```

```
[Service]
```

```
ExecStartPost=/bin/sleep 0.1
```

□ 使用脚本

```
[root@localhost ~]# systemctl daemon-reload
```

```
[root@localhost ~]# systemctl start nginx
```

```
[root@localhost ~]# systemctl enable nginx
```

Welcome to tengine!

If you see this page, the tengine web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to tengine.taobao.org.

Thank you for using tengine.

Tengine配置文件

□ nginx.conf

```
[root@web01 conf]# cat nginx.conf
```

```
worker_processes 1;          ← worker 进程数量
events {                    ← 事件区块
    worker_connections 1024; ← 每个worker进程可以处理的连接数
}                             ← 事件区块结束
http {                      ← HTTP 区块
    include mime.types;     ← 支持的媒体文件
    default_type application/octet-stream; ← 默认的媒体类型
    sendfile on;            ← 高效传输模式
    keepalive_timeout 65;   ← 超时时间
    server {                ← server 区块
        listen 80;          ← 端口
        server_name localhost; ← 域名
        location / {        ← 第一个location区块
            root html;       ← 站点目录
            index index.html index.htm; ← 首页文件
        }                  ← 第一个location区块结束
        error_page 500 502 503 504 /50x.html; ← 错误信息配置
        location = /50x.html { ← 文件位置
            root html;       ← 在哪找：路径
        }
    }
}                             ← server 区块结束
                              ← HTTP 区块结束
```

静态页面

□ 编写静态访问页面文件信息

```
[root@localhost tengine]# cd /usr/local/tengine/html
```

```
[root@localhost html]# vim index.html
```

```
<html>
```

```
<meta charset="utf-8">
```

```
<head>
```

```
    <title>TEST Site</title>
```

```
</head>
```

```
<body>
```

测试页面

```
    <table border=1>
```

```
        <tr> <td>01</td> <td>云计算 </td> </tr>
```

```
        <tr> <td>02</td> <td>大数据</td> </tr>
```

```
        <tr> <td>03</td> <td>人工智</td> </tr>
```

```
    </table>
```

```
</body>
```

```
</html>
```


部署文件共享

□ 前提：首页文件不存在

通过配置 `autoindex on;` 参数
使用 `autoindex` 参数，nginx能识别的直接显示，不识别的直接
下载

配置完 `autoindex on;` 参数以后 会显示站点下的文件信息
对于nginx可以解析的资源会解析相应的内容
对于nginx不可以解析的资源会直接下载

```
location / {  
    root    html;  
    autoindex on; # 增加内容  
    index  index.html index.htm;  
}
```


部署文件共享

□ 前提：首页文件不存在

准备测试目录及文件：

```
[root@localhost html]# mkdir tools
[root@localhost html]# cd tools/
[root@localhost tools]# touch file{1..6}.txt
[root@localhost tools]# echo "hello world" > hello.txt
```

重启服务测试：

```
[root@localhost tools]# systemctl restart nginx
```

Index of /tools/

<hr/>		
../		
file1.txt	01-Nov-2019 09:21	0
file2.txt	01-Nov-2019 09:21	0
file3.txt	01-Nov-2019 09:21	0
file4.txt	01-Nov-2019 09:21	0
file5.txt	01-Nov-2019 09:21	0
file6.txt	01-Nov-2019 09:21	0
hello.txt	01-Nov-2019 09:21	12

虚拟主机

□ 常见的虚拟主机类型有如下几种

1) 基于域名的虚拟主机

所谓基于域名的虚拟主机，意思就是通过不同的域名区分不同的虚拟主机，基于域名的虚拟主机是企业应用最广的虚拟主机类型，几乎所有对外提供服务的网站使用的都是基于域名的虚拟主机，例如：www.znix.top。

2) 基于端口的虚拟主机

同理，所谓基于端口的虚拟主机，意思就是通过不同的端口来区分不同的虚拟主机，此类虚拟主机对应的企业应用主要为公司内部网站，例如：一些不希望直接对外提供用户访问的网站后台等，访问基于端口的虚拟主机，地址里要带有端口，例如：
http://blog.znix.top:80

3) 基于IP的虚拟主机

所谓基于IP的虚拟主机，意思是通过不同的IP区分不同的虚拟主机

虚拟主机

□ 基于域名虚拟主机示例

1) 配置文件添加虚拟主机部分

```
server {  
    listen      80;  
    server_name bbs.wanmen.com;  
    location / {  
        root    html/bbs;  
        index   index.html index.htm;  
    }  
}
```

```
server {  
    listen      80;  
    server_name blog.wanmen.com;  
    location / {  
        root    html/blog;  
        index   index.html index.htm;  
    }  
}
```

虚拟主机

□ 创建站点目录

```
[root@localhost tengine]# for name in blog bbs;do mkdir  
html/$name;done
```

□ 创建主页文件

```
[root@localhost tengine]# for name in blog bbs ;do echo " $name  
test" > html/$name/index.html ;done
```

□ 重启服务

```
[root@localhost tengine]# nginx -t  
[root@localhost tengine]# systemctl restart nginx
```

□ 测试

```
echo "192.168.95.12 blog.wanmen.com bbs.wanmen.com" >>  
/etc/hosts  
[root@localhost tools]# curl http://blog.wanmen.com  
blog test  
[root@localhost tools]# curl http://bbs.wanmen.com  
bbs test
```

状态模块

□ 修改配置文件，添加status模块

```
location /nginx_status {  
    stub_status on;  
    access_log off;  
}
```

□ 重新启动测试:

```
systemctl restart nginx
```

```
Active connections: 2  
server accepts handled requests request_time  
2 2 1 0  
Reading: 0 Writing: 1 Waiting: 1
```


状态模块

参数	参数说明
Active connections	当前的活动客户端连接数量
accepts	接受客户端连接的总数
handled	处理的连接总数
requests	客户端请求的总数
Reading	nginx正在读请求头的当前连接数。
Writing	nginx正在将响应写回客户端的当前连接数。
Waiting	当前空闲客户端连接数等待一个请求。

日志管理

□ nginx的两种日志种类

错误日志：记录nginx运行错误情况信息

访问日志：记录用户访问日志信息

官方说明：http://nginx.org/en/docs/nginx_core_module.html#error_log

1>配置错误日志。

系统默认配置

```
#error_log logs/error.log;
```

```
#error_log logs/error.log notice;
```

```
#error_log logs/error.log info;
```

本文源码安装，默认路径为/usr/local/tengine/logs

日志管理

□ nginx的两种日志种类

2>配置访问日志。

系统默认配置

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

```
#access_log logs/access.log main;
```

本文源码安装，默认路径为/usr/local/tengine/logs

日志管理

□ nginx的两种日志种类

2>配置访问日志。

访问日志说明：

```
[root@www ~]# tail -1 /usr/local/tengine/logs/access.log
192.168.95.12 - - [01/Nov/2019:17:45:12 +0800] "GET / HTTP/1.1"
200 10 "-" "curl/7.29.0"
```

\$remote_addr: 客户端地址

\$remote_user: 远程访问用户

[\$time_local]: 访问时间

\$request: 请求行信息

\$status: 状态码

\$body_bytes_sent: 响应报文主体内容大小

\$http_user_agent: 客户端浏览网页信息工具

\$http_x_forwarded_for: 反向代理转发

location

location 指令的作用是根据用户请求的URI来执行不同的应用。

locationn使用的语法为

```
location [=|~|~*|^~] uri {  
    ....  
}
```

location 语法说明表

location	[= ~ ~* ^~]	uri	{....}
指令	匹配标识	匹配的网站地址	匹配URI后要执行的配置段

~ 与~* 的区别

~ 匹配内容区分大小写

~* 匹配内容不区分的小写

!~ 取反

^~ 但多个匹配同时存在, 优先匹配 ^~ 匹配的内容;不做正则表达式的检查 (优先处理)

location

location 指令的作用是根据用户请求的URI来执行不同的应用。

不同uri及特殊字符组合匹配的顺序说明

顺序	不用URI及特殊字符组合匹配	匹配说明
1	location = / { }	精确匹配 /
2	location ^~ /image/{	匹配常规字符串，不做正则表达式匹配检查
3	location ~* \.(gif jpg jpeg)\$ {	正则匹配
4	location /documents/ {	匹配常规字符串，如果有正则，则优先匹配正则
5	location / {	所有location 都不能匹配后的默认匹配

location

□ 测试location 的访问

```
#location / {  
#    root    html;  
#    autoindex on;  
#    index  index.html index.htm;  
#}
```

```
location / {  
    return 401;  
}
```

```
location = / {  
    return 402;  
}
```

```
location /documents/ {  
    return 403;  
}
```

```
location ^~ /images/ {  
    return 404;  
}
```

```
location ~* \. (gif|jpg|jpeg) $ {  
    return 500;  
}
```


location

□ 测试location 的访问

访问测试:

```
[root@www conf]# curl -I -w "%{http_code}\n" -o /dev/null -s  
192.168.95.12/docuements  
401
```

```
[root@www conf]# curl -I -w "%{http_code}\n" -o /dev/null -s 192.168.95.12  
402
```

```
[root@www conf]# curl -I -w "%{http_code}\n" -o /dev/null -s  
http://192.168.95.12/documents/  
403
```

```
[root@www conf]# curl -I -w "%{http_code}\n" -o /dev/null -s  
192.168.95.12/images/a.jpg  
404
```

```
[root@www conf]# curl -I -w "%{http_code}\n" -o /dev/null -s  
192.168.95.12/docuements/abc.jpg  
500
```

rewrite地址重写

□ rewrite重写模块

rewrite 语法格式

rewrite regex replacement [flag]

rewrite应用标签: server、location、if

□ rewrite功能

1. 实现网站地址信息跳转

2. 实现伪静态

使用if:

```
if ($host ~* "^nmtui.com$") {  
    rewrite ^/(.*) http://www.nmtui.com/$1 permanent;  
}
```

server应用:

```
server {  
    server_name nmtui.com;  
    rewrite ^/(.*) http://www.nmtui.com/$1 permanent;  
}
```

rewrite地址重写

□ rewrite重写企业应用场景

可以调整用户浏览的URL，使其看起来更规范，合乎开发及产品人员的需求。为了让搜索引擎收录网站内容，并让用户体验更好，企业会将动态URL地址伪装成静态地址提供服务。

网站换新域名后，让旧域名的访问跳转到新的域名上，例如：让京东的360buy换成了jd.com。

根据特殊变量、目录、客户端的信息进行URL跳转等。

访问控制

□ 基于地址访问控制

ngx_http_access_module实现nginx访问控制， 内置

语法:

allow

语法: allow address | CIDR | unix: | all;

默认值: —

配置段: http, server, location, limit_except

deny

语法: deny address | CIDR | unix: | all;

默认值: —

配置段: http, server, location, limit_except

示例:

```
deny 192.168.95.11;  
allow 192.168.95.0/24;
```

访问控制

□ 基于用户访问控制

注意：使用的htpasswd命令默认是没有的，需要通过yum install httpd-tools -y 安装

配置：

```
auth_basic "Restricted";  
auth_basic_user_file /data/web2/webpass;
```

```
htpasswd /data/web2/webpass
```

Tengine优化

- 淘宝维护的nginx分支Tengine可以使用auto命令自动配置worker_processes和worker_cpu_affinity。

```
# grep worker_processes nginx.conf
worker_processes      auto;
```

- 调整nginx单个进程允许的客户最大连接数

```
events
{
    use epoll;
    worker_connections 51200;
}
```

#worker_connections 是一个事件模块指令,用于定义nginx每个进程的最大连接数,默认是1024.最大客户连接数由worker_processes和worker_connections决定,限max_client=worker_processes*work_connections.

进程的最大连接数受linux系统进程的最大打开文件数限制,在执行操作系统命令"ulimit -HSn 51200"或配置 相应文件后,worker_connections的设置才能生效.

Tengine优化

□ 开启高效文件传输模式

sendfile参数用于开启文件的高效传输模式.同时将tcp_nopush 和 tcp_nodelay两个指令设置 为on,可以防止网络和磁盘I/O阻塞,提升Nginx工作效率.

```
sendfile on;
```

```
tcp_nodelay on;
```

```
tcp_nopush on;
```

tcp_nodelay:默认情况下当数据发送时,内核并不会马上发送,可能会等待更多的字节组成一个数据包,这样可以提高I/O性能.但是,在每次只发送很少字节的务业场景中,使用tcp_nodelay功能,等待时间会比较长.

Tengine优化

□ nginx配置 gzip压缩功能

NGINX的GZIP压缩功依赖于:ngx_http_gzip_module模块(默认已安装)

对应压缩参数说明如下:

```
gzip on; #开启gzip压缩工能
gzip_vary on;#gzip_vary on
gzip_comp_level 6;#GZIP的压缩比列,1表示压缩最小,外理最
快;9表示压缩最大,传输速度最快,但处理慢,消耗CPU;
gzip_buffers 16 8k;#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。
16 8k代表以8k为单位, 安装原始数据大小以8k为单位的16倍申请内存。
gzip_min_length 1024;#允许压缩最小页面的字节数,页面字节数
从header头的content_Length中获取.默认值为0,表示不管页面有多大都
进行压缩.建议大于1KB.
gzip_proxied any;#无条件启用压缩
gzip_disable "msie6";#(IE5.5和IE6 SP1使用msie6参数来禁
止gzip压缩 )指定哪些不需要gzip压缩的浏览器(将和User-Agents进行
匹配),依赖于PCRE库
gzip_types text/plain text/css application/json
application/x-javascript text/xml application/xml
application/xml+rss text/javascript application/javascript;
```

总结

- Nginx简介
- Tengine简介
- Tengine配置文件
- Tengine虚拟主机
- Tengine安全优化

作业

- 1、安装部署Tengine，并配置服务脚本。
- 2、配置基于域名的虚拟主机。



谢谢观看

更多好课，请关注[万门大学APP](#)

