# 讲师介绍

Hash　QQ：805921455

从事Java软件研发近十年。

前新浪支付核心成员、

咪咕视讯(中国移动)项目经理、

对分布式架构、高性能编程有深入的研究。

**明天，你一定会感谢今天奋力拼搏的自己**

# Nginx从入门到上千机器负载均衡的搭建

## 分布式高并发—负载均衡

# 目录

## 课程安排

*01*

Nginx入门

Nginx相关概念、安装使用

*02*

官方文档阅读技巧

把握整体结构、从新手指南开始

*03*

实现高并发分流

基于反向代理的负载均衡策略

*04*

总结

本堂课程内容总结

# 目录


Nginx入门

# 高并发分流

**分流限流**、多读写少用缓存，写多读少用缓冲。

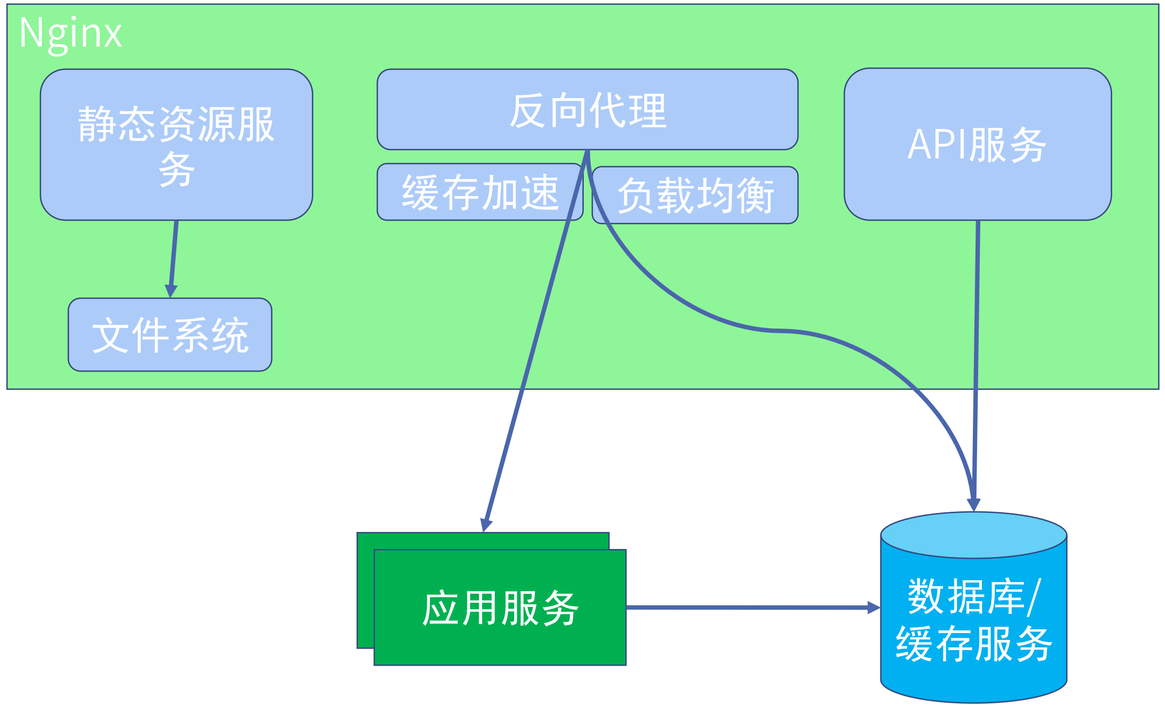
分流的技巧、分而治之的思想，将在这个专题完全展开。

# 什么是Nginx

一个俄国人用C语言编写的，开源的高性能的HTTP和反向代理服务软件。

Nginx主要应用于静态资源服务、反向代理服务、API服务

1. 静态资源主要借助于服务器本地文件系统来完成
2. 反向代理可以做到Nginx强大的性能、缓存、负载均衡
3. API服务通过集成nginx_lua模块来实现，比如OpenResty就是用ngixn和lua集成特性，整合了大量常用的第三方模块



# 正向代理/反向代理

正向代理，代表客户端进行网络或服务访问


反向代理，代表服务端接收客户端的请求

# 为什么会选Nginx？

在高并发的互联网行业，硬件负载均衡器、软件负载均衡器的选择

**高并发、高性能**

基于NIO非阻塞事件模型处理网络请求，slab内存管理机制。

**可扩展性好**

核心模块+扩展模块+第三方插件，丰富的模块及第三方插件是Nginx生命力顽强的原因。

**高可靠性**

部署后常年稳定运行

**热部署**

无需启动，更新配置文件

**开源BSD协议**

# 安装Nginx

## Windows版本

1. 进入Nginx官网下载页面
2. 选择nginx/Windows-xxx.zip的安装包下载，xxx表示最新的版本
3. 解压zip到你的程序安装目录
4. 进入nginx-xxx目录，双击nginx.exe启动Nginx服务

## Linux版本

1. 检查更新yum依赖
2. 添加yum的nginx仓库地址
3. 通过yum安装

更多具体操作细节，参考《Nginx安装手册》

# Nginx常规操作

Nginx常规操作

```
sudo ./nginx –h       # 显示帮助文档
sudo ./nginx -c /usr/local/nginx/conf/nginx.conf        #指定配置文件
sudo ./nginx -s quit         #比stop更优雅的退出
sudo ./nginx -s reload      # 热更新配置文件，无需启动
```

其他操作

```
sudo ./nginx -s reopen     # 打开新的日志文件
sudo ./nginx -t        # 测试配置文件是否正确
sudo ./nginx -v # 版本信息
sudo ./nginx -V # 版本信息及配置选项
```

# Nginx的配置文件

Nginx有一个非常强大的配置文件，可以配置应用对应的模块指令，类似Perl语法风格。配置文件指令主要分为两大块：简单指令、块指令。

**简单指令**

简单的指令由名称和参数组成，用空格分隔，以分号;结尾。

**块指令**

块指令，以大括号`{}`包围的一组附加指令，块指令在大括号内可以有其他指令，则称为上下文。

在任意上下文之外的指令称为主上下文，http、events指令放在主上下文，server在http中，location在server中。

# 动静分离—静态资源服务

**动静分离术**

Web服务当中，html、js、图片、css等静态资源放在webapp目录下。当静态资源越来越大时，一个网页获取大量静态资源时，影响到web服务的整体性能。

通过Nginx将静态资源独立部署，减轻web服务的压力。

动态计算的数据则通过web服务来获取。

```
worker_processes  1;        # 简单指令以;结尾
events {            # 大括号属于块指令
    worker_connections  1024;
}
http {              # http属于主上下文
    server {                # server在http的上下文中
    location / { # location在server上下文中
        root   /data/www/;
        index  welcome.html;
     }
}
}
```

通过实现一个静态资源服务，来学习配置文件指令的使用

启动Nginx，浏览器访问http://hostname:80

# Nginx的配置热更新

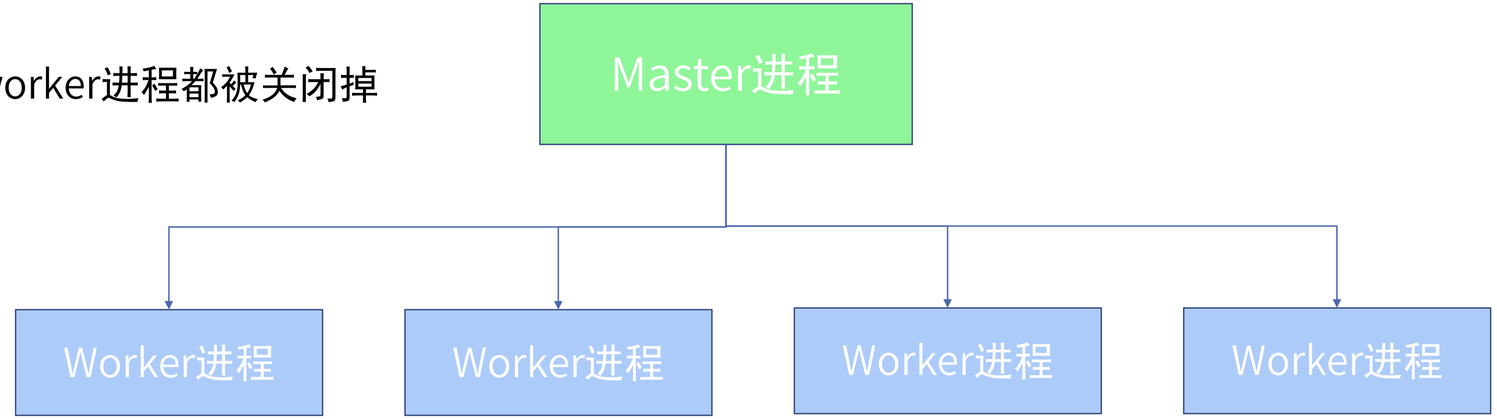
**Nginx如何做到配置文件热更新？**

1. master检查配置文件的正确性，若是错误则返回错误信息，nginx继续用原配置文件进行工作
2. Nginx启动新的worker进程，采用新的配置文件
3. Nginx将新的请求分配新的worker进程
4. Nginx等待以前的worker进程的全部请求已经都返回后，关闭相关worker进程
5. 重复上面过程，直到全部旧的worker进程都被关闭掉

Nginx中有Master、Worker两种进程。

Master进程负责加载配置、接收命令、监控子进程

Worker进程负责处理网络请求




```
ps -ef|grep nginx
root    12791    1  0 8月31 ?      00:00:00 nginx: master process ./nginx -c conf/nginx_proxy.conf
nobody   12792 12791  0 8月31 ?      00:00:00 nginx: worker process
```

# 目录


## 官方文档阅读技巧

# Nginx官方文档整体结构

文档主要有：首页、关于、下载页面、安全、使用文档、faq、博客等部分

about
download
security
documentation
faq
books
support

trac
twitter
blog

unit
njs

作为开发人员，主要关注，下载页面、使用文档、博客这几个部分。其中使用文档和博客是最为重要的内容。

**博客**中涉及一些高级的特性和商业支持。**使用文档**包含，Nginx常规操作介绍、功能场景操作、贡献提交源代码、Nginx模块参考。

**常规操作**

包含如何编译安装Nginx、初学者指南、管理员指南、Nginx操作、Nginx的连接处理事件类型、等信息。

**功能场景配置**

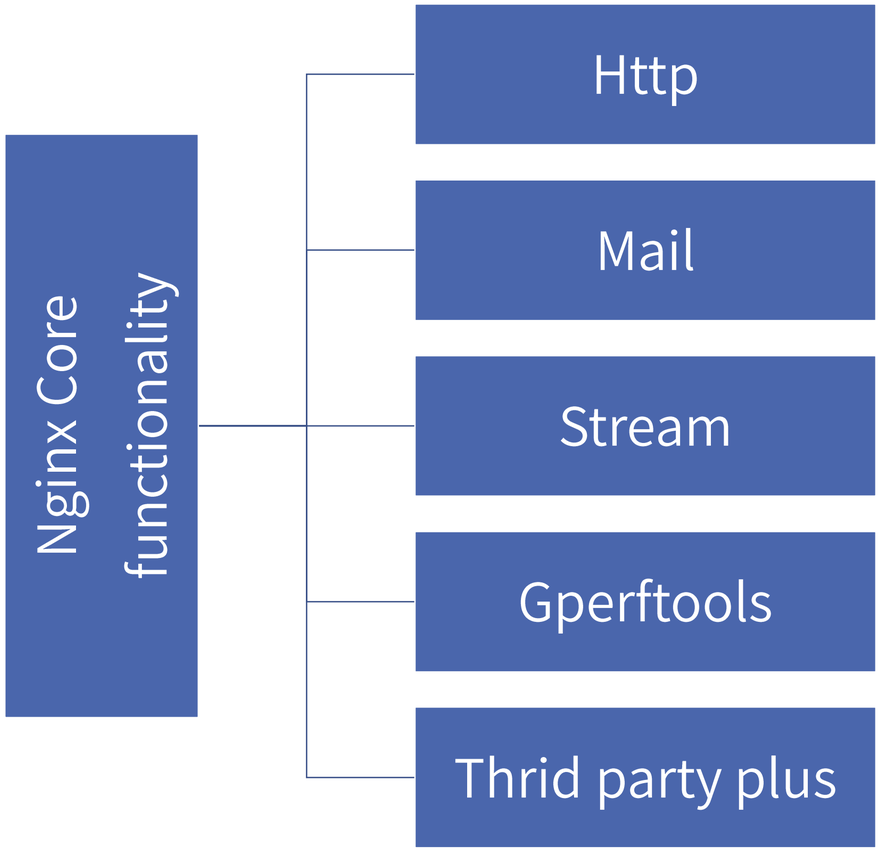
一般只有在对应的功能场景下使用配置，例如负载均衡反向代理、HTTPS服务

# Nginx模块

Nginx整体设计采用：**微核心模块+插件形式**

很多优秀的开源组件都在采用这样的设计思想，为第三方扩展提供了强大的支撑。

也是nginx受欢迎的原因之一。




用来处理实现七层协议的服务

用来实现邮件服务

处理四层协议服务

用来集成采集程序性能耗时数据

第三方插件模块

# 核心模块常用指令

核心功能指令

| 指令 | 描述 |
| --- | --- |
| user | 指定用来运行nginx的用户名，跟linux权限系统相关 |
| use | 指定处理网络IO的事件模型：select、kqueue、poll、epool |
| worker_processes | 指定运行多少个工作进程 |
| events | 指定连接处理相关的指令块：use、worker_connections等简单指令 |
| worker_connections | Worker进程能够处理的最大连接数 |

http核心指令

| 指令 | 描述 |
| --- | --- |
| http | 表示http模块的块指令 |
| srever | 表示一个虚拟服务 |
| listen | 监听端口号，有大量可选项，ssl、http2等 |
| location | 设置访问的URI，6种玩法 |

# location 7种玩法

最常用又最容易弄错的location指令，它可以由**前缀字符串**或**正则表达式**定义

| 方式 | 描述 |
| --- | --- |
| = | URI和位置的精确匹配，如找到完全匹配，则搜索终止，优先级最高 |
| /xxx | 字符串前缀匹配，匹配到xxx，记住，仍会继续匹配正则表达式 |
| ^~ | 字符串前缀匹配，有匹配，记住，但后继不再做正则表达式匹配，会使用最长前缀的匹配配置 |
| ~* | 不区分大小写的匹配正则表达式 |
| ~ | 区分大小写的匹配正则表达式 |
| / | 最后的接盘侠，所有指令都不匹配的时候则用它。相当于java中switch的default |
| @ | 非常规请求，用于请求重定向。自己不能嵌套，也不能嵌套location指令 |

前缀字符串

正则表达式

# location 匹配规则

当一个请求过来时候， location怎么进行匹配?

首先 nginx检查使用了前缀字符串定义的location指令块

选中并记住具有最长匹配前缀的location指令块

location如有^~匹配，则不查找正则表达式，使用该location

然后 按照它们在配置文件中的出现顺序检查正则表达式

正则表达式的搜索在第一次匹配时终止，并使用相应的配置

如果未找到与正则表达式的匹配，则使用先前记住的最长匹配前缀位置的配置

最后 还没有找到？ 接盘侠/

# 目录


## 实现高并发分流

# 单台服务的能力

**模拟一个耗时服务**

http://hostname:port/bench/1000

**使用压测工具压测**

```
# 总共10w次请求，100的并发
ab -n 100000 -c 100 http://localhost:8080/test/bench/1000
```

**测试结果**

```
# 平均吞吐率（总请求数/总时间），括号中的 mean 表示这是一个平均值
Requests per second:    197.98 [#/sec] (mean)
```

# Nginx负载均衡

负载均衡涉及的模块

## Module ngx_http_upstream_module

Example Configuration
Directives
- upstream
- server
- zone
- state
- hash
- ip_hash
- keepalive
- keepalive_requests
- keepalive_timeout
- ntlm
- least_conn
- least_time
- queue
- random
- sticky
- sticky_cookie_insert

Embedded Variables

http://nginx.org/en/docs/http/ngx_http_upstream_module.html


## Module ngx_http_proxy_module

Example Configuration
Directives
- proxy_bind
- proxy_buffer_size
- proxy_buffering
- proxy_buffers
- proxy_busy_buffers_size
- proxy_cache
- proxy_cache_background_update
- proxy_cache_bypass
- proxy_cache_convert_head
- proxy_cache_key
- proxy_cache_lock
- proxy_cache_lock_age
- proxy_cache_lock_timeout
- proxy_cache_max_range_offset
- proxy_cache_methods
- proxy_cache_min_uses
- proxy_cache_path
- proxy_cache_purge
- proxy_cache_revalidate
- proxy_cache_use_stale

http://nginx.org/en/docs/http/ngx_http_proxy_module.html

# Nginx完成反向代理负载均衡

## upstream

http_upstream模块定义一组服务，能被proxy_pass指令引用。

upstream name { ... }　　在http指令块中，定义一组可以进行负载均衡的上游服务器，并给取个组名。服务可以是不同的端口、TCP、Unix域名套接字。

```
upstream tomcatServer {
    server backend1.example.com weight=5;
    server 127.0.0.1:8080       max_fails=3 fail_timeout=30s;
    server unix:/tmp/backend3;
    server backup1.example.com  backup;
}
```

通过upstream，就能为成千上百台的后端服务集群提供强有力的支撑

# Nginx完成反向代理负载均衡

proxy_pass

表示将请求传递到另外一个服务

```
# server在http的上下文中
    server {
        # location在server上下文中
        location / {
            proxy_pass http://tomcatServer;
            health_check;
        }
    }
```

# 再次压测

## 修改Nginx配置并启动

参考示例配置文件

## 使用压测工具压测

```
# 总共10w次请求，100的并发
ab -n 100000 -c 100 http://localhost:8080/test/bench/1000
```

## 测试结果

```
?
```

# Nginx负载均衡策略

| 策略 | 均衡性 | 一致性 | 容灾性 | 描述 |
| --- | --- | --- | --- | --- |
| round-robin | ★★★ | ★ | ★★★ | 默认的处理反向代理服务的方法，还能根据加权值来传递请求。<br>通用性强，无特殊需求的场景下均可 |
| hash | ★★★ | ★★★ | ★★ | 请求传递根据key的hash值决定，key可以包含文本、变量、文本变量的组合。有consistent参数表示开启一致性hash算法。<br>请求一致性，例如缓存服务器 |
| ip_hash | ★★ | ★★★ | ★★★ | 根据客户端ip地址在服务间进行分发请求。要求ip一致性的场景 |
| random | ★★ | ★ | ★★ | 将请求传递给随机选择的服务器 |
| least_conn | ★★★ | ★ | ★★★ | 将请求传递到活动连接数量最少的服务器，同时考虑服务器的权重。自适应强，适合服务差异复杂的情况 |
| least_time | ★★ | ★ | ★★★ | 将请求传递给具有最小平均响应时间和最少活动连接数的服务，同时考虑服务器权重。<br>自适应强，适合网络环境复杂的情况 |

# 目录

04

总结

# 总结

Nginx的入门

安装、核心功能、核心模块

配置文件、指令结构、Nginx的进程

Nginx官方文档阅读技巧

整体结构、设计思想及模块结构、起航

负载均衡实战

upsteam、proxy模块

ab测试工具

负载均衡策略

# 谢谢观看