

 **愿景："让编程不再难学，让技术与生活更加有趣" 更多架构课程请访问 xdclass.net**

# 第一章 千万级连接-高性能服务器 Nginx

**专题课程介绍**

## 第1集 千万级连接高性能服务器Nginx专题课程介绍

### 简介：讲解高性能Nginx专题课程介绍，课程适合人员和学后水平

课程介绍

千万级连接-高性能服务器 Nginx专题课程是针对中高级工程师而定制的，从零基础讲解反向代理服务器到阿里云ECS+本地虚拟机部署Nginx，玩转核心配置和面试题，超多经典案例实战：

1. 搭建前端静态资源服务器、文件服务器、BAT大厂自研运维平台数据统计案例
2. 负载均衡Upstream配置实战、后端节点高可用性探测、全局异常兜底数据配置
3. Nginx封禁恶意IP、配置跨域、location和rewrite实战4）Websocket配置、后端业务数据缓存前置、静态资源压缩

5）阿里云ECS部署Nginx + HTTPS证书

6）高级拓展Nginx整合Openresty开发内网访问限制、文件资源下载限速实现原理

7) Nginx高可用解决访问 LVS + KeepAlived讲解+多节点配置实操



为什么要学Nginx课程

据统计，世界上每3个网站中就有一个使用Nginx

市场上使用情况：阿里(Tnginx)、腾讯、百度等，全球反向代理服务器中排名

前端-后端-架构-运维基本都离不开Nginx

适合人群

后端开发工程师-中高级工程师前端开发工程师-中高级工程师技术leader或者架构师

测试-运维工程师

课程技术技术栈和测试环境说明

统一新版环境，Nginx-1.18版本 + openresty-1.17.8.2

+ LVS

操作系统：阿里云Linux Centos7 64位 + 本地虚拟机

Centos7 64位

学后水平

零基础掌握反向代理Nginx核心知识掌握案例实战+原理分析+面试题

搭建前端静态资源服务器、文件服务器、BAT大厂自研运维平台数据统计案例

负载均衡Upstream配置实战、后端节点高可用性探

测、全局异常兜底数据配置

Nginx封禁恶意IP、配置跨域、location和rewrite实战

Websocket配置、后端业务数据缓存前置、静态资

源压缩

阿里云ECS部署Nginx + HTTPS证书

高级拓展Nginx整合Openresty开发内网访问限制、文件资源下载限速实现原理

Nginx高可用解决访问 LVS + KeepAlived讲解+多节

点配置实操

Nginx架构Master-Work多进程模型



学习形式

视频讲解 + 文字笔记 + 原理分析 + 交互流程图

配套源码 + 笔记 + 技术群答疑( 我答疑，联系客服进群) 只要是我的课程和项目-我会一直维护下去，大家不用担心！！！

## 第2集 高性能Nginx服务器专题课程大纲速览

### 简介：讲解Nginx高性能服务器课程大纲

课程学前基础

有Linux就行，不会的话我们有专题课程

目录大纲浏览

学习要求：

课程有配套资料，如果有用到就会在对应章集的资料里面，如果自己操作的情况和视频不一样，仔细对比对比验证基本就可以发现问题了

Nginx比较坑的点，就是配置多，然后没有比较好的编

辑器，所以遇到和课程效果不一样的，建议看多几次结合笔记里面的配置，网上的博文

很多错误配置，就是配置少加了 英文分号 ; 或者用了中

文的； 或者防火墙问题，或者配置多了空格等特殊字符。

保持谦虚好学、术业有专攻、在校的学生，有些知识掌握

也比工作好几年掌握的人厉害

 **愿景："让编程不再难学，让技术与生活更加有趣" 更多架构课程请访问 xdclass.net**

# 第二章 高性能Nginx服务器快速入门

## 第1集 快速认知什么是Nginx服务器

### 简介：什么是Nignx服务器

Nginx介绍

官网：<http://nginx.org/>

是一个高性能的[HTTP]和[反向代理]web服务器

Nginx代码完全用[C语言]从头写成系统：Mac/Windows/Linux

市场上使用情况

阿里、腾讯、百度等，全球反向代理服务器中排名据统计，世界上每3个网站中就有一个使用Nginx

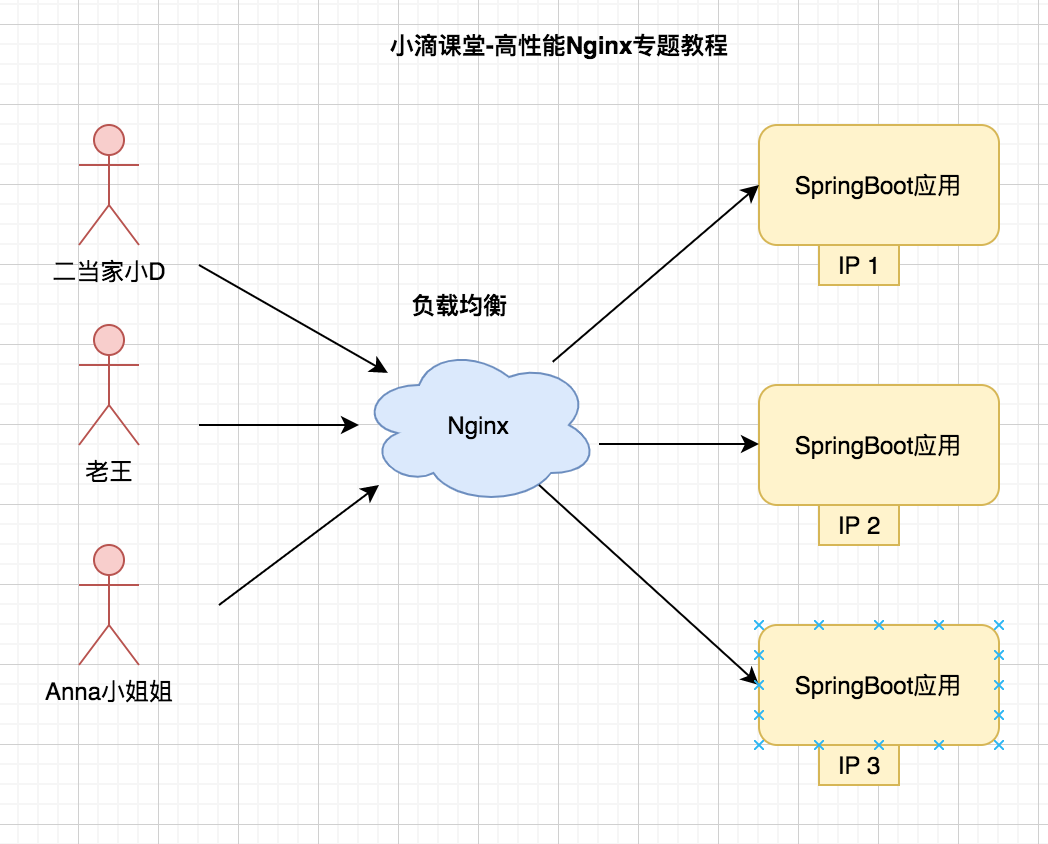
为什么要用这个

社区活跃

高性能-支持单机千万级连接强大的第三方库支持

功能强大：负载均衡、静态文件服务器、支持多种协议

https、POP3等等



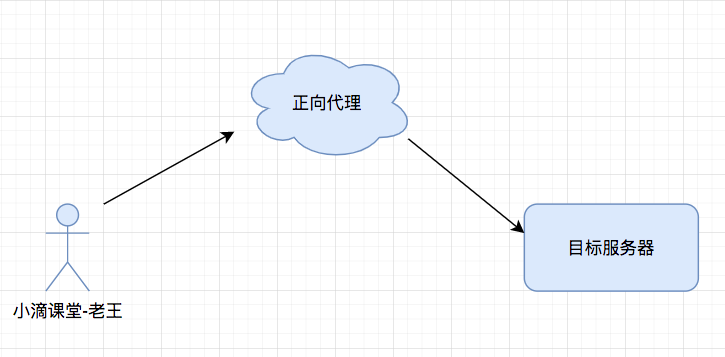
## 第2集 【面试题】什么是反向代理服务器

### 简介：什么是正向代理服务器和反向代理服务器

正向代理服务器

客户端和目标服务器之间的服务器，客户端向代理发送一个请求指定目标服务器，然后代理向目标服务器请求并获得内容，并返回给客户端，平时说的代理服务器一般是正向代理服务器

核心：用户知道自己访问的目标服务器

场景：跳板机、访问原来无法访问的网站, 比如国外的一些站点

反向代理服务器（Nginx）

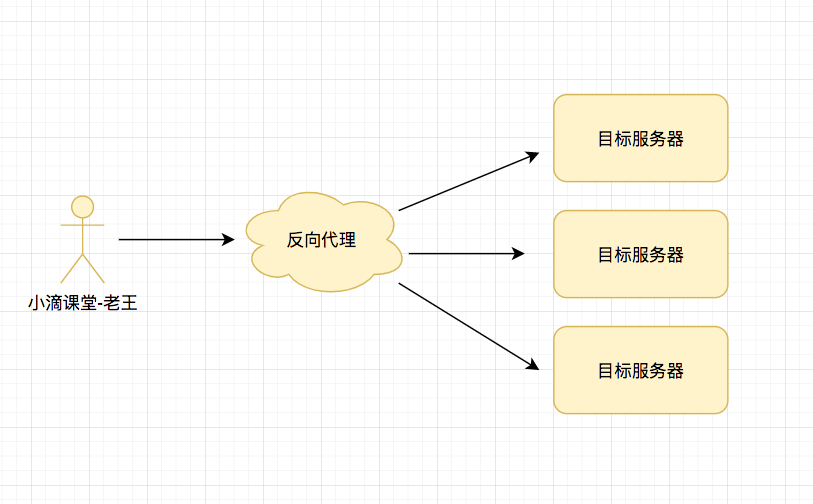
客户端和目标服务器之间的服务器，客户端向代理发送一个请求，然后代理向目标服务器请求并获得内容，并

返回给客户端。反向代理隐藏了真实的服务器

核心：客户端不知道要访问的目标服务器是哪台服务 器，代理会根据一定的策略选择一个真实的服务器进行请求

场景：访问淘宝，知道访问的域名是taobao.com， 但

是后面提供数据的具体是什么域名或ip我们是不知道的



 **愿景："让编程不再难学，让技术与生活更加有趣" 更多架构课程请访问 xdclass.net**

# 第三章 阿里云CentOS服务器-安装部署

**Nginx**

## 第1集 云服务器介绍和阿里云服务器ECS服务器选购

### 简介：什么是云服务器及目前主要的几个厂商介绍

云厂商

阿里云：<https://www.aliyun.com/>腾讯云：<https://cloud.tencent.com/>

亚马逊云：<https://aws.amazon.com/>

阿里云新用户地址（如果地址失效，联系我或者客服即可，1折）

[https://www.aliyun.com/minisite/goods?userCod e=r5saexap&share\_source=cop y\_link](https://www.aliyun.com/minisite/goods?userCode=r5saexap&share_source=copy_link)

环境问题说明

Win7、Win8、Win10、Mac、虚拟机等等，可能存在兼容问题

务必使用CentOS 7 以上版本，64位系统，不要在

Windows系统操作！！！！

这章统一使用阿里云服务器，部分功能阿里云不支持， 下一章再用本地虚拟机进行部署

选购实操

## 第2集 阿里云服务器远程登录和常用工具

### 简介：讲解阿里云服务器登录使用和常见终端工具

备注：(服务器、域名等使用你们自己购买的哈，上面有提供低价购买链接，失效找我)

[阿里云新用户地址 https://www.aliyun.com/minisite/go ods?userCode=r5saexap&share\_source=cop y\_link](https://www.aliyun.com/minisite/goods?userCode=r5saexap&share_source=copy_link)

控制台修改阿里云远程连接密码

windows工具 putty，xshell, security CRT

参考资料：

[https://jingyan.baidu.com/article/e75057f210c6d cebc91a89dd.html](https://jingyan.baidu.com/article/e75057f210c6dcebc91a89dd.html) <https://www.jb51.net/softjc/88235.html>

苹果系统MAC ： 通过终端登录

ssh root@ip 回车后输入密码

ssh [root@120.24.216.117](mailto:root@120.24.216.117)

linux图形操作工具（用于远程连接上传文件） mac: filezilla

sftp://120.24.216.117

windows: winscp

[参考资料：https://jingyan.baidu.com/article/ed2a5d 1f346fd409f6be179a.html](https://jingyan.baidu.com/article/ed2a5d1f346fd409f6be179a.html)

可以尝试自己通过百度进行找文档， 安装mysql jdk nginx maven git redis等，也可以看我们的课程

## 第3集 阿里云Linux服务器上Nginx安装

### 简介：nginx下载和阿里快速安装

下载压缩包 并上传

<http://nginx.org/en/download.html>

安装依赖

yum -y install gcc zlib zlib-devel pcre-devel openssl openssl-devel

创建一个文件夹，上传本地提供的nginx包

tar -zxvf nginx-1.18.0.tar.gz

//执行命令

./configure make

make install

默认安装路径

/usr/local/nginx

访问配置

cd /usr/local/nginx/sbin

./nginx

防火墙开放端口，阿里云网络安全组配置80端口

## 第4集 域名备案和线上访问服务器应用流程解析

### 简介：讲解应用部署到可以公网访问需要的知识

一个http请求基本流程

客户端通过发起域名资源请求 -> DNS解析获得IP -> 寻找服务器获得资源

域名和ip的关系，DNS作用

DNS：Domain Name Server 域名服务器 域名虽然便于人们记忆，但网络中的计算机之间只能互相认识IP地址，它们之间的转换工作称为域名解析，域名解析需要由专门的域名解析服务器来完成，DNS 就是进行域名解析的服务器

什么是cname和a记录

a记录

用户可以在此设置域名并指向到自己的目标主机地址上，从而实现通过域名找到服务器（也叫ip指向域名配置）aabbcc.com -> 120.24.216.117

cname

别名指向，可以为一个主机设置别名。比如设置open1024.com，用来指向一个主机 xdclass.net 那么以后就可以用open1024.com来代替访问xdclass.net 了

[www.xdclass.net](http://www.xdclass.net/) --> xdclass.net

购买服务器，阿里云，腾讯云，亚马逊云aws 购买域名，备案

阿里云 备案地址：<https://beian.aliyun.com/>

安装项目依赖的基本环境，比如java、nginx等软件(看项目)

配置域名解析到服务器，nginx是80端口，我们解析到80 端口就行

【注意】

备案需要1~2个工作日，推荐使用这个方式，掌握上线 流程

实在不行，则去网上找虚拟机教程，本地搭建虚拟机，

然后看下一章视频进行配置

 **愿景："让编程不再难学，让技术与生活更加有趣" 更多架构课程请访问 xdclass.net**

# 第四章 本地虚拟机CentOS服务器-安装部署Nginx

## 第1集 本地虚拟机环境说明和注意事项

### 简介: 本地虚拟机环境介绍和注意事项

如果阿里云备案方式不行，自行找网上的虚拟机搭建，我们这边不专门讲虚拟机和各个配置(太耗时)

搜索 系统 + 虚拟机安装 关键词看博文

虚拟机系统版本要求：CentOS 7以上版本 64位 的镜像

网络访问：需要宿主机和虚拟机网络可以访问，防火墙关闭，或者开放80端口

安装部署和阿里云上一致

下载压缩包 并上传安装依赖

yum -y install gcc zlib zlib-devel pcre-devel

openssl openssl-devel

创建一个文件夹，上传本地提供的nginx包

tar -zxvf nginx-xx.xx.xx.tar.gz （注意替换版本号）

//执行命令

./configure make

make install

默认安装路径

/usr/local/nginx

访问配置

cd /usr/local/nginx/sbin

./nginx

防火墙开放端口，阿里云网络安全组配置80端口

## 第2集 本地域名映射虚拟机IP

### 简介:讲解本地域名映射虚拟机IP

本地域名映射

浏览器输入个域名，经过DNS解析获取IP，如果我们没有公网域名和IP，怎么去测试这块内容？

DNS解析，会先获取本地的Host文件，先看本地有没域名>IP 的映射，有的话直接使用

Windows :

[https://www.cnblogs.com/spirit-ling/p/8646895.ht ml](https://www.cnblogs.com/spirit-ling/p/8646895.html) [https://jingyan.baidu.com/article/5bbb5a1b15c97c1 3eba1798a.html](https://jingyan.baidu.com/article/5bbb5a1b15c97c13eba1798a.html)

Mac苹果

cd /private/etc sudo vim hosts

 **愿景："让编程不再难学，让技术与生活更加有趣" 更多架构课程请访问 xdclass.net**

# 第五章 Nginx核心基础知识讲解

## 第1集 Nginx目录文件讲解

### 简介：Nginx目录文件讲解

源码编译安装后，默认目录

/usr/local/nginx

目录核心介绍

conf #所有配置文件目录

nginx.conf #默认的主要的配置文件nginx.conf.default #默认模板

html # 这是编译安装时Nginx的默认站点目录

50x.html #错误页面index.html #默认首页

logs # nginx默认的日志路径，包括错误日志及访问日志error.log #错误日志

nginx.pid #nginx启动后的进程id

access.log #nginx访问日志

sbin #nginx命令的目录

nginx #启动命令

常见命令

./nginx #默认配置文件启动

./nginx -s reload #重启，加载默认配置文件

./nginx -c /usr/local/nginx/conf/nginx.conf #启动指定某个配置文件

./nginx -s stop #停止

#关闭进程，nginx有master process 和worker process,关闭master即可

ps -ef | grep "nginx" kill -9 PID

## 第2集 Nginx核心之配置文件剖析

### 简介：Nginx核心知识之默认配置文件讲解

全局配置

server 主机设置

location（URL匹配特定位置的设置）

# 每个配置项由配置指令和指令参数 2 个部分构成

#user

组

nobody;

# 指定Nginx Worker进程运行以及用户

worker\_processes 1;

#

志的存放路径

|  |  |  |
| --- | --- | --- |
| #error\_log | logs/error.log; | # 错误日 |
| 和错误日志 |  |  |
| #error\_log | logs/error.log | notice; |
| #error\_log | logs/error.log | info; |

#pid logs/nginx.pid; # 进 程 PID 存 放 路 径

# 事件模块指令，用来指定Nginx的IO模型，Nginx支持的有select、poll、kqueue、epoll 等。不同的是epoll用在Linux平台上，而kqueue用在BSD系统中，对于Linux系统， epoll工作模式是首选

events {

use epoll;

# 定义Nginx每个进程的最大连接数， 作为服务器来说: worker\_connections \* worker\_processes,

# 作为反向代理来说，最大并发数量应该是

worker\_connections \* worker\_processes/2。因为反向代理服务器，每个 并发会建立与客户端的连接和与后端服务的连接，会占用两个连接

worker\_connections 1024;

}

http {

include mime.types;

default\_type application/octet-stream;

# 自定义服务日志

#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;

# 是否开启高效传输模式 on开启 off关闭sendfile on;

#减少网络报文段的数量#tcp\_nopush on;

#keepalive\_timeout 0;

# 客户端连接保持活动的超时时间，超过这个时间之后， 服务器会关闭该连接

keepalive\_timeout 65; #gzip on;

# 虚拟主机的配置

server {

listen 80; # 虚拟主机的服务端口

server\_name localhost; #用来指定IP地址或域名，多个域名之间用空格分开

#charset koi8-r;

#access\_log logs/host.access.log

main;

#URL地址匹配location / {

root html; # 服务默认启动目录

index index.html index.htm; #默认访问文件，按照顺序找

}

#error\_page 404

/404.html; #错误状态码的显示页面

# redirect server error pages to the static page /50x.html

#

error\_page 500 502 503 504

/50x.html;

location = /50x.html { root html;

}

# proxy the PHP scripts to Apache listening on 127.0.0.1:80

#

#location ~ \.php$ {

# proxy\_pass http://127.0.0.1;

#}

# pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000

#

#location ~ \.php$ {

# root html;

# fastcgi\_pass 127.0.0.1:9000; # fastcgi\_index index.php;

# fastcgi\_param SCRIPT\_FILENAME

/scripts$fastcgi\_script\_name;

# include fastcgi\_params; #}

# deny access to .htaccess files, if Apache's document root

# concurs with nginx's one #

#location ~ /\.ht { # deny all;

#}

}

# another virtual host using mix of IP-, name-, and port-based configuration

#

#server {

# listen 8000;

# listen somename:8080; # server\_name somename alias

another.alias;

|  |  |  |
| --- | --- | --- |
| #  # # | location /  root index | {  html;  index.html index.htm; |
| # | } |  |
| #} |  |  |

# HTTPS server #

#server {

# listen 443 ssl; # server\_name localhost;

# ssl\_certificate cert.pem; # ssl\_certificate\_key cert.key;

# ssl\_session\_cache shared:SSL:1m; # ssl\_session\_timeout 5m;

# ssl\_ciphers HIGH:!aNULL:!MD5; # ssl\_prefer\_server\_ciphers on;

# location / {

# root html;

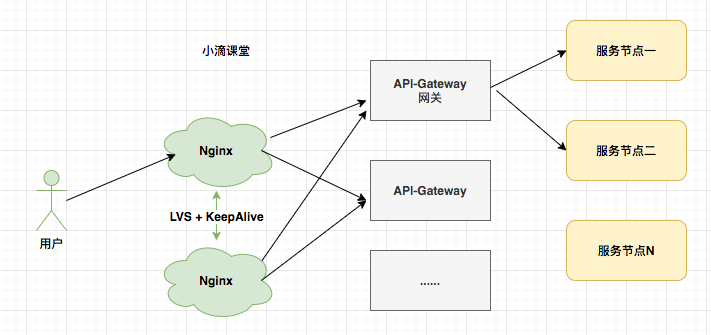
# index index.html index.htm;

#

#}

}

}



## 第3集 玩转Nginx虚拟主机-搭建前端静态服务器

### 简介 玩转Nginx配置虚拟主机，搭建前端静态资源服务器

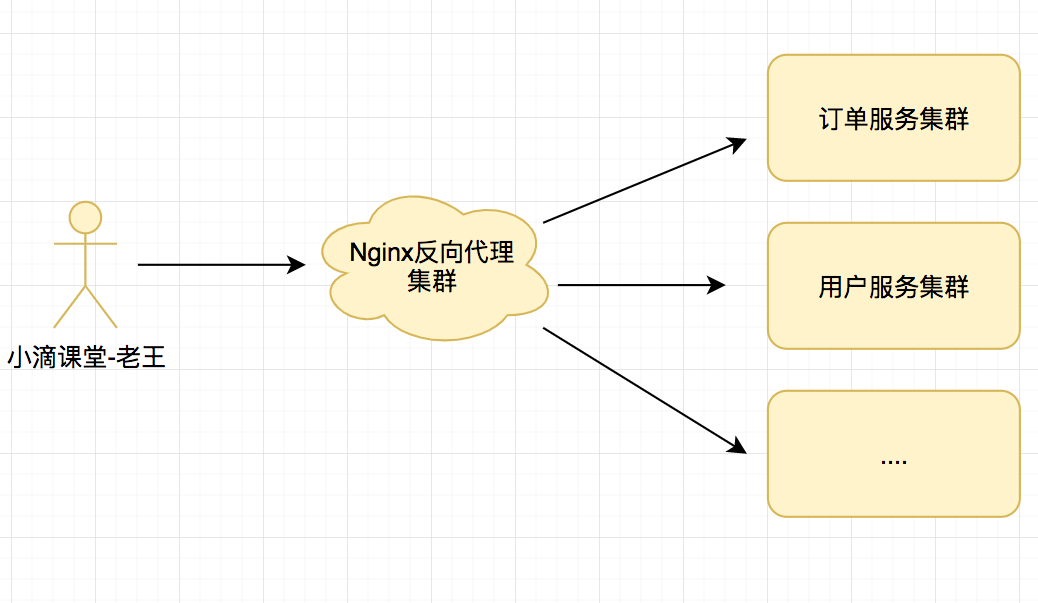
什么是虚拟主机

指在一台物理主机服务器上划分出多个磁盘空间，每个磁盘空间都是一个虚拟主机，每台虚拟主机都可以对外提供Web服务，并且互不干扰，就类似虚拟机

利用虚拟主机把多个不同域名的网站部署在同一台服务器上，节省了服务器硬件成本和相关的维护费用

【注意】配置之前，拷贝一份默认的出来，nginx配置容

易出错且肉眼难看出来，自己仔细核对



Nginx虚拟主机配置

server {

listen 80;

server\_name aabbcc.com;

location / {

root /usr/local/nginx/html; index xdclass.html;

}

}

server {

listen 80;

server\_name aabbccdd.com;

location / {

root html;

index xdclass.html index.htm;

}

}

## 第4集 案例实战之配置文件-图片服务器

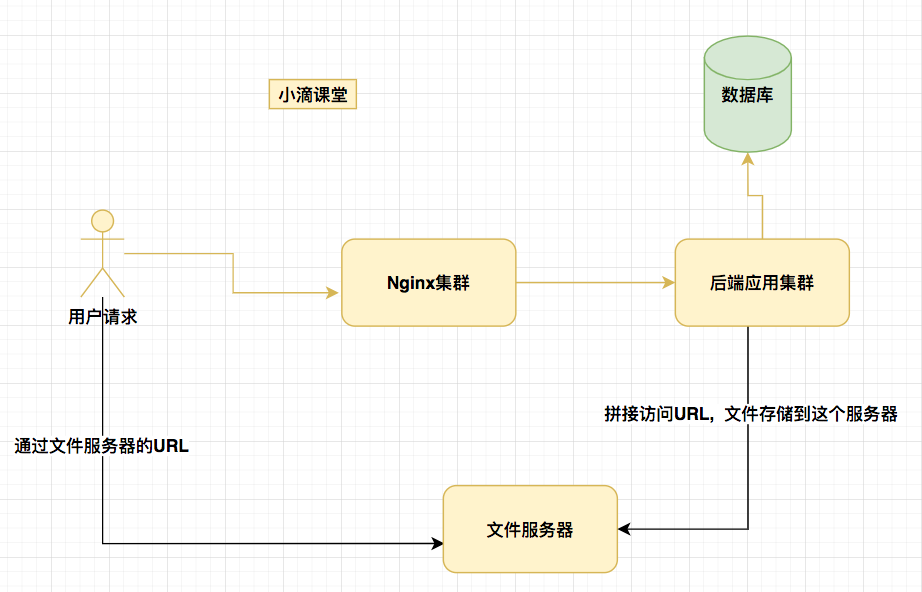
### 简介：使用nignx搭建图片-文件服务器

图片服务器

学javaweb、node、或者其他基础web项目，基本都是图片上传到项目本身，这个是生成很少用

公司一般会使用图片服务器或者云厂商提供的CDN

使用流程

前端提交图片->后端处理->存储到图片服务器->拼接好访 问路径存储到数据库和范围前端

本地图片上传上去，配置专属访问路径

server {

listen 80; server\_name aabbccdd.com; location /app/img {

alias /usr/local/software/img/;

}

}

注意

在location / 中配置root目录

在location /path中配置alias虚拟目录， 目录后面的"/"符号一定要带上

 **愿景："让编程不再难学，让技术与生活更加有趣" 更多架构课程请访问 xdclass.net**

# 第六章 Nginx经典案例挖掘accessLog

**日志**

## 第1集 你知道的Nginx访问日志的用处不？

### 简介：介绍nginx的访问日志的用处

access.log日志用处

统计站点访问ip来源、某个时间段的访问频率

查看访问最频的页面、Http响应状态码、接口性能接口秒级访问量、分钟访问量、小时和天访问量

...

默认配置解析

#log\_format

$remote\_user #

main

'$remote\_addr -

[$time\_local] "$request" '

'$status $body\_bytes\_sent

"$http\_referer" '

# '"$http\_user\_agent" "$http\_x\_forwarded\_for"';

案例

122.70.148.18 - - [04/Aug/2020:14:46:48

+0800] "GET

/user/api/v1/product/order/query\_state? product\_id=1&token=xdclasseyJhbGciOJE HTTP/1.1" 200 48 "https://xdclass.net/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.89 Safari/537.36"

解析

$remote\_addr 对应的是真实日志里的122.70.148.18， 即客户端的IP。

$remote\_user 对应的是第二个中杠“-”，没有远程用户， 所以用“-”填充。

［$time\_local］对应的是[04/Aug/2020:14:46:48

+0800]。

“$request”对应的是"GET

/user/api/v1/product/order/query\_state? product\_id=1&token=xdclasseyJhbGciOJE HTTP/1.1"。

$status对应的是200状态码，200表示正常访问。

$body\_bytes\_sent对应的是48字节，即响应body的大小。

“$http\_referer” 对应的

是”https://xdclass.net/“，若是直接打开域名浏览的时，referer就会没有值，为”-“。

“$http\_user\_agent” 对应的是”Mozilla/5.0

(Macintosh; Intel Mac OS X 10.12; rv:56.0) Gecko/20100101 Firefox/56.0”。

“$http\_x\_forwarded\_for” 对应的是”-“或者空。

## 第2集 案例实战之BAT大厂应用运维平台案例统计

### 简介：Nginx统计站点访问量、高频url统计

查看访问最频繁的前100个IP

awk '{print $1}' access\_temp.log | sort -n

|uniq -c | sort -rn | head -n 100

统计访问最多的url 前20名

cat access\_temp.log |awk '{print $7}'| sort|uniq -c| sort -rn| head -20 | more

基础

awk 是文本处理工具，默认按照空格切分，$N 是第切割后

第N个，从1开始

sort命令用于将文本文件内容加以排序，-n 按照数值排，- r 按照倒序来排

案例的sort -n 是按照第一列的数值大小进行排序，从

小到大，倒序就是 sort -rn

uniq 去除重复出现的行列, -c 在每列旁边显示该行重复

出现的次数。

## 第3集 案例实战之自定义日志统计接口性能

### 简介 自定义日志格式，统计接口响应耗时

日志格式增加 **$request\_time**

从接受用户请求的第一个字节到发送完响应数据的时间，即

包括接收请求数据时间、程序响应时间、输出响应数据时间

$upstream\_response\_time：指从Nginx向后端建立连接

开始到接受完数据然后关闭连接为止的时间

$request\_time一般会比upstream\_response\_time

大，因为用户网络较差，或者传递数据较大时，前者会耗时大很多

配置自定义日志格式

log\_format main '$remote\_addr -

$remote\_user [$time\_local] "$request" '

'$status

$body\_bytes\_sent "$http\_referer" '

'"$http\_user\_agent" "$http\_x\_forwarded\_for" $request\_time';

server {

listen 80; server\_name aabbcc.com;

location / {

root /usr/local/nginx/html; index xdclass.html;

}

#charset koi8-r; #

access\_log logs/host.access.log

main;

}

统计耗时接口, 列出传输时间超过 2 秒的接口，显示前5条

cat time\_temp.log|awk '($NF > 2){print

$7}'|sort -n|uniq -c|sort -nr|head -5

备注：$NF 表示最后一列, awk '{print $NF}'

 **愿景："让编程不再难学，让技术与生活更加有趣" 更多架构课程请访问 xdclass.net**

# 第七章 高级篇-Nginx配置集群应用-负载均衡策略

## 第1集 Linux服务器安装JDK8和jar包上传

### 简介：Linux服务器安装JDK8和jar上传启动

安装JDK8环境

[官方地址：http://www.oracle.com/technetwork/jav a/javase/downloads/jdk8-downloads-2133151.html](http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html)

本地上传JDK1.8到服务器（课程提供安装包，对应章集的资料里面）

配置全局环境变量

解压：tar -zxvf jdk-8u171-linux-x64.tar.gz

重命名

vim /etc/profile

配置

JAVA\_HOME=/usr/local/software/jdk/jdk8 CLASSPATH=$JAVA\_HOME/lib/ PATH=$PATH:$JAVA\_HOME/bin

export PATH JAVA\_HOME CLASSPATH

环境变量立刻生效

source /etc/profile

查看安装情况 java -version

上传jar包

启动应用关闭应用

lsof -i:port 或者 ps -ef | grep java

kill -9 PID

## 第2集 后端应用集群构建-SpringBoot应用和接口说明

### 简介：集群构建准备-springboot应用和接口说明

Linux服务器需要安装JDK8

准备两个一样的Jar包demo-1.jar监听8080端口demo-2.jar监听8081端口

接口说明

接口一

GET请求，返回json数据，控制输出日志http://127.0.0.1:8080/api/v1/pub/info/check

接口二

返回HTML页面，两个jar返回的HTML内容不一样， 方 便 区 分 访 问 的 是 哪 个 jar http://localhost:8080/api/v1/pub/web

直接启动

java -jar demo-1.jar java -jar demo-2.jar

守护进程方式

nohup java -jar demo-1.jar & nohup java -jar demo-2.jar &

## 第3集 Nginx负载均衡upstream讲解

### 简介：Nginx的upstream模板介绍

负载均衡（Load Balance）

分布式系统中一个非常重要的概念，当访问的服务具有多个实例时，需要根据某种“均衡”的策略决定请求发往哪个节点，这就是所谓的负载均衡，

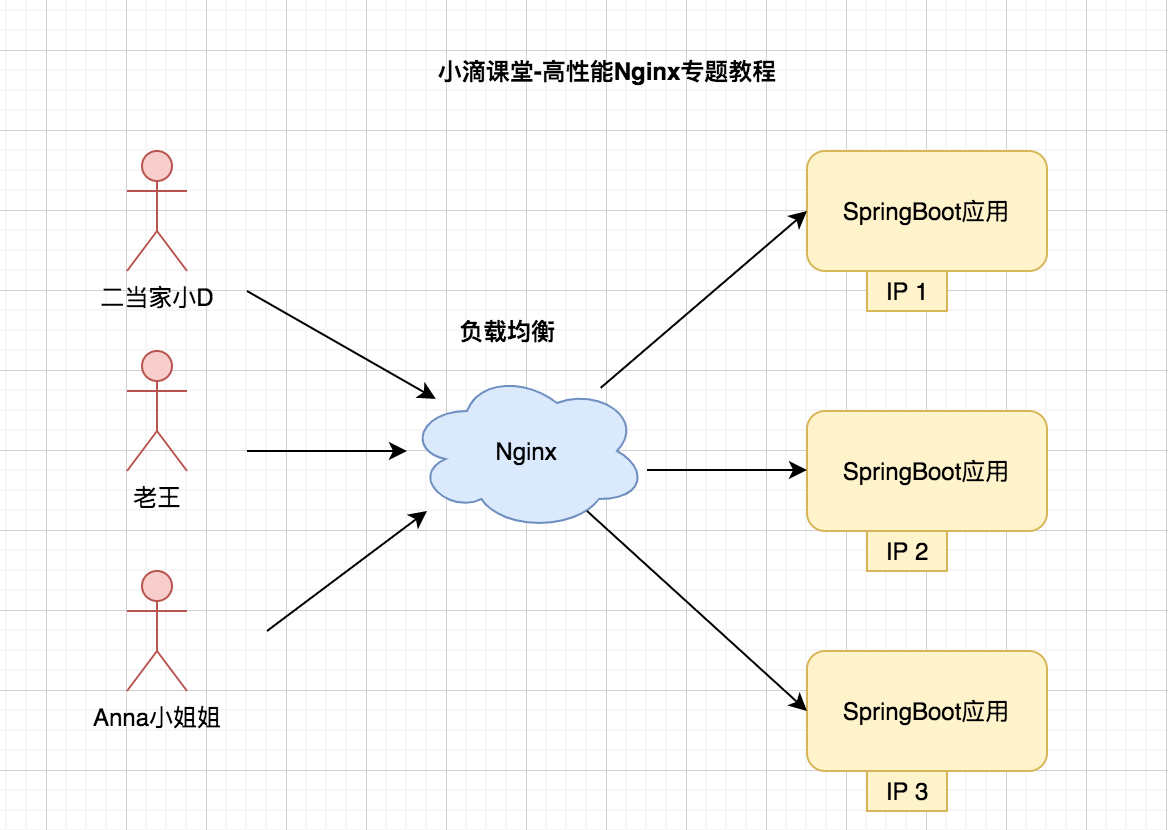
原理是将数据流量分摊到多个服务器执行，减轻每台服务器的压力，从而提高了数据的吞吐量

负载均衡的种类

通过硬件来进行解决，常见的硬件有NetScaler、F5、

Radware和Array等商用的负载均衡器，但比较昂贵的通过软件来进行解决，常见的软件有LVS、Nginx等,它们是基于Linux系统并且开源的负载均衡策略

目前性能和成本来看，Nginx是目前多数公司选择使用的



配置案例

upstream lbs {

server 192.168.0.106:8080;

server 192.168.0.106:8081;

}

location /api/ {

proxy\_pass http://lbs; proxy\_redirect default;

}

## 第4集 Nginx常见的负载均衡策略解析《上》

### 简介：Nginx常见的负载均衡策略解析

Nginx常见的负载均衡策略节点轮询（默认）

简介：每个请求按顺序分配到不同的后端服务器 场景：会造成可靠性低和负载分配不均衡，适合静态文件服务器

weight 权重配置

简介：weight和访问比率成正比，数字越大，分配得到的流量越高

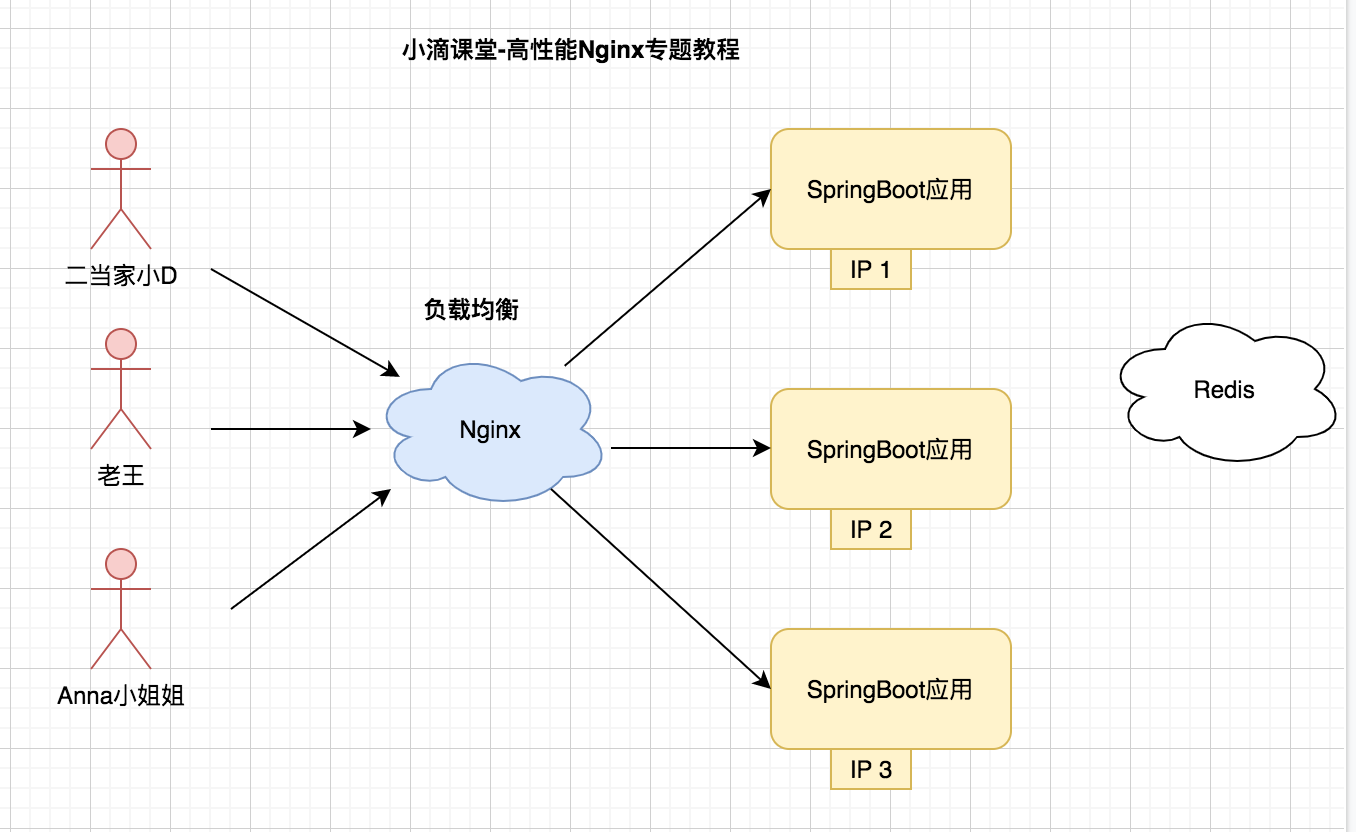
场景：服务器性能差异大的情况使用

upstream lbs {

server 192.168.159.133:8080 weight=5;

server 192.168.159.133:8081 weight=10;

}



## 第5集 Nginx常见的负载均衡策略解析《下》

### 简介：Nginx常见的负载均衡策略解析《下》

Nginx常见的负载均衡策略ip\_hash（固定分发）

简介：根据请求按访问ip的hash结果分配，这样每个用户就可以固定访问一个后端服务器

场景：服务器业务分区、业务缓存、Session需要单

点的情况

upstream lbs { ip\_hash;

server 192.168.159.133:8080;

server 192.168.159.133:8081;

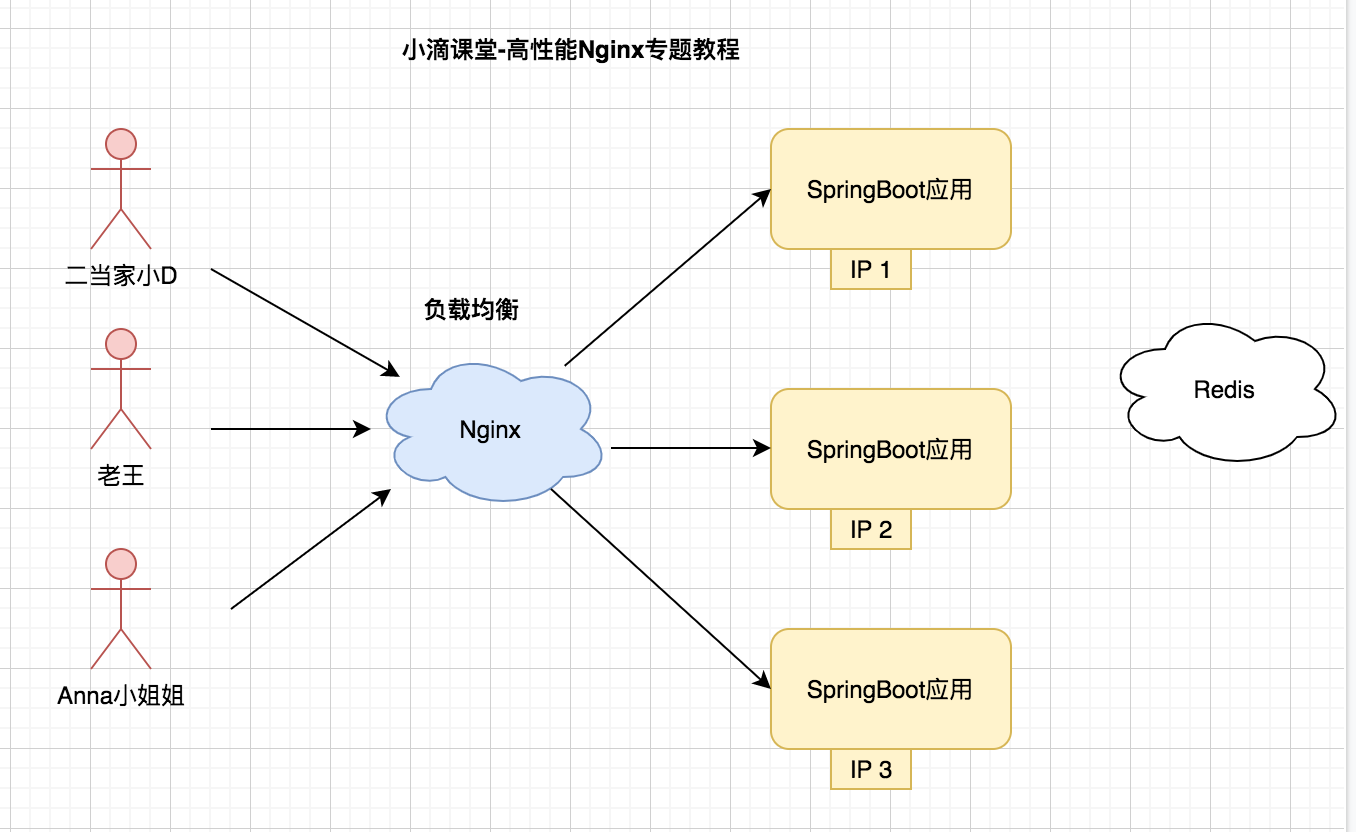
}

upstream还可以为每个节点设置状态值

down 表示当前的server暂时不参与负载

server 192.168.159.133:8080 down;

backup 其它所有的非backup机器down的时候，会请求

backup机器，这台机器压力会最轻，配置也会相对低server 192.168.159.133:8080 backup;

## 第6集 高级篇幅-Nginx后端节点可用性探测和配置实操

### 简介：Nginx探测后端节点可用性和配置实操

如果某个应用挂了，请求不应该继续分发过去

max\_fails 允许请求失败的次数，默认为1.当超过最大次数时就不会请求

fail\_timeout : max\_fails次失败后，暂停的时间，默认：fail\_timeout为10s

参数解释

max\_fails=N 设定Nginx与后端节点通信的尝试失败

的次数。

在fail\_timeout参数定义的时间内，如果失败的次数达到此值，Nginx就这个节点不可用。

在下一个fail\_timeout时间段到来前，服务器不会再被尝试。

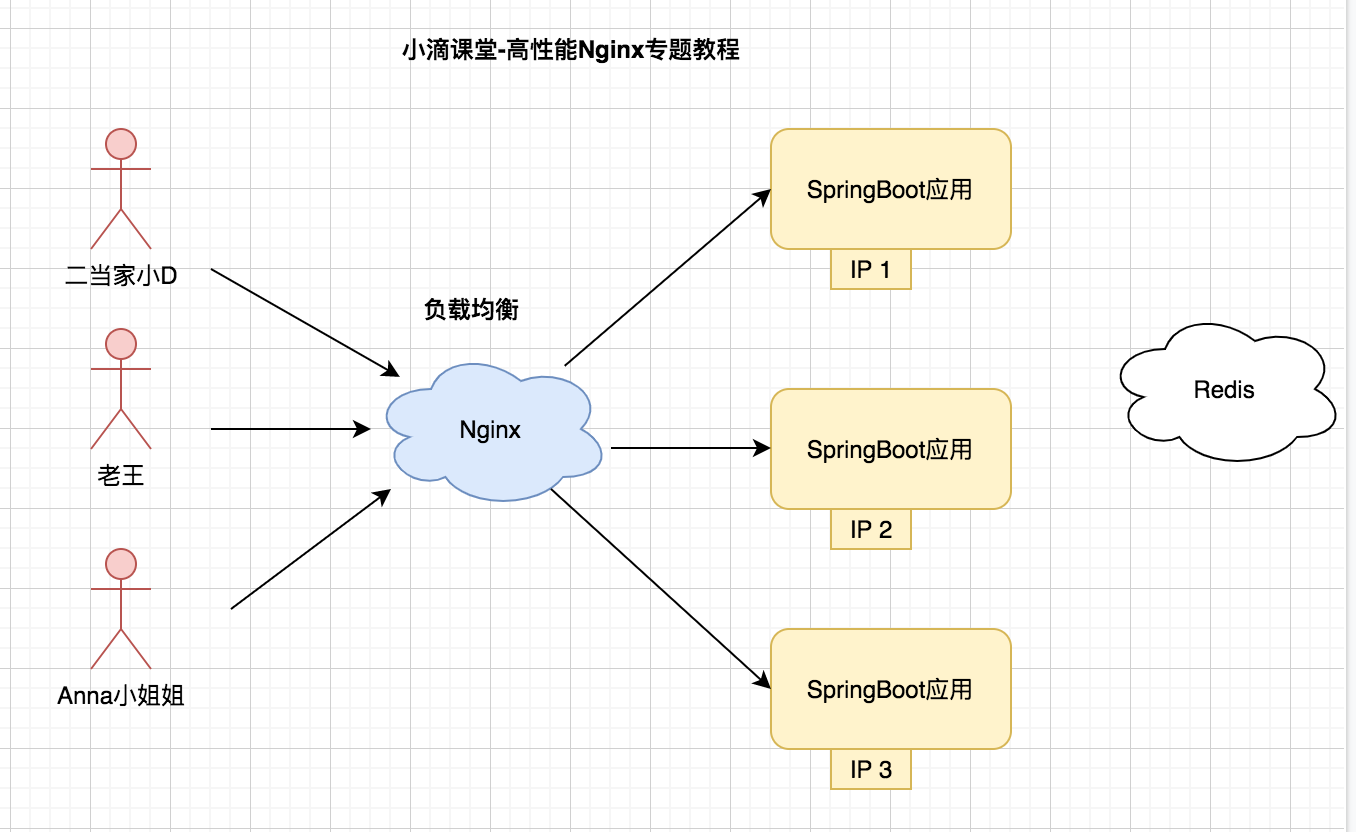
失败的尝试次数默认是1，如果设为0就会停止统计

尝试次数，认为服务器是一直可用的。具体什么是nginx认为的失败呢

可以通过指令proxy\_next\_upstream来配置什么是 失败的尝试。

注意默认配置时，http\_404状态不被认为是失败的

尝试。



配置实操

upstream lbs {

server 192.168.0.106:8080

max\_fails=2 fail\_timeout=60s ;

server 192.168.0.106:8081

max\_fails=2 fail\_timeout=60s;

}

location /api/ {

proxy\_pass http://lbs; proxy\_next\_upstream error timeout

http\_500 http\_503 http\_404;

}

案例实操

暂停一个后节点，然后访问接口大于10次，Nginx会把这个节点剔除

重启这个节点，在fail\_timeout周期里面不会再获取流

量

 **愿景："让编程不再难学，让技术与生活更加有趣" 更多架构课程请访问 xdclass.net**

# 第八章 案例实战-玩转Nginx经典应用

## 第1集 Nginx-全局异常兜底数据返回

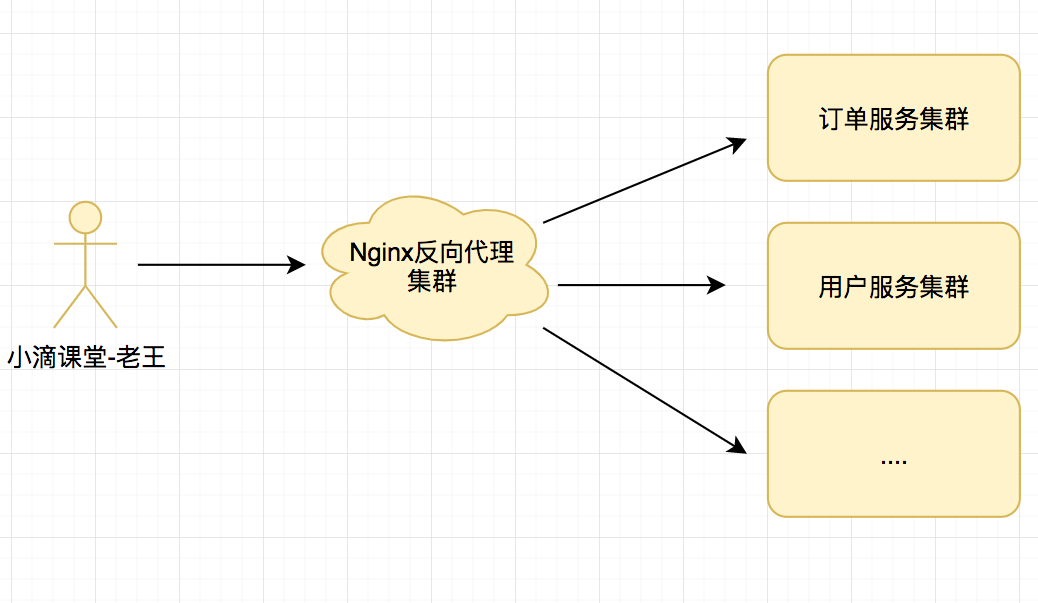
### 简介：Nginx自定义全局异常json数据

任何接口都是可能出错，4xx、5xx等

如果业务没有做好统一的错误管理，直接暴露给用户，无疑是看不懂

所以假如后端某个业务出错，nginx层也需要进行转换

让前端知道Http响应是200，其实是将错误的状态码定向至200，返回了全局兜底数据



location / {

proxy\_pass http://lbs; proxy\_redirect default;

# 存 放 用 户 的 真 实 ip proxy\_set\_header Host $host; proxy\_set\_header X-Real-IP

$remote\_addr;

proxy\_set\_header X-Forwarded-For

$proxy\_add\_x\_forwarded\_for;

proxy\_next\_upstream error timeout http\_503 non\_idempotent;

#开启错误拦截配置,一定要开启proxy\_intercept\_errors on;

}

# 不加 =200，则返回的就是原先的http错误码；配上后如果出现500等错误都返回给用户200状态，并跳转

至/default\_api

error\_page 404 500 502 503 504 =200

/default\_api;

location = /default\_api { default\_type application/json;

return 200 '{"code":"-1","msg":"invoke fail, not found "}';

}

## 第2集 网络安全-Nginx封禁恶意IP

### 简介：Nginx封禁恶意IP

网络攻击时有发生，

TCP洪水攻击、注入攻击、DOS等比较难防的有DDOS等

数据安全，防止对手爬虫恶意爬取，封禁IP 一般就是封禁ip

linux server的层面封IP：iptables

nginx的层面封IP ，方式多种 (但 req还是会打进来， 让nginx 返回 403, 占用资源)

Nginx作为网关，可以有效的封禁ip

单独网站屏蔽IP的方法，把include 应的在server{}语句块,虚拟主机 所有网站屏蔽IP的方法，把include

{}语句块。

xxx;

放到网址对

xxx;

放到http

nginx配置如下：

http{

# ....

include blacklist.conf;

}

location / {

proxy\_pass http://lbs;

proxy\_redirect default;

}

#blacklist.conf目录下文件内容deny 192.168.159.2;

deny 192.168.159.32;

./nginx -s reload #重新加载配置，不中断服务

拓展-自动化封禁思路

编写shell脚本

AWK统计access.log，记录每秒访问超过60次的ip，然后配合nginx或者iptables进行封禁

crontab定时跑脚本

## 第3集 跨域案例-Nginx如何配置浏览器跨域

### 简介：Nginx配置解决浏览器跨域

跨域：浏览器同源策略 1995年，同源政策由 Netscape 公司引入浏览器。目前，所有浏览器都实行这个政策。 最初，它的含义是指，A网页设置的 Cookie，B网页不能打开，除非这两个网页"同源"。所谓"同源"指的是"三个相同"

协议相同域名相同

端口相同

http https

[www.xdclass.net](http://www.xdclass.net/)

80

81

一句话：浏览器从一个域名的网页去请求另一个域名的资源

时，域名、端口、协议任一不同，都是跨域

浏览器控制台跨域提示：

No 'Access-Control-Allow-Origin' header is present on the requested resource. Origin 'null' is therefore not allowed access.

解决方法

JSONP

Http响应头配置允许跨域nginx层配置

程序代码中处理通过拦截器配置

Nginx开启跨域配置location下配置

location / {

add\_header 'Access-Control-Allow- Origin' $http\_origin;

add\_header 'Access-Control-Allow- Credentials' 'true';

add\_header 'Access-Control-Allow- Headers' 'DNT,web-token,app- token,Authorization,Accept,Origin,Keep- Alive,User-Agent,X-Mx-ReqToken,X-Data- Type,X-Auth-Token,X-Requested-With,If- Modified-Since,Cache-Control,Content- Type,Range';

add\_header Access-Control-Allow- Methods 'GET,POST,OPTIONS';

#如果预检请求则返回成功,不需要转发到后端 if ($request\_method = 'OPTIONS') {

add\_header 'Access-Control-Max-Age' 1728000;

add\_header 'Content-Type' 'text/plain; charset=utf-8';

add\_header 'Content-Length' 0;

return 200;

}

}

## 第4集 路径匹配-Nginx的locatioin规则应用

### 简介：讲解nginx的location规则

正则

^ 以什么开始

$ 以什么结束

^/api/user$

location 路径匹配

### 语法 location [ = | ~ | ~\* | ^~ ] uri { }

location = /uri

= 表示精准匹配，只要完全匹配上才能生效

location /uri

不带任何修饰符，表示前缀匹配

location ^~ /uri/

匹配任何已 /uri/ 开头的任何查询并且停止搜索

location /

通用匹配，任何未匹配到其他location的请求都会匹配到

正则匹配

区分大小写匹配（~） 不区分大小写匹配（~\*）

优先级(不要写复杂，容易出问题和遗忘)

精准匹配 > 字符串匹配(若有多个匹配项匹配成功，那么选择匹配长的并记录) > 正则匹配

案例

server {

server\_name xdclass.net;

location ~^/api/pub$ {

...

}

}

^/api/pub$这个正则表达式表示字符串必须以/开始，以b

$ 结 束 ， 中 间 必 须 是 /api/pub <http://xdclass.net/api/v1>匹配（完全匹配） <http://xdclass.net/API/PUB>不匹配，大小写敏感<http://xdclass.net/api/pub?key1=value1>匹配<http://xdclass.net/api/pub/>不匹配

<http://xdclass.net/api/public>不匹配，不能匹配正则表达式

测试

location = /img/test.png {

return 1;

}

location /img/test.png {

return 2;

}

location ^~/img/ {

return 3;

}

location = / { return 4;

}

location / {

return 5;

}

## 第5集 地址重定向-Nginx的rewrite规则应用

### 简介：讲解nginx的rewrite规则和应用

重写-重定向

rewrite 地址重定向，实现URL重定向的重要指令，他根据

regex(正则表达式)来匹配内容跳转到

语法 rewrite regex replacement[flag]

rewrite ^/(.\*) https://xdclass.net/$1 permanent

# 这是一个正则表达式，匹配完整的域名和后面的路径地址

# replacement部分是https://xdclass.net/$1，$1

是取自regex部分()里的内容

常用正则表达式：

|  |  |
| --- | --- |
| **字符** | **描述** |
| ^ | 匹配输入字符串的起始位置 |
| $ | 匹配输入字符串的结束位置 |
| \* | 匹配前面的字符零次或者多次 |
| + | 匹配前面字符串一次或者多次 |
| ? | 匹配前面字符串的零次或者一次 |
| . | 匹配除“\n”之外的所有单个字符 |
| (pattern) | 匹配括号内的pattern |

rewrite 最后一项flag参数

|  |  |
| --- | --- |
| **标记符号** | **说明** |
| last | 本条规则匹配完成后继续向下匹配新的  location URI规则 |
| break | 本条规则匹配完成后终止，不在匹配任何规则 |
| redirect | 返回302临时重定向 |
| permanent | 返回301永久重定向 |

应用场景

非法访问跳转，防盗链网站更换新域名

http跳转https

不同地址访问同一个虚拟主机的资源

## 第6集 实时通信-Nginx配置Websocket反向代理

### 简介：讲解nginx 配置websocket反向代理

配置

server {

listen 80; server\_name xdclass.net; location / {

proxy\_pass http://lbs;

proxy\_read\_timeout 300s; //websocket空闲保持时长

proxy\_set\_header Host $host; proxy\_set\_header X-Real-IP $remote\_addr; proxy\_set\_header X-Forwarded-For

$proxy\_add\_x\_forwarded\_for; proxy\_http\_version 1.1;

proxy\_set\_header Upgrade $http\_upgrade; proxy\_set\_header Connection

$connection\_upgrade;

}

}

核心是下面的配置 其他和普通反向代理没区别, 表示请求服务器升级协议为WebSocket

proxy\_set\_header Upgrade $http\_upgrade; proxy\_set\_header Connection

$connection\_upgrade;

服务器处理完请求后，响应如下报文# 状态码为101

HTTP/1.1 101 Switching Protocols Upgrade: websocket

Connection: upgrade

 **愿景："让编程不再难学，让技术与生活更加有趣" 更多架构课程请访问 xdclass.net**

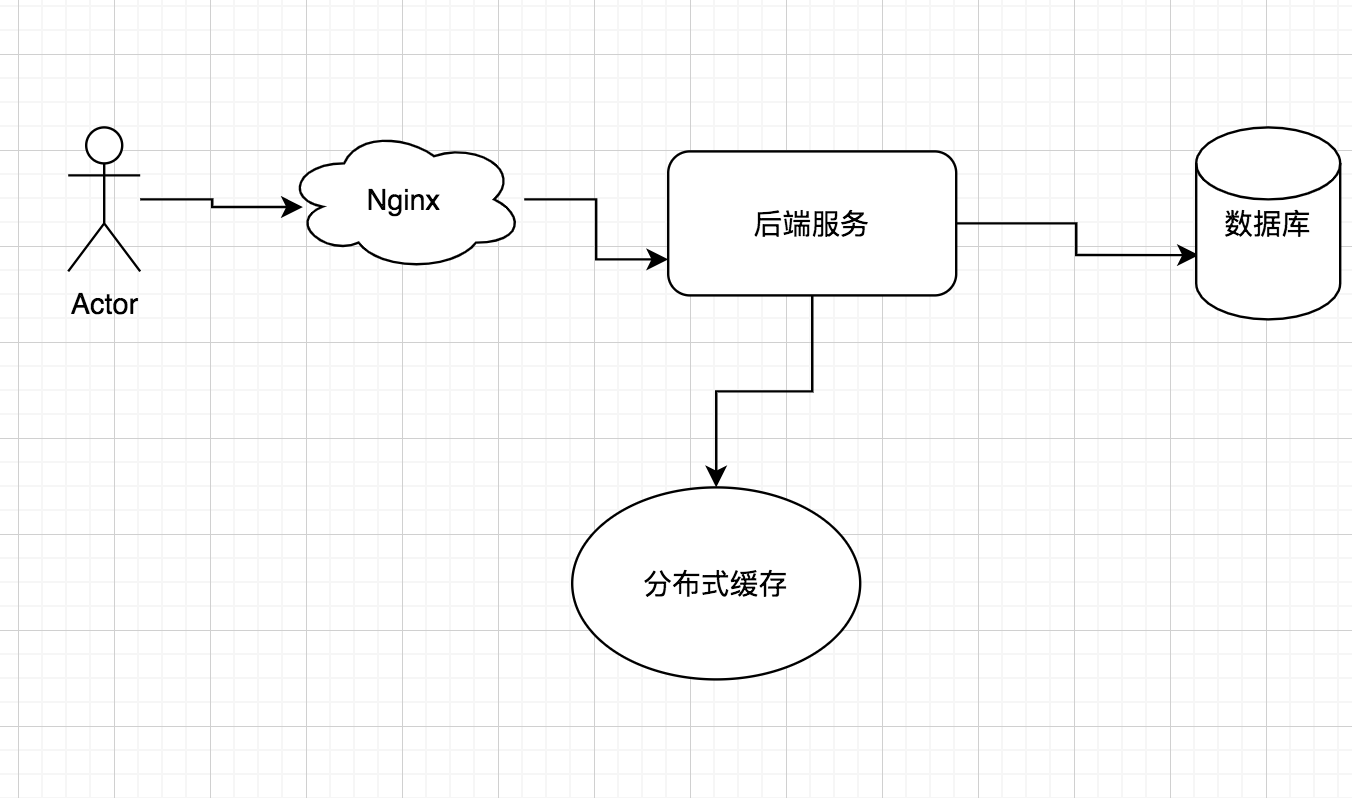
# 第九章 Nginx业务接口性能优化

## 第1集 高并发必备利器之服务端缓存前置《上》

### 简介：讲解Nginx的配置服务端缓存核心配置

常见的开发人员控制的缓存分类数据库缓存

应用程序缓存**Nginx网关缓存**前端缓存



让后端结果缓存离用户更进一步

### /root/cache

本地路径，用来设置Nginx缓存资源的存放地址

### levels=1:2

默认所有缓存文件都放在上面指定的根路径中，可能影响缓存的性能，推荐指定为 2 级目录来存储缓存文件；1和2表示用1位和2位16进制来命名目录名称。第一级目录用1位16进制命名，如a；第二级目 录用2位16进制命名，如3a。所以此例中一级目录有16个，二级目录有16\*16=256个,总目录数为16 \* 256=4096个。

当levels=1:1:1时，表示是三级目录，且每级目录数均为16个

### key\_zone

在共享内存中定义一块存储区域来存放缓存的 key

和 metadata

### max\_size

最大 缓存空间, 如果不指定会使用掉所有磁盘空间。当达到 disk 上限后，会删除最少使用的 cache

### inactive

某个缓存在inactive指定的时间内如果不访问，将会从缓存中删除

### proxy\_cache\_valid

配置nginx cache中的缓存文件的缓存时

间,proxy\_cache\_valid 200 304 2m 对于状态为200 和304的缓存文件的缓存时间是2分钟

### use\_temp\_path

建议为 off，则 nginx 会将缓存文件直接写入指定的

cache 文件中

### proxy\_cache

启用proxy cache，并指定key\_zone，如果proxy\_cache off表示关闭掉缓存

### add\_header Nging-Cache "$upstream\_cache\_status"

用于前端判断是否是缓存，miss、hit、expired(缓存过期)、updating(更新，使用旧的应答)

proxy\_cache\_path /root/cache levels=1:2 keys\_zone=xd\_cache:10m max\_size=1g inactive=60m use\_temp\_path=off;

server {

location /{

...

proxy\_cache xd\_cache; proxy\_cache\_valid 200 304 10m;

proxy\_cache\_valid 404 1m; proxy\_cache\_key

$host$uri$is\_args$args;

add\_header Nginx-Cache "$upstream\_cache\_status";

}

}

还原nginx配置，只保留upstream模块

## 第2集 高并发必备利器之服务端缓存前置《下》

### 简介：讲解Nginx的配置服务端缓存核心配置

配置实操

请求后端json接口，通过控制台日志判断是否有到后端 服务

注意：

nginx缓存过期影响的优先级进行排序为：inactvie >

源服务器端Expires/max-age > proxy\_cache\_valid

如果出现 Permission denied 修改nginx.conf，将第一行修改为 user root

默认情况下GET请求及HEAD请求会被缓存，而POST请求不会被缓存，并非全部都要缓存，可以过滤部分路径不用缓存

缓存清空

直接rm删除

ngx\_cache\_purge

缓存命中率统计

前端打点日志上报

nginx日志模板增加信息

$upstream\_cache\_status

## 第3集 Nginx性能优化之静态资源压缩

### 简介：讲解Nginx的压缩配置和实操

压缩配置

对文本、js和css文件等进行压缩，一般是压缩后的大小是原始大小的25%

#开启gzip,减少我们发送的数据量gzip on;

gzip\_min\_length 1k;

#4个单位为16k的内存作为压缩结果流缓存

gzip\_buffers 4 16k;

#gzip压缩比，可在1~9中设置，1压缩比最小，速度最快，

9压缩比最大，速度最慢，消耗CPU

gzip\_comp\_level 4;

#压缩的类型

gzip\_types application/javascript text/plain text/css application/json application/xml text/javascript;

#给代理服务器用的，有的浏览器支持压缩，有的不支持，所以避免浪费不支持的也压缩，所以根据客户端的HTTP头来判断，是否需要压缩

gzip\_vary on;

#禁用IE6以下的gzip压缩，IE某些版本对gzip的压缩支持很不好

gzip\_disable "MSIE [1-6].";

压缩前后区别（上传js文件进行验证）

location /static {

alias /usr/local/software/static;

}

面试题：压缩是时间换空间，还是空间换时间？

web层主要涉及浏览器和服务器的网络交互，而网络交 互显然是耗费时间的

要尽量减少交互次数

降低每次请求或响应数据量。开启压缩

在服务端是时间换空间的策略，服务端需要牺牲时间进行压缩以减小响应数据大小

压缩后的内容可以获得更快的网络传输速度，时间是得到了优化

所以是双向的

 **愿景："让编程不再难学，让技术与生活更加有趣" 更多架构课程请访问 xdclass.net**

# 第十章 【高级篇幅】 Nginx和Https实战的那些事

## 第1集 大话新一代传输协议Https

### 简介: 介绍什么是https，使用的好处

什么是Https，和http的区别

HTTPS (Secure Hypertext Transfer Protocol)安全超文本传输协议，是身披SSL外壳的HTTP

HTTPS是一种通过计算机网络进行安全通信的传输协议，经由HTTP进行通信，利用SSL/TLS建立全信道，加密数据包。

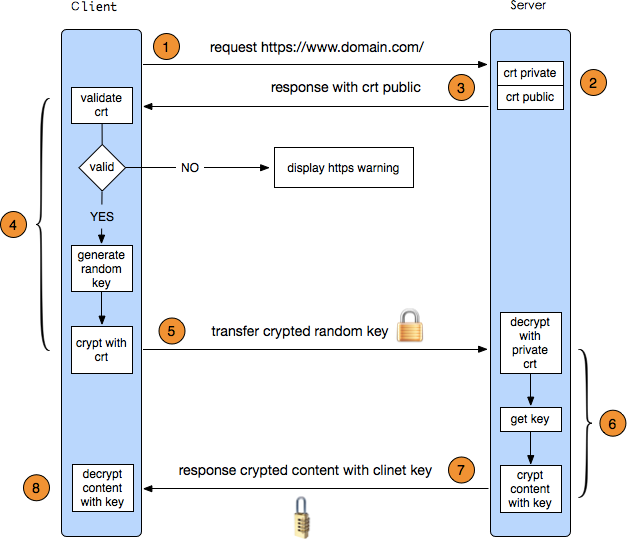
为什么要用呢

HTTPS 协议是由 SSL+HTTP 协议构建的可进行加密传输、身份认证的网络协议，要比 HTTP 协议安全，可防止数据在传输过程中被窃取、改变，确保数据的完整性

流程

秘钥交换使用非对称加密，内容传输使用对称加密的方

式



## 第2集 阿里云Https证书申请和准备

### 简介: 阿里云免费https证书申请和准备

证书申请->审核等待

[https://common-buy.aliyun.com/?commodityCode= cas](https://common-buy.aliyun.com/?commodityCode=cas)



证书上传

## 第3集 阿里云Nginx配置Https证书案例实操

### 简介: 阿里云 Nginx配置https证书配置实操

删除原先的nginx，新增ssl模块

./configure --prefix=/usr/local/nginx --

with-http\_stub\_status\_module --with- http\_ssl\_module

make

make install

#查看是否成功

/usr/local/nginx/sbin/nginx -V

Nginx配置https证书

server {

listen 443 ssl; server\_name 16web.net;

ssl\_certificate

/usr/local/software/biz/key/4383407\_16web.ne t.pem;

ssl\_certificate\_key

/usr/local/software/biz/key/4383407\_16web.ne t.key;

ssl\_session\_cache shared:SSL:1m; ssl\_session\_timeout 5m;

ssl\_ciphers HIGH:!aNULL:!MD5; ssl\_prefer\_server\_ciphers on;

location / {

root html;

index index.html index.htm;

}

}

https访问实操杀掉原先进程

防火墙关闭或者开放443端口

service firewalld stop

网络安全组开放端口

 **愿景："让编程不再难学，让技术与生活更加有趣" 更多架构课程请访问 xdclass.net**

# 第十一章 【高级篇】玩转Nginx整合

**OpenResty**

## 第1集 Nginx的第三方利器OpenResty+Lua介绍

### 简介: 介绍Nginx的扩展 整合框架OpenResty

什么是OpenResty, 为什么要用OpenResty？

由章亦春发起，是基于Ngnix和Lua的高性能web平台，内部集成精良的LUa库、第三方模块、依赖, 开发者可以方便搭建能够处理高并发、扩展性极高的动态web应用、web服务、动态网关。

OpenResty将Nginx核心、LuaJIT、许多有用的Lua库和Nginx第三方模块打包在一起

Nginx是C语言开发，如果要二次扩展是很麻烦的，而基于OpenResty，开发人员可以使用 Lua 编程语言对 Nginx 核心模块进行二次开发拓展

性能强大，OpenResty可以快速构造出1万以上并发连接响应的超高性能Web应用系统

官网：[http://openresty.org](http://openresty.org/)

阿里、腾讯、新浪、酷狗音乐等都是 OpenResty 的深度用户

拓展

让Web 服务直接跑在 Nginx 服务内部,充分利用 Nginx 的非阻塞 I/O 模型,不仅仅对 HTTP 客户端请求,甚至于对远程后端诸如 MySQL, Memcaches 以及 Redis 等都进行一致的高性能响应。所以对于一些高性能的服务来说，可以直接使用 OpenResty 访问 Mysql或Redis等，而不需要通过第三方语言（PHP、Python、Ruby）等来访问数据库再返回，这大大提高了应用的性能

Lua脚本介绍

官网：<http://www.lua.org/start.html>

Lua 由标准 C 编写而成,没有提供强大的库,但可以很容易的被 C/C++ 代码调用，也可以反过来调用 C/C++ 的函数。

在应用程序中可以被广泛应用，不过Lua是一种脚本/动态语言，不适合业务逻辑比较重的场景，适合小巧的应用场景，

代码行数保持在几十行到几千行。

LuaJIT 是采用 C 和汇编语言编写的 Lua 解释器与即时

编译器

什么是ngx\_lua

ngx\_lua是Nginx的一个模块，将Lua嵌入到Nginx中，从

而可以使用Lua来编写脚本，部署到Nginx中运行，

即Nginx变成了一个Web容器；开发人员就可以使用Lua语言

开发高性能Web应用了。

OpenResty提供了常用的ngx\_lua开发模块

lua-resty-memcached lua-resty-mysql

lua-resty-redis lua-resty-dns

lua-resty-limit-traffic

通过上述的模块，可以用来操作 mysql数据库、redis、memcached等，也可以自定义模块满足其他业务需求，

很多经典的应用，比如开发缓存前置、数据过滤、API请求聚合、AB测试、灰度发布、降级、监控、限流、防火墙、黑白名单等

## 第2集 OpenResty + Lua相关环境准备

### 简介：OpenResty和Lua相关环境准备

OpenResty安装

[下载：http://openresty.org/en/linux-packages.html #centos](http://openresty.org/en/linux-packages.html#centos)

# add the yum repo:

wget https://openresty.org/package/centos/openresty. repo

sudo mv openresty.repo /etc/yum.repos.d/

# update the yum index:

sudo yum check-update

sudo yum install openresty

#安装命令行工具

sudo yum install openresty-resty

# 列出所有 openresty 仓库里的软件包sudo yum --disablerepo="\*" -- enablerepo="openresty" list available

#查看版本resty -V

## 第3集 高并发利器-Nginx+OpenResty第一个例子

### 简介: 讲解Nginx+OpenRestry开发第一个例子

Nginx+OpenRestry开发

编辑：/usr/local/openresty/nginx/conf/nginx.conf

http{

# 虚拟机主机块

server{

# 监听端口

listen 80;

# 配置请求的路由

location /{

default\_type text/html; content\_by\_lua\_block{

ngx.say("hello world; xdclass.net 小滴课堂");

}

}

}

}

#使用其他方式http{

# 虚拟机主机块,还需要配置lua文件扫描路径

lua\_package\_path "$prefix/lualib/?.lua;;"; lua\_package\_cpath "$prefix/lualib/?.so;;"; server{

# 监听端口

listen 80;

# 配置请求的路由

location /{

default\_type text/html; content\_by\_lua\_file

lua/xdclass.lua;

}

}

}

启动Nginx（直接用openresty里面的nginx即可，默认安 装了多个模块）

访 问 curl 127.0.0.1

如果浏览器访问会出现文件下载，因为没有Html头信息

注意：如果需要指定配置文件 nginx -c 配置文件路径比如 ./nginx -c

/usr/local/nginx/conf/nginx.conf

## 第4集 Nginx内置变量 和 OpenResty 请求阶段划分讲解

### 简介: 讲解Nginx+OpenRestry常见基础知识

nginx内置变量

提供丰富的内置变量, openresty里面使用参考下面的文档

[https://github.com/openresty/lua-nginx-module#n](https://github.com/openresty/lua-nginx-module#ngxvarvariable)

[gxvarvariable](https://github.com/openresty/lua-nginx-module#ngxvarvariable)

部分变量是可以被修改的，部分是不给修改

|  |  |
| --- | --- |
| **名称** | **说明** |
| $arg\_name | 请求中的name参数 |
| $args | 请求中的参数 |
| $content\_length | HTTP请求信息里的"Content-Length" |
| $content\_type | 请求信息里的"Content-Type" |
| $host | 请求信息中的"Host"，如果请求中没有Host 行，则等于设置的服务器名 |
| $hostname | 机器名使用 gethostname系统调用的值 |
| $http\_cookie | cookie 信息 |
| $http\_referer | 引用地址 |
| $http\_user\_agent | 客户端代理信息 |
| $http\_via | 最后一个访问服务器的Ip地址。 |
| $http\_x\_forwarded\_for | 相当于网络访问路径 |
| $is\_args | 如果请求行带有参数，返回“?”，否则返回空 |

|  |  |
| --- | --- |
|  | 字符串 |
| $limit\_rate | 对连接速率的限制 |
| $nginx\_version | 当前运行的nginx版本号 |
| $pid | worker进程的PID |
| $query\_string | 与$args相同 |
| $remote\_addr | 客户端IP地址 |
| $remote\_port | 客户端端口号 |
| $request | 用户请求 |
| $request\_method | 请求的方法，比如"GET"、"POST"等 |
| $request\_uri | 请求的URI，带参数 |
| $scheme | 所用的协议，比如http或者是https |
| $server\_name | 请求到达的服务器名 |
| $server\_port | 请求到达的服务器端口号 |
| $server\_protocol | 请求的协议版本，"HTTP/1.0"或"HTTP/1.1" |
| $uri | 请求的URI，可能和最初的值有不同，比如经过重定向之类的 |

nginx对于请求的处理分多个阶段,Nginx , 从而让第三方模块通过挂载行为在不同的阶段来控制, 大致如下

初始化阶段（Initialization Phase）

init\_by\_lua\_file init\_worker\_by\_lua\_file

重写与访问阶段（Rewrite / Access Phase） rewrite\_by\_lua\_file

access\_by\_lua\_file

内容生成阶段（Content Phase）

content\_by\_lua\_file

日志记录阶段（Log Phase）

## 第5集 Nginx+OpenResty +Lua开发内网访问限制

### 简介: 讲解Nginx+OpenRestry开发网络访问限制

生产环境-管理后台一般需要指定的网络才可以访问，网段/ip等

Nginx+OpenRestry+Lua开发

http{

# 这里设置为 off，是为了避免每次修改之后都要重新reload 的麻烦。

# 在生产环境上需要 lua\_code\_cache 设置成 on。lua\_code\_cache off;

# lua\_package\_path可以配置openresty的文件寻址路径，

$PREFIX 为openresty安装路径

# 文件名使用“?”作为通配符，多个路径使用“;”分隔，默认的查找路径用“;;”

# 设置纯 Lua 扩展库的搜寻路径

lua\_package\_path "$prefix/lualib/?.lua;;";

# 设置 C 编写的 Lua 扩展模块的搜寻路径(也可以用 ';;') lua\_package\_cpath "$prefix/lualib/?.so;;";

server {

location / {

access\_by\_lua\_file lua/white\_ip\_list.lua; proxy\_pass http://lbs;

}

}

lua/white\_ip\_list.lua

local black\_ips = {["127.0.0.1"]=true}

local ip = ngx.var.remote\_addr

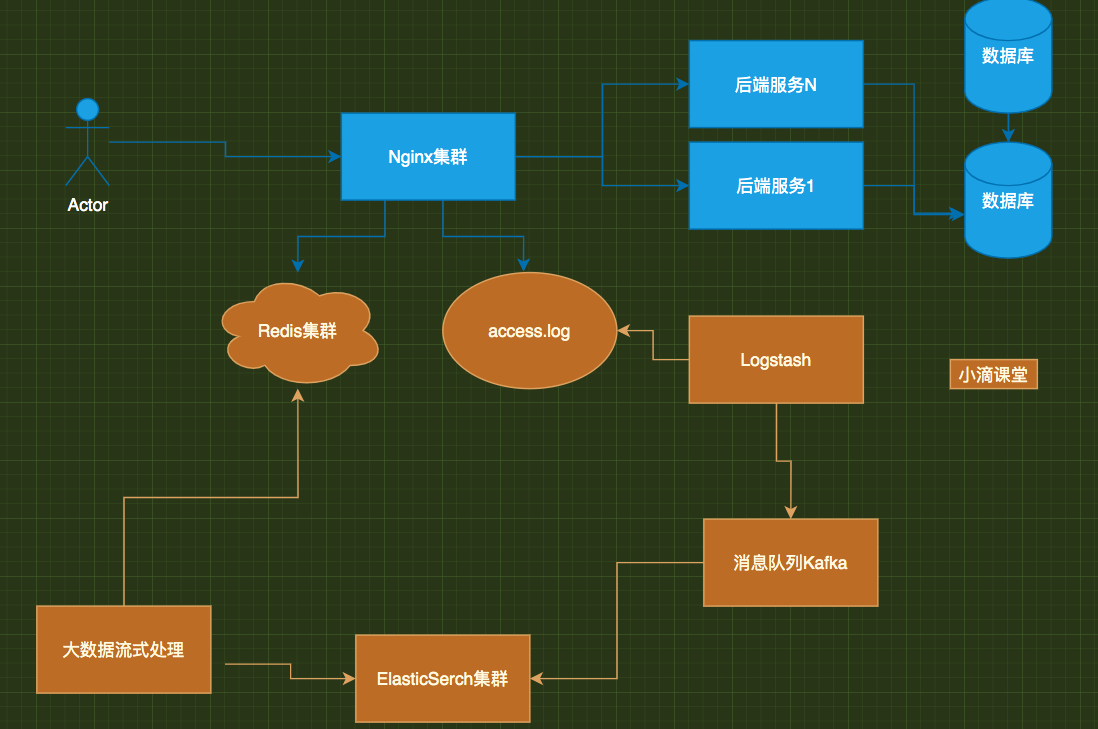
if true == black\_ips[ip] then ngx.exit(ngx.HTTP\_FORBIDDEN) return;

end

拓展

如何做一个动态黑名单控制?

里面 /usr/local/openresty/lualib/resty 很多第三方模块



## 第6集 案例实战-Nginx+OpenResty实现资源下载限速

### 简介: 讲解Nginx+OpenRestry开发下载限速

限速限流应用场景

下载限速：保护带宽及服务器的IO资源

请求限流：防止恶意攻击，保护服务器及资源安全

限制某个用户在一个给定时间段内能够产生的HTTP 请求数

限流用在保护上游应用服务器不被在同一时刻的大量用户访问

openResty下载限速案例实操

Nginx 有一个 $limit\_rate ，这个反映的是当前请求每秒能响应的字节数, 该字节数默认为配置文件中

指令的设值

limit\_rate

#当前请求的响应上限是 每秒 300K 字节

location /download { access\_by\_lua\_block {

ngx.var.limit\_rate = "300K"

}

alias /usr/local/software/app;

}

## 第7集 【面试题】网盘静态资源下载限速实现原理

### 简介: 讲解常见的资源下载限速实现原理

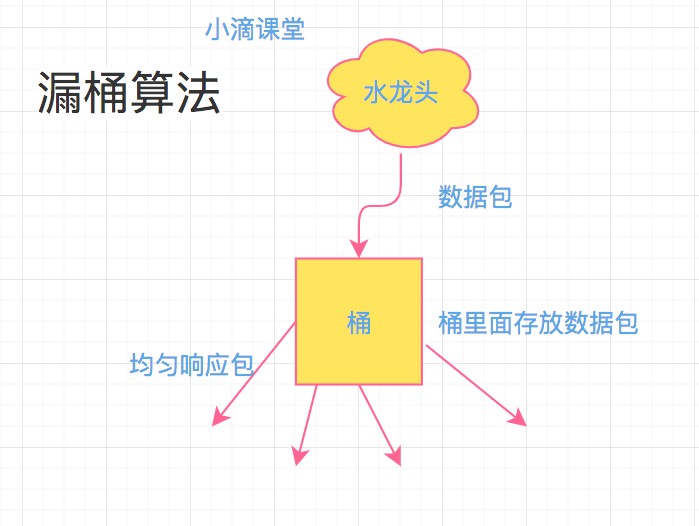
下载限速实现原理

目的：限制下载速度

常用的是漏桶原理和令牌桶原理

什么是漏桶算法

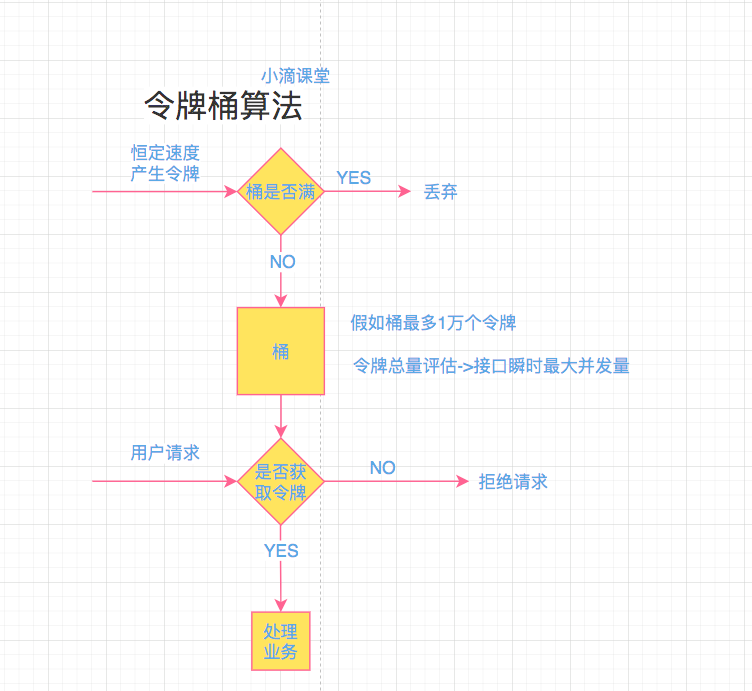
备注：如果是请求限流，请求先进入到漏桶里，漏桶以固定的速度出水，也就是处理请求，当水加的过快也就是请求过多，桶就会直接溢出，也就是请求被丢弃拒绝了，所以漏桶算法能强行限制数据的传输速率或请求数



什么是令牌桶算法

备注：只要突发并发量不高于桶里面存储的令牌数据， 就可以充分利用好机器网络资源。

如果桶内令牌数量小于被消耗的量，则产生的令牌的速度就是均匀处理请求的速度



 **愿景："让编程不再难学，让技术与生活更加有趣" 更多架构课程请访问 xdclass.net**

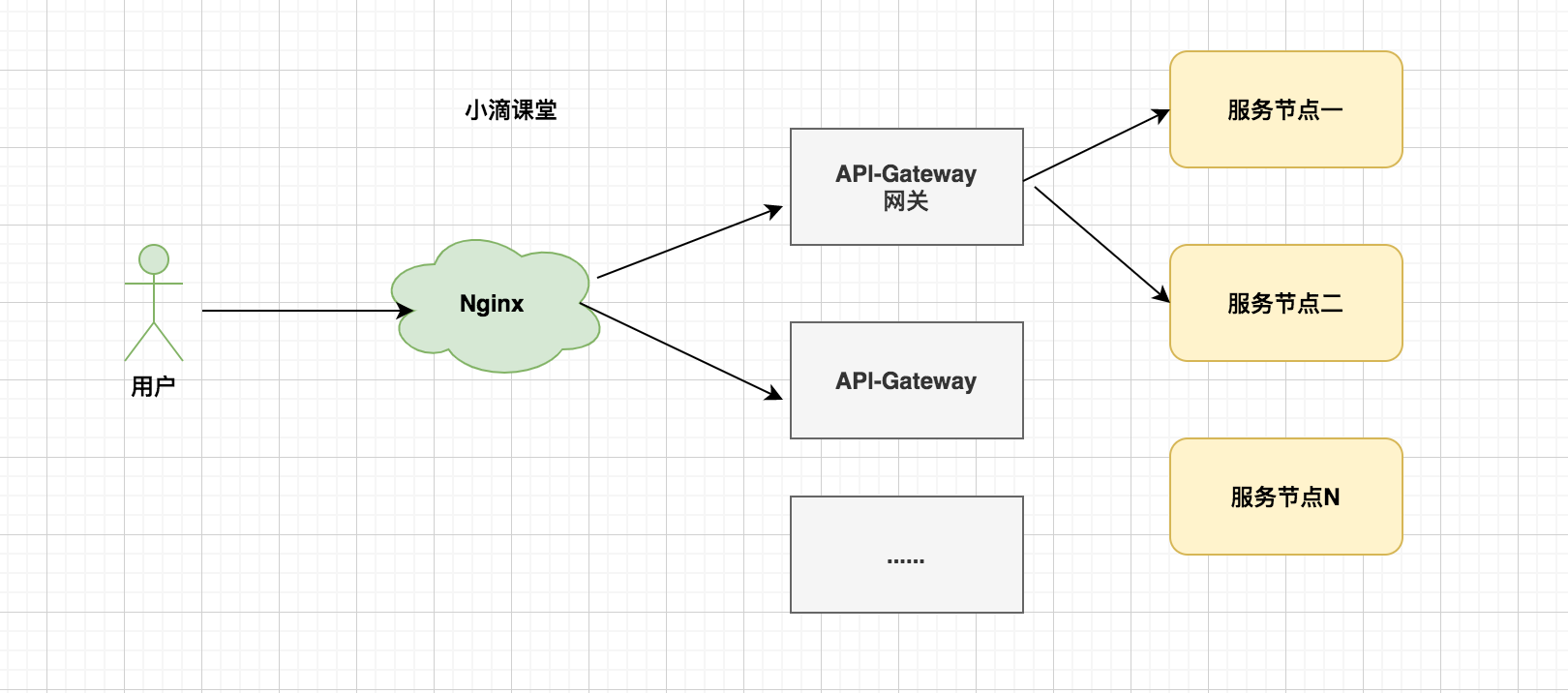
# 第十二章 【高级篇幅】 Ngnix高可用解决方案LVS+KeepAlived

## 第1集 全链路高可用之Nginx基础架构问题分析

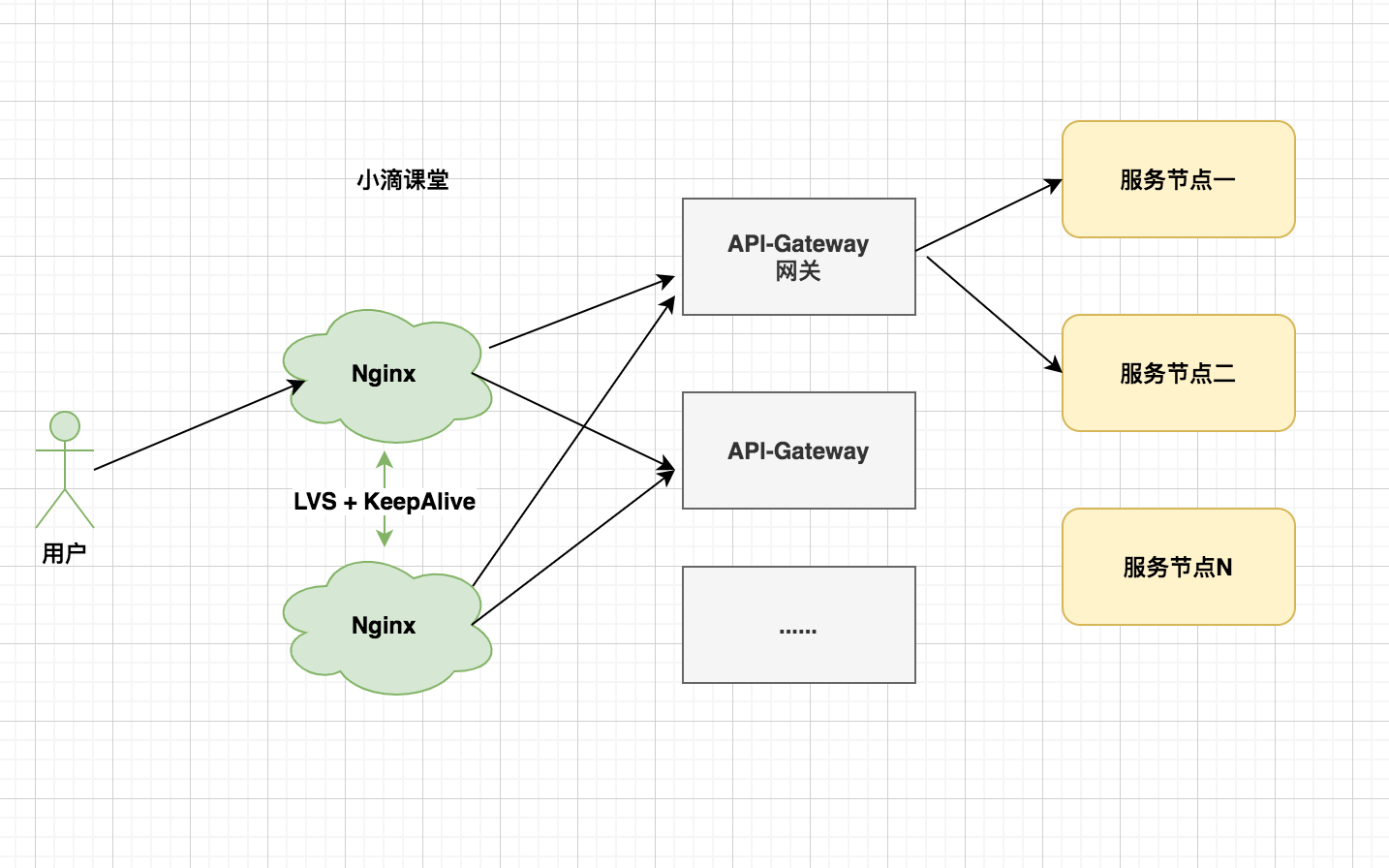
### 简介：讲解Nginx单点问题剖析

全链路高可用之Nginx反向代理单点故障分析

dns轮训多个ip，假如某个nginx挂了，怎么办



Nginx集群架构（vip ）



Nginx高可用解决方案-基础

国际标准化组织（ISO）制定的一个用于计算机或通信系统间

互联的标准体系。

从低到高分别是：

物理层、数据链路层、网络层、传输层、会话层、表示层和

应用层

四层工作在OSI第四层 也就是传输层

七层工作在最高层，也就是应用层

F5、LVS（四层负载 **tcp**）

用虚拟ip+port接收请求,再转发到对应的真实机器

HAproxy、Nginx(七层负载)

用虚拟的url或主机名接收请求,再转向相应的处理服 务器

## 第2集 业界主流的高可用方案 Linux虚拟服务器LVS 讲解

### 简介：讲解业界主流高可用方案Linux虚拟服务器 LVS讲解

什么是LVS

官网 [www.linuxvirtualserver.org](http://www.linuxvirtualserver.org/)

LVS是Linux Virtual Server,Linux虚拟服务器，是一个

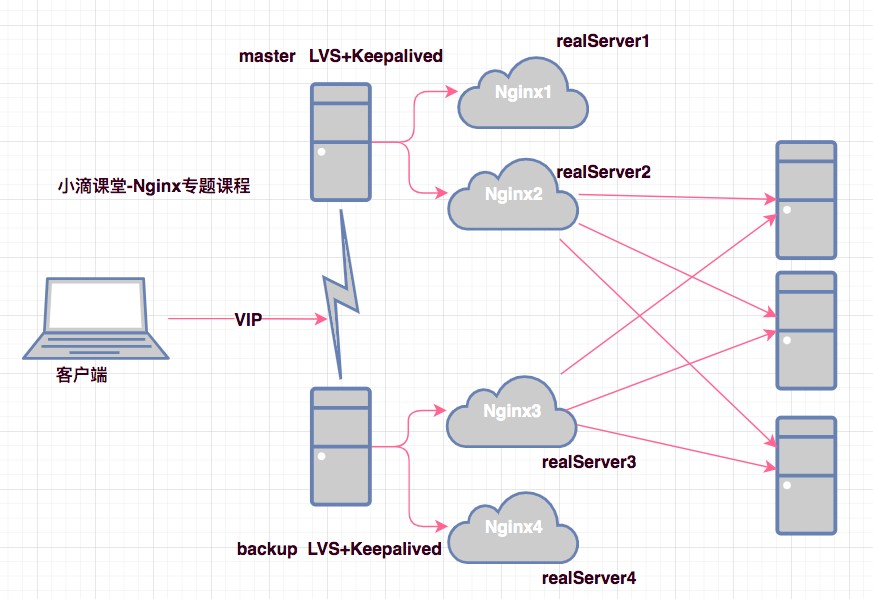
虚拟的服务器集群系统

项目是由章文嵩博士成立，是中国国内最早出现的自由软件项目

之一

Linux2.4 内核以后，LVS 已经是 Linux 标准内核的一部分

软件负载解决的两个核心问题是：选谁、转发



提供了10多种调度算法： 轮询、加权轮询、最小连接、目标地址散列、源地址散列等

三种负载均衡转发技术

NAT：数据进出都通过 LVS, 前端的Master既要处理客户端发起的请求，又要处理后台RealServer的响应信 息，将RealServer响应的信息再转发给客户端, 容易成为整个集群系统性能的瓶颈; (支持任意系统且可以实现端口映射)

DR: 移花接木,最高效的负载均衡规则,前端的Master只处理客户端的请求，将请求转发给RealServer，由后台的RealServer直接响应客户端，不再经过Master, 性能要优于LVS-NAT; 需要LVS和RS集群绑定同一个VIP（支持多数系统，不可以实现端口映射) TUNL：隧道技术，前端的Master只处理客户端的请 求，将请求转发给RealServer，然后由后台的RealServer直接响应客户端，不再经过Master；（支持少数系统，不可以实现端口映射)）

## 第3集 业界主流的高可用方案 keepalived讲解

### 简介：讲解业界主流高可用方案keepalived讲解

什么是**keepalived**

核心：监控并管理 LVS 集群系统中各个服务节点的状态

keepalived是一个类似于交换机制的软件,核心作用是检测服务器的状态，如果有一台web服务器工作出现故障， Keepalived将检测到并将有故障的服务器从系统中剔除，使用其他服务器代替该服务器的工作，当服务器工作正常后Keepalived自动将服务器加入到服务器群中，这些工作全部自

动完成。

后来加入了vrrp(虚拟路由器冗余协议)，除了为lvs提供高可用还可以为其他服务器比如Mysql、Haproxy等软件提供高可用

方案

安装

yum install -y keepalived

#路径

cd /etc/keepalived

启动和查看命令

#启动

service keepalived start

#停止

service keepalived stop

#查看状态

service keepalived status

#重启

service keepalived restart

#停止防火墙

systemctl stop firewalld.service

注意: 如果有缺少依赖可以执行下面的命令

|  |  |  |  |
| --- | --- | --- | --- |
| yum | install | -y | gcc |
| yum | install | -y | openssl-devel |
| yum | install | -y | libnl libnl-devel |
| yum | install | -y | libnfnetlink-devel |
| yum | install | -y | net-tools |
| yum | install | -y | vim wget |

## 第4集 Keepalived核心配置讲解

### 简介：Keepalived核心配置讲解

配置/etc/keepalived/keepalived.conf

! Configuration File for keepalived global\_defs {

router\_id LVS\_DEVEL # 设置lvs的id，在一个网络内

应该是唯一的

enable\_script\_security #允许执行外部脚本

}

#配置vrrp\_script，主要用于健康检查及检查失败后执行的动作。

vrrp\_script chk\_real\_server {

#健康检查脚本，当脚本返回值不为0时认为失败script

"/usr/local/software/conf/chk\_server.sh" #检查频率，以下配置每2秒检查1次

interval 2

#当检查失败后，将vrrp\_instance的priority减小5 weight -5

#连续监测失败3次，才认为真的健康检查失败。并调整优先级

fall 3

#连续监测2次成功，就认为成功。但不调整优先级rise 2

user root

}

#配置对外提供服务的VIP vrrp\_instance配置vrrp\_instance VI\_1 {

#指定vrrp\_instance的状态，是MASTER还是BACKUP主要还是看优先级。

state MASTER

#指定vrrp\_instance绑定的网卡，最终通过指定的网卡绑定VIP

interface ens33

#相当于VRID，用于在一个网内区分组播，需要组播域内内唯一。

virtual\_router\_id 51

#本机的优先级，VRID相同的机器中，优先级最高的会被选举为MASTER

priority 100

#心跳间隔检查，默认为1s，MASTER会每隔1秒发送一个报文告知组内其他机器自己还活着。

advert\_int 1

authentication { auth\_type PASS auth\_pass 1111

}

#定义虚拟IP(VIP)为192.168.159.100，可多设，每行一个virtual\_ipaddress {

192.168.159.100

}

#本vrrp\_instance所引用的脚本配置，名称就是vrrp\_script 定义的容器名

track\_script { chk\_real\_server

}

}

# 定义对外提供服务的LVS的VIP以及port virtual\_server 192.168.159.100 80 {

# 设置健康检查时间，单位是秒

delay\_loop 6

# 设置负载调度的算法为rr lb\_algo rr

# 设置LVS实现负载的机制，有NAT、TUN、DR三个模式

lb\_kind NAT

# 会话保持时间

persistence\_timeout 50

#指定转发协议类型(TCP、UDP) protocol TCP

# 指定real server1的IP地址

real\_server 192.168.159.146 80 { # 配置节点权值，数字越大权重越高weight 1

方式

接的端口

# 健康检查方式

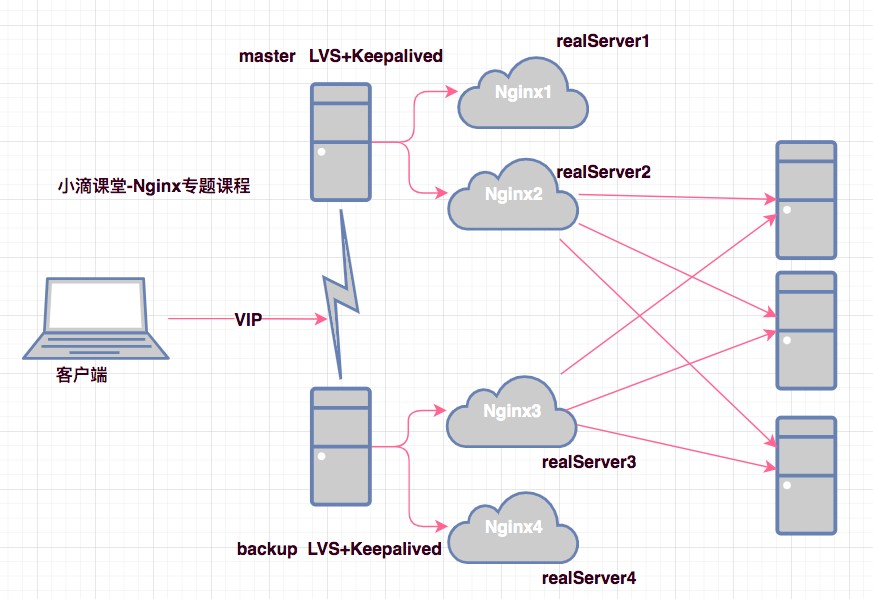
TCP\_CHECK { # 健康检查

connect\_timeout 10 # 连接超时retry 3 # 重试次数delay\_before\_retry 3 # 重试间隔connect\_port 80 # 检查时连

}

}

}



配置注意

router\_id后面跟的自定义的ID在同一个网络下是一致的

state后跟的MASTER和BACKUP必须是大写；否则会造成配置无法生效的问题

interface 网卡ID；要根据自己的实际情况来看，可以使用以下方式查询 ip a 查询

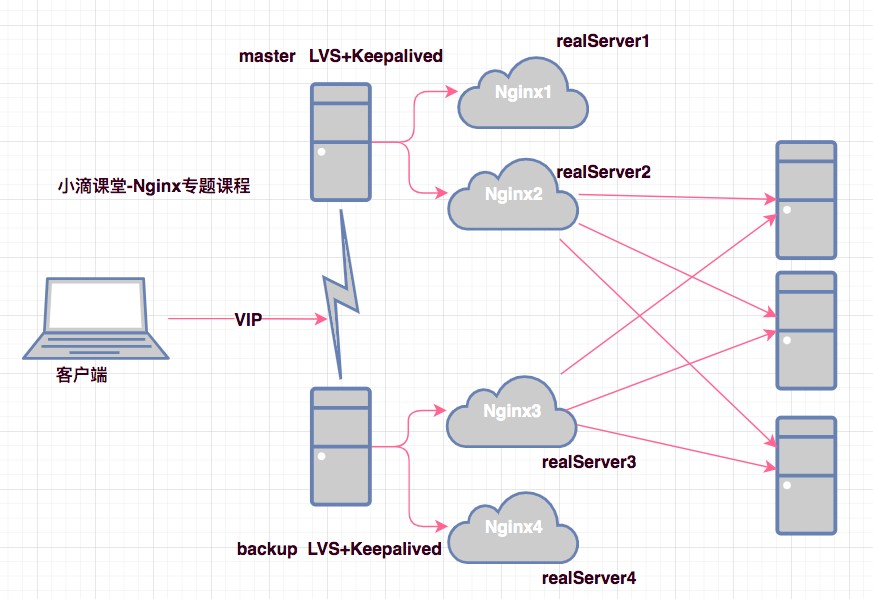
在BACKUP节点上，其keepalived.conf与Master上基本一致，修改state为BACKUP，priority值改小即可

authentication主备之间的认证方式，一般使用PASS即可；主备的配置必须一致，不能超过8位

## 第5集 架构系列之Nginx高可用方案相关环境准备

### 简介：准备Nginx+Lvs+KeepAlive相关软件环境

配置Nginx, 修改网页



启动keepalived

#启动

service keepalived start

#停止

service keepalived stop

#查看状态

service keepalived status

#重启

service keepalived restart

#停止防火墙

systemctl stop firewalld.service

## 第6集 架构系列之Nginx+LVS+KeepAlived方案实施《上》

### 简介：Nginx+Lvs+KeepAlive高可用方案实施《上》

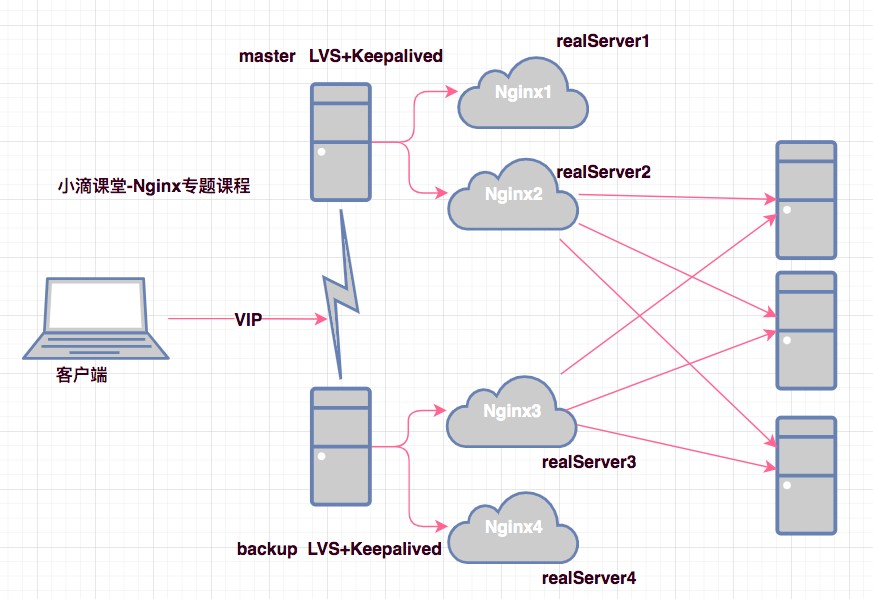
根据需求配置多个节点演示

如果其中keepalived挂了，那就会vip就会分发到另外

一个keepalived节点，响应正常

如果某个realServer挂了，比如是Nginx挂了，那对应

keepalived节点存活依旧可以转发过去，但是响应失败



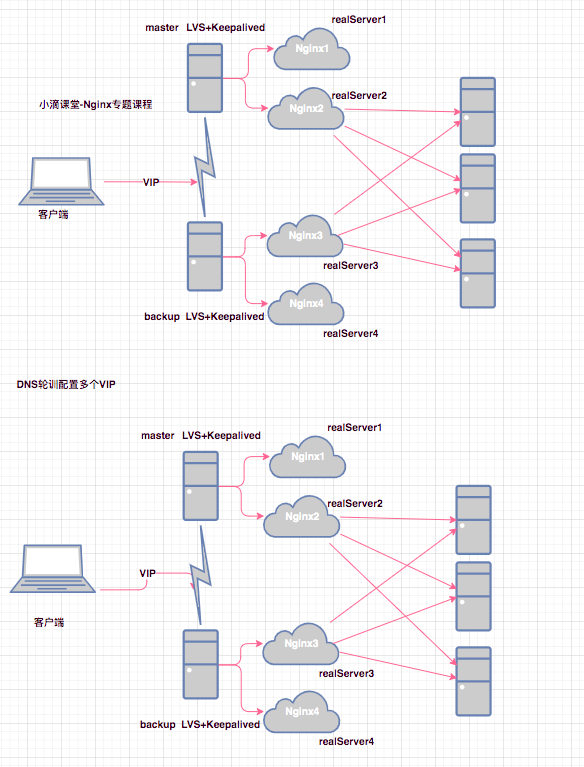
## 第7集 架构系列之Nginx+LVS+KeepAlived方案实施《下》

### 简介：Nginx+Lvs+KeepAlive高可用方案实施《下》

解决问题

如果某个realServer挂了，比如是Nginx挂了，那对应

keepalived节点存活依旧可以转发过去，但是响应失败



脚本监听

#配置vrrp\_script，主要用于健康检查及检查失败后执行的动作。

vrrp\_script chk\_real\_server {

#健康检查脚本，当脚本返回值不为0时认为失败script

"/usr/local/software/conf/chk\_server.sh" #检查频率，以下配置每2秒检查1次

interval 2

#当检查失败后，将vrrp\_instance的priority减小5 weight -5

#连续监测失败3次，才认为真的健康检查失败。并调整优先级

fall 3

#连续监测2次成功，就认为成功。但不调整优先级rise 2

user root

}

chk\_server.sh脚本内容（需要 chmod +x chk\_server.sh）

#!/bin/bash

#检查nginx进程是否存在

counter=$(ps -C nginx --no-heading|wc -l) if [ "${counter}" -eq "0" ]; then

service keepalived stop

echo 'nginx server is died. '

fi

常见问题

vip能ping通，vip监听的端口不通: 第一个原因:nginx1和

nginx2两台服务器的服务没有正常启动

vip ping不通: 核对是否出现裂脑,常见原因为防火墙配置所

致导致多播心跳失败,核对keepalived的配置是否正确

特别注意： 需要关闭selinux，不然sh脚本可能不生效

getenforce 查看

setenforce 0 关闭

生产环境问题

VIP : 阿里云(LBS)、华为云、腾讯云、AWS

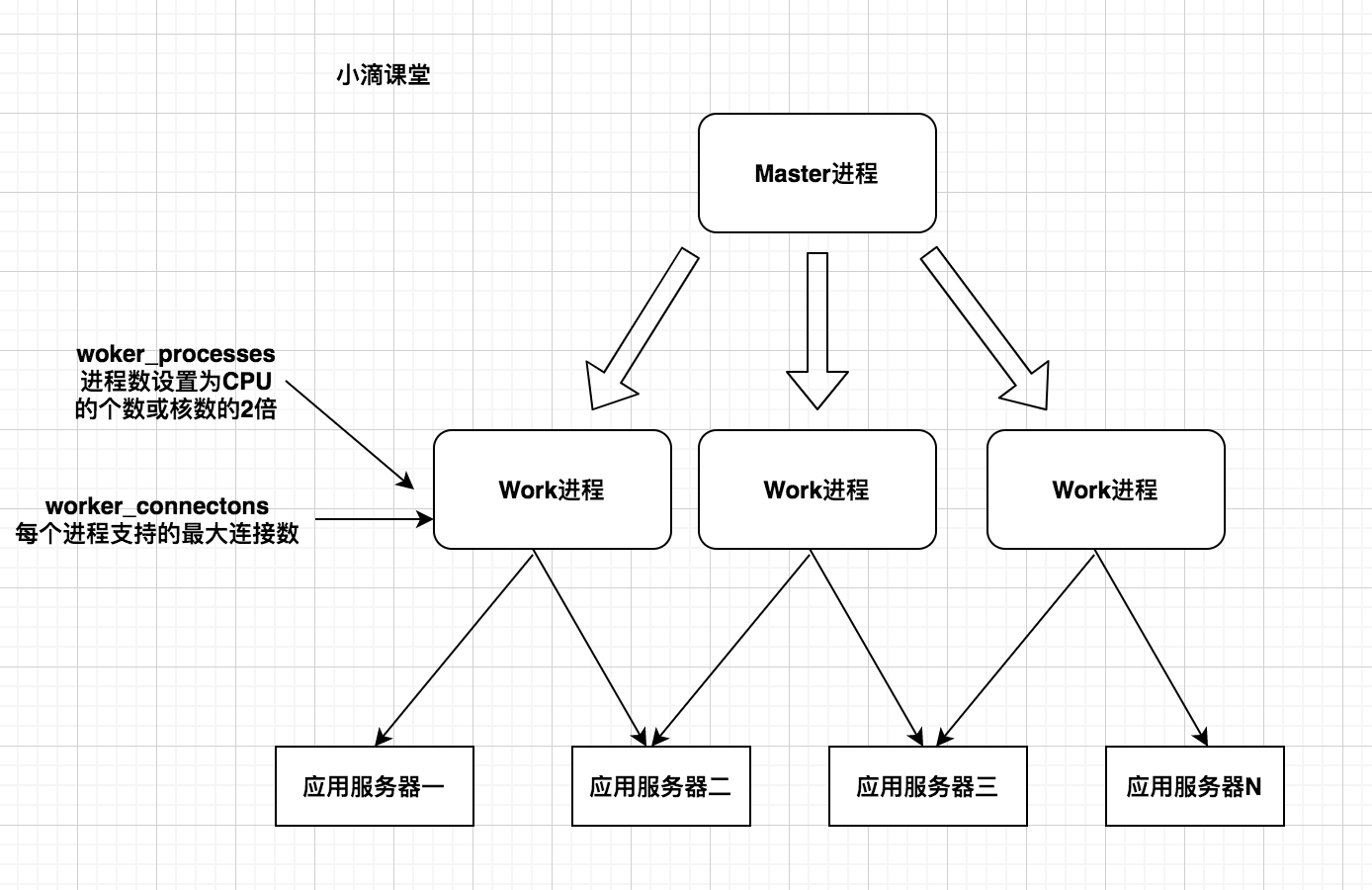
 **愿景："让编程不再难学，让技术与生活更加有趣" 更多架构课程请访问 xdclass.net**

# 第十三章 Nginx高性能原理和课程总结

## 第1集 Nginx基础架构master-worker进程剖析

### 简介：Nginx基础架构模型剖析

master 进程负责管理 Nginx 本身和其他 worker 进程

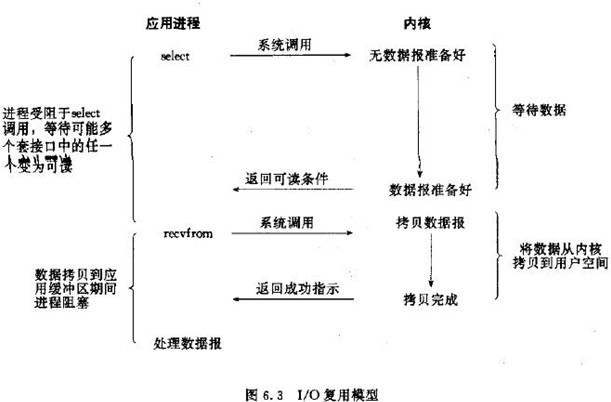


高性能原理

nginx 通过 多进程 + io多路复用（epoll） 实现了高并发

采用多个worker 进程实现对 多cpu 的利用 通过eopll

对 多个文件描述符 事件回调机制



拓展：linux I/O多路复用有select，poll，epoll

I/O模式一般分为同步IO和异步IO。

同步IO会阻塞进程，异步IO不会阻塞进程。

目前linux上大部分用的是同步IO，异步IO在linux上还不太成熟(有部分)

同步IO又分为阻塞IO，非阻塞IO，IO多路复用, 很多人对这个就有疑问了？？？？

同步IO会阻塞进程，为什么也包括非阻塞IO？ 因为非阻塞IO虽然在请求数据时不阻塞，但真正数据来临时，也就是内核数据拷贝到用户数据时，此时进程是阻塞的。

推荐书籍《Unix网络编程》

推荐课程《高并发架构-Netty百万连接实战》

## 第2集 高并发Nginx课程总结和学习路线

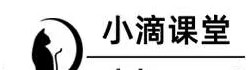
### 简介：Nginx课程总结和学习路线推荐

课程目录回顾

公司里面谁负责Nginx？ 开发？运维？

DevOps

学习路线推荐

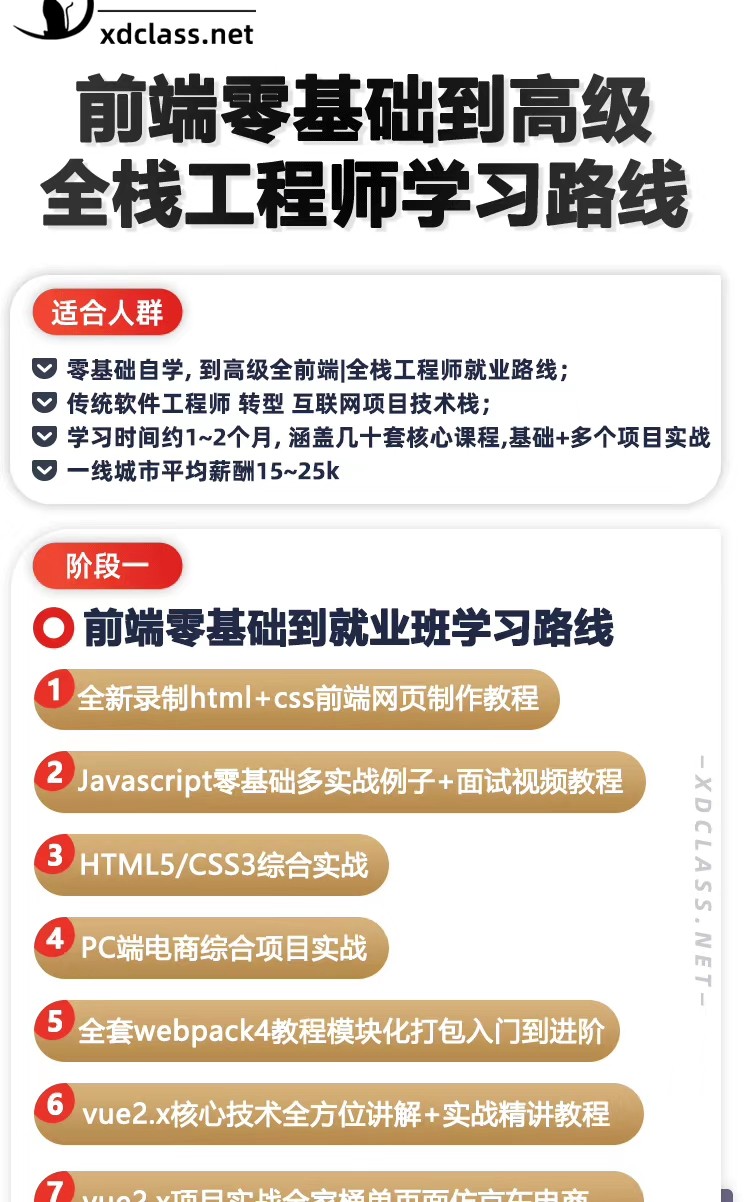








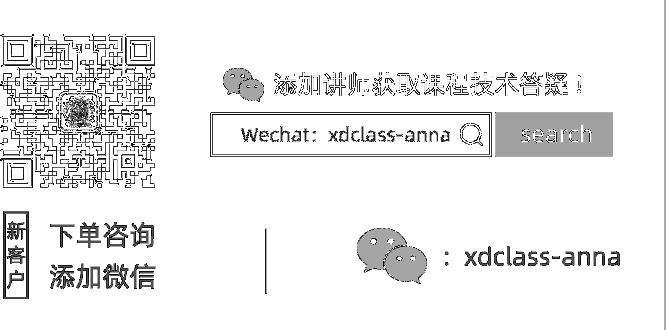












### 小滴课堂，愿景：让编程不在难学，让技术与生活更加有趣

**相信我们，这个是可以让你学习更加轻松的平台，里面的课程绝对会让你技术不断提升**

**欢迎加小D讲师的微信： xdclass-lw** 我们官方网站：[https://xdclass.net](https://xdclass.net/) **千人IT技术交流QQ群： 718617859**

重点来啦：加讲师微信 免费赠送你干货文档大集合，包含前

端，后端，测试，大数据，运维主流技术文档（持续更新）

<https://mp.weixin.qq.com/s/qYnjcDYGFDQorWmSfE7lpQ>