

Nginx 基础使用

目录结构

- conf
- html
- sbin

基本运行原理

Nginx配置与应用场景

最小配置

```
worker_processes  
worker_connections  
include mime.types;  
default_type application/octet-stream;  
sendfile on;  
keepalive_timeout 65;  
server
```

虚拟主机

- servername匹配规则
- 完整匹配
- 通配符匹配
- 通配符结束匹配
- 正则匹配

反向代理

反向代理

基于反向代理的负载均衡

负载均衡策略

轮询

weight(权重)

- ip_hash

- least_conn

- url_hash

- fair

动静分离

- 配置反向代理

- 增加每一个location

- 使用一个location

- alias与root

UrlRewrite

- rewrite语法格式及参数语法:

- 同时使用负载均衡

- 应用服务器防火墙配置

- 开启防火墙

- 重启防火墙

- 重载规则

- 查看已配置规则

- 指定端口和ip访问

- 移除规则

- 网关配置

防盗链配置

- 使用curl测试

- 带引用

高可用配置

- 安装Keepalived
 - 编译安装
 - yum安装
- 配置
 - 最小配置
- Https证书配置
 - 不安全的http协议
- openssl
 - 自签名
 - OpenSSL
 - 图形化工具 XCA
- CA 签名

Nginx 基础使用



目录结构

进入Nginx的主目录我们可以看到这些文件夹

```
client_body_temp  conf  fastcgi_temp  html  logs  proxy_temp  sbin  scgi_temp  uwsgi_temp
```

其中这几个文件夹在刚安装后是没有的，主要用来存放运行过程中的临时文件

```
client_body_temp fastcgi_temp proxy_temp scgi_temp
```

conf

用来存放配置文件相关

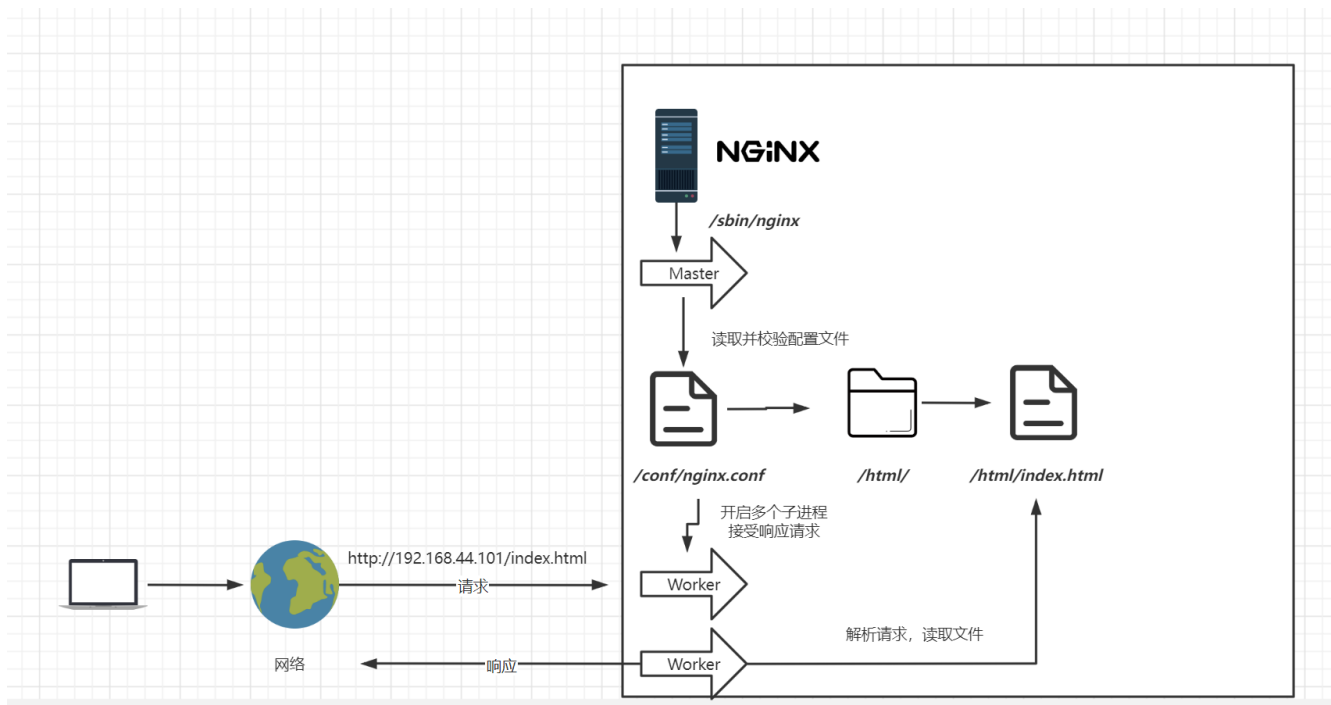
html

用来存放静态文件的默认目录 html、css等

sbin

nginx的主程序

基本运行原理



Nginx配置与应用场景

最小配置

worker_processes

`worker_processes 1;` 默认为1, 表示开启一个业务进程

worker_connections

`worker_connections 1024;` 单个业务进程可接受连接数

include mime.types;

`include mime.types;` 引入http mime类型

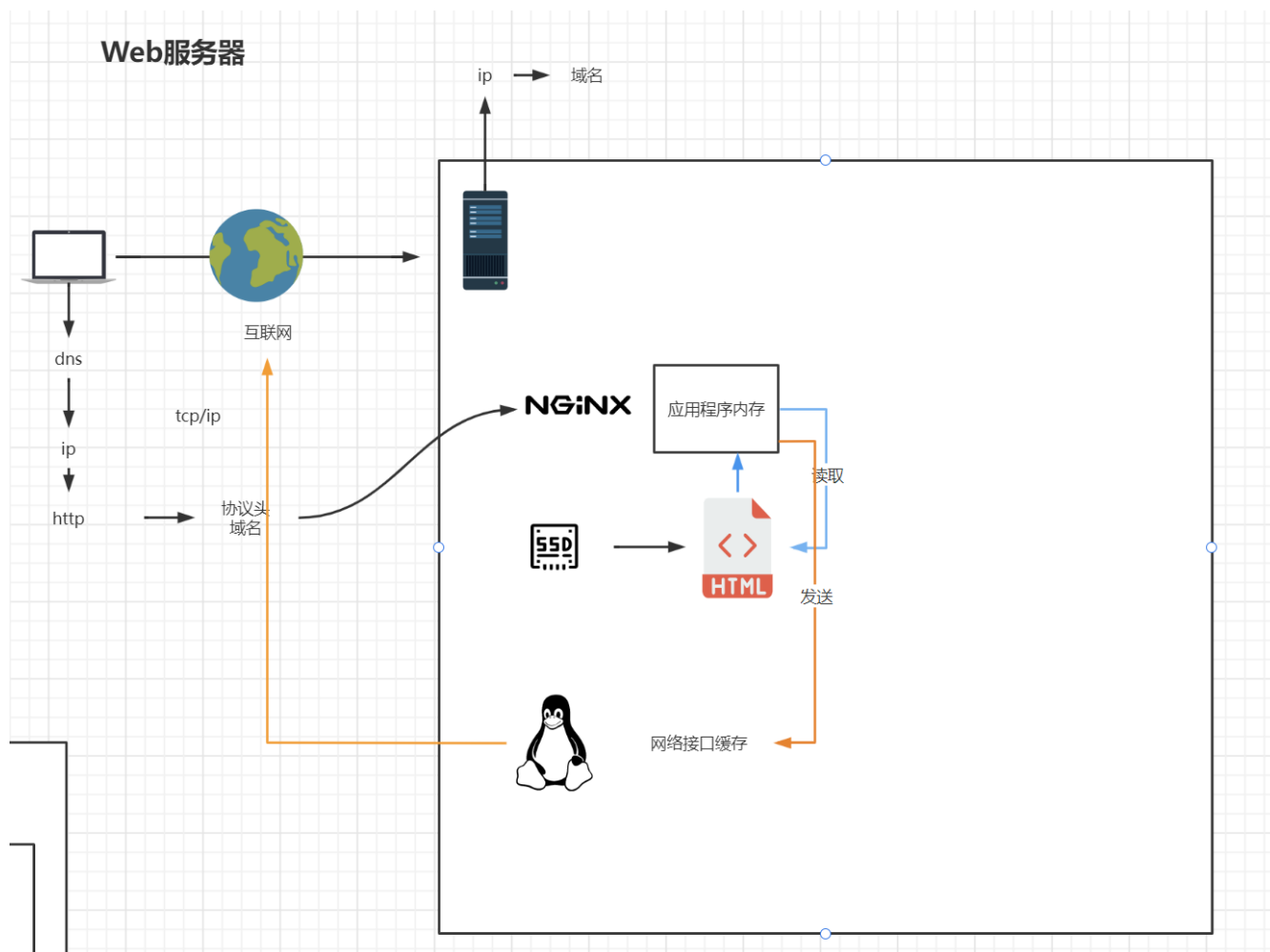
default_type application/octet-stream;

`default_type application/octet-stream;` 如果mime类型没匹配上, 默认使用二进制流的方式传输。

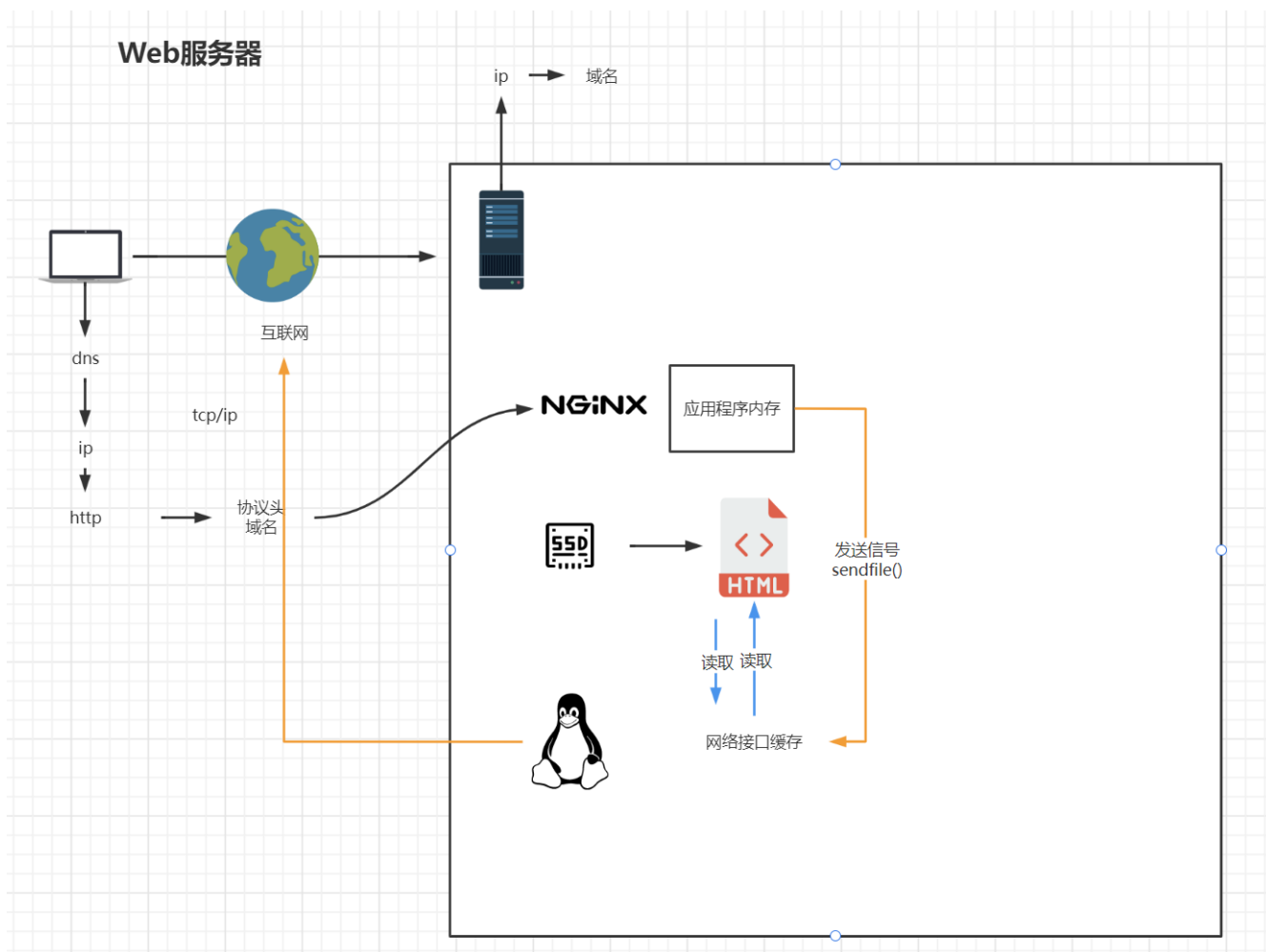
sendfile on;

`sendfile on;` 使用linux的 `sendfile(socket, file, len)` 高效网络传输, 也就是数据0拷贝。

未开启sendfile



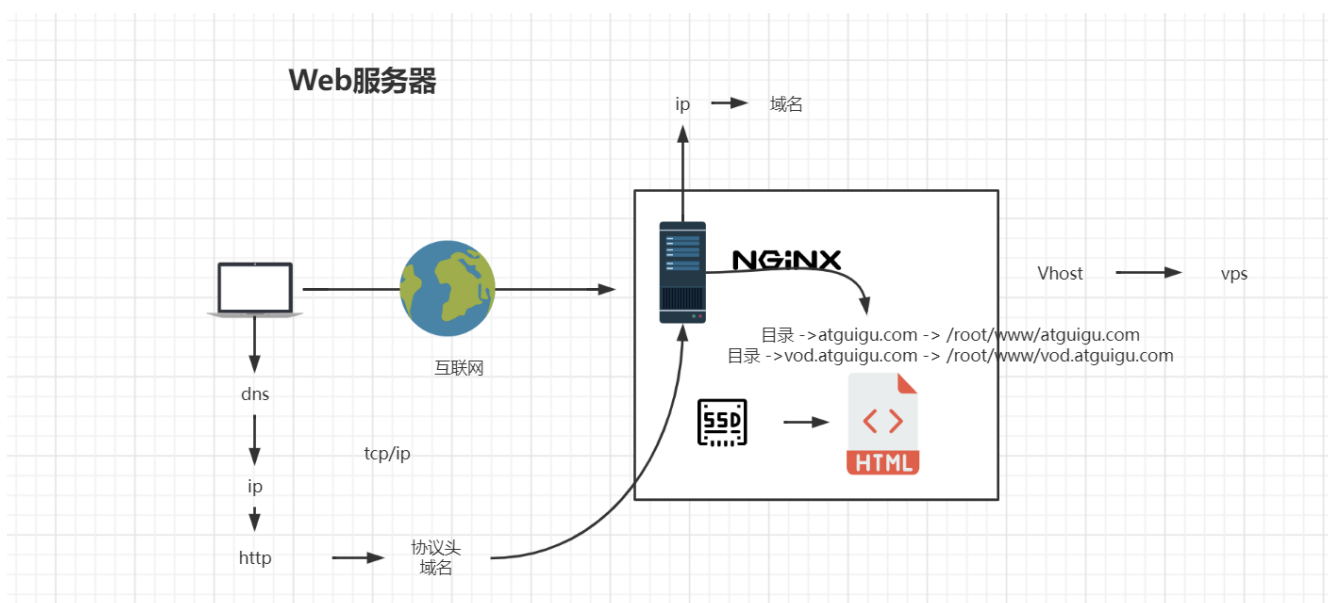
开启后



keepalive_timeout 65;

```
keepalive_timeout 65;
```

server



虚拟主机配置

```
server {  
    listen      80; 监听端口号  
    server_name localhost; 主机名  
    location / { 匹配路径  
        root    html; 文件根目录  
        index   index.html index.htm; 默认页名称  
    }  
  
    error_page  500 502 503 504  /50x.html; 报错编码对应页面  
    location = /50x.html {  
        root    html;  
    }  
}
```

虚拟主机

原本一台服务器只能对应一个站点，通过虚拟主机技术可以虚拟化成多个站点同时对外提供服务

servername匹配规则

我们需要注意的是servername匹配分先后顺序，写在前面的匹配上就不会继续往下匹配了。

完整匹配

我们可以在同一servername中匹配多个域名

```
server_name  vod.mmban.com www1.mmban.com;
```

通配符匹配

```
server_name  *.mmban.com
```

通配符结束匹配

```
server_name  vod.*;
```

正则匹配

```
server_name  ~^[0-9]+\..mmban\.com$;
```

反向代理

反向代理

```
proxy_pass http://baidu.com;
```

```
location / {  
  
    proxy_pass http://atguigu.com/;  
}
```

基于反向代理的负载均衡

```
upstream httpd {  
    server 192.168.44.102:80;  
    server 192.168.43.103:80;  
}
```

负载均衡策略

轮询

默认情况下使用轮询方式，逐一转发，这种方式适用于无状态请求。

weight(权重)

指定轮询几率，weight和访问比率成正比，用于后端服务器性能不均的情况。

```
upstream httpd {  
    server 127.0.0.1:8050    weight=10 down;  
    server 127.0.0.1:8060    weight=1;  
    server 127.0.0.1:8060    weight=1 backup;  
}
```

- down：表示当前的server暂时不参与负载
- weight：默认为1.weight越大，负载的权重就越大。
- backup：其它所有的非backup机器down或者忙的时候，请求backup机器。

ip_hash

根据客户端的ip地址转发同一台服务器，可以保持回话。

least_conn

最少连接访问

url_hash

根据用户访问的url定向转发请求

fair

根据后端服务器响应时间转发请求

动静分离

配置反向代理

```
location / {  
    proxy_pass http://127.0.0.1:8080;  
    root    html;  
    index  index.html index.htm;  
}
```

增加每一个location

```
location /css {  
  
    root    /usr/local/nginx/static;  
    index  index.html index.htm;  
}  
location /images {  
  
    root    /usr/local/nginx/static;  
    index  index.html index.htm;  
}  
  
location /js {  
  
    root    /usr/local/nginx/static;  
    index  index.html index.htm;  
}
```

使用一个location

使用正则

location 前缀

/ 通用匹配，任何请求都会匹配到。

= 精准匹配，不是以指定模式开头

~ 正则匹配，区分大小写

~* 正则匹配，不区分大小写

^~ 非正则匹配，匹配以指定模式开头的location

location匹配顺序

- 多个正则location直接按书写顺序匹配，成功后就不会继续往后面匹配
- 普通（非正则）location会一直往下，直到找到匹配度最高的（最大前缀匹配）
- 当普通location与正则location同时存在，如果正则匹配成功，则不会再执行普通匹配
- 所有类型location存在时，“=”匹配 > “^~”匹配 > 正则匹配 > 普通（最大前缀匹配）

```
location ~*/(css|img|js) {  
  
    root    /usr/local/nginx/static;  
    index   index.html index.htm;  
}
```

alias与root

```
location /css {  
  
    alias    /usr/local/nginx/static/css;  
    index   index.html index.htm;  
}
```

root用来设置根目录，而alias在接受请求的时候在路径上不会加上location。

1) alias指定的目录是准确的，即location匹配访问的path目录下的文件直接是在alias目录下查找的；2) root指定的目录是location匹配访问的path目录的上一级目录，这个path目录一定要是真实存在root指定目录下的；3) 使用alias标签的目录块中不能使用rewrite的break（具体原因不明）；另外，alias指定的目录后面必须要加上"/"符号！！4) alias虚拟目录配置中，location匹配的path目录如果后面不带"/"，那么访问的url地址中这个path目录后面加不加"/"不影响访问，访问时它会自动加上"/"；但是如果location匹配的path目录后面加上"/"，那么访问的url地址中这个path目录必须要加上"/"，访问时它不会自动加上"/"。如果不加上"/"，访问就会失败！5) root目录配置中，location匹配的path目录后面带不带"/"，都不会影响访问。

UrlRewrite

rewrite语法格式及参数语法:

rewrite是实现URL重写的关键指令，根据regex（正则表达式）部分内容，重定向到replacement，结尾是flag标记。

rewrite	<regex>	<replacement>	[flag];
关键字	正则	替代内容	flag标记

关键字：其中关键字error_log不能改变

正则：perl兼容正则表达式语句进行规则匹配
替代内容：将正则匹配的内容替换成replacement
flag标记：rewrite支持的flag标记

rewrite参数的标签段位置：
server,location,if

flag标记说明：
last #本条规则匹配完成后，继续向下匹配新的location URI规则
break #本条规则匹配完成即终止，不再匹配后面的任何规则
redirect #返回302临时重定向，浏览器地址会显示跳转后的URL地址
permanent #返回301永久重定向，浏览器地址栏会显示跳转后的URL地址

实例

```
rewrite ^/([0-9]+).html$ /index.jsp?pageNum=$1 break;
```

同时使用负载均衡

应用服务器防火墙配置

开启防火墙

```
systemctl start firewalld
```

重启防火墙

```
systemctl restart firewalld
```

重载规则

```
firewall-cmd --reload
```

查看已配置规则

```
firewall-cmd --list-all
```

指定端口和ip访问

```
firewall-cmd --permanent --add-rich-rule="rule family="ipv4" source address="192.168.44.101" port protocol="tcp" port="8080" accept"
```

移除规则

```
firewall-cmd --permanent --remove-rich-rule="rule family="ipv4" source
address="192.168.44.101" port port="8080" protocol="tcp" accept"
```

网关配置

```
upstream httpds {

server 192.168.44.102 weight=8 down;
server 192.168.44.103:8080 weight=2;
server 192.168.44.104:8080 weight=1 backup;
}

location / {

rewrite ^/([0-9]+).html$ /index.jsp?pageNum=$1 redirect;
proxy_pass http://httpds ;
}
```

防盗链配置

```
valid_referers none | blocked | server_names | strings ....;
```

- none，检测 Referer 头域不存在的情况。
- blocked，检测 Referer 头域的值被防火墙或者代理服务器删除或伪装的情况。这种情况该头域的值不以“http://”或“https://”开头。
- server_names，设置一个或多个 URL，检测 Referer 头域的值是否是这些 URL 中的某一个。

在需要防盗链的location中配置

```
valid_referers 192.168.44.101;
if ($invalid_referer) {
    return 403;
}
```

使用curl测试

```
curl -I http://192.168.44.101/img/logo.png
```

带引用

```
curl -e "http://baidu.com" -I http://192.168.44.101/img/logo.png
```

高可用配置

安装Keepalived

编译安装

下载地址

```
https://www.keepalived.org/download.html#
```

使用 `./configure` 编译安装

如遇报错提示

```
configure: error:
  !!! OpenSSL is not properly installed on your system. !!!
  !!! Can not include openssl headers files.             !!!
```

安装依赖

```
yum install openssl-devel
```

yum安装

```
yum install keepalived
```

配置

使用yum安装后配置文件在

```
/etc/keepalived/keepalived.conf
```

最小配置

第一台机器

```
! Configuration File for keepalived

global_defs {

    router_id lb111

}

vrrp_instance atguigu {
    state MASTER
    interface ens33
    virtual_router_id 51
```

```
priority 100
advert_int 1
authentication {
    auth_type PASS
    auth_pass 1111
}
virtual_ipaddress {
    192.168.44.200
}
}
```

第二台机器

```
! Configuration File for keepalived

global_defs {

    router_id lb110

}

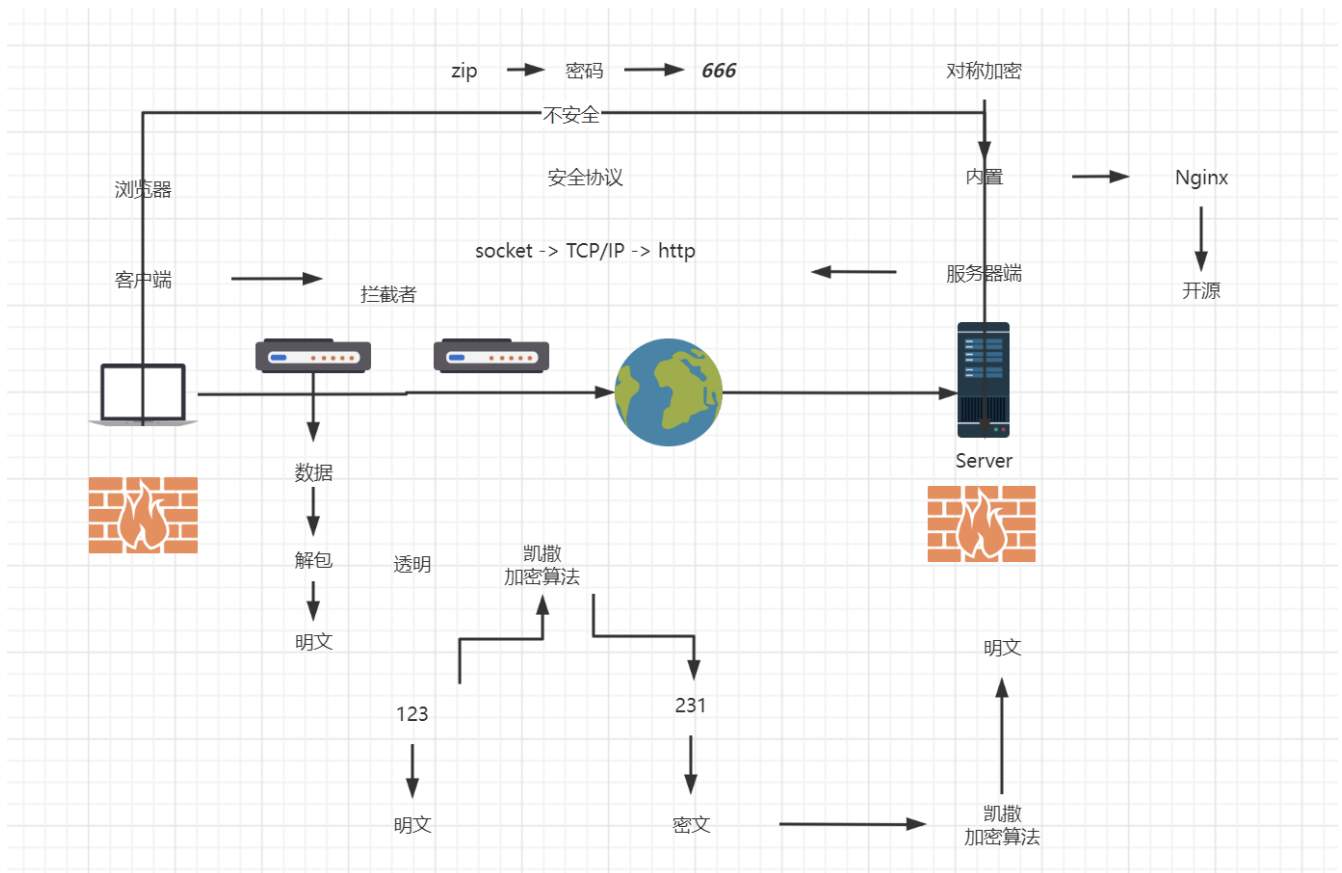
vrrp_instance atguigu {
    state BACKUP
    interface ens33
    virtual_router_id 51
    priority 50
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1111
    }
    virtual_ipaddress {
        192.168.44.200
    }
}
```

启动服务

```
systemctl start keepalived
```

Https证书配置

不安全的http协议



openssl

openssl包含：SSL协议库、应用程序以及密码算法库

自签名

OpenSSL

系统内置

图形化工具 XCA

下载地址

<https://www.hohnstaedt.de/xca/index.php/download>

CA 签名