宝兰德Web服务器软件 3.1.0用户手册

北京宝兰德软件股份有限公司
Beijing Baolande Software Corporation

版权所有 侵权必究
All rights reserved

目录

第1章 前 言

本文档是BES WebServer的用户手册，详细介绍BES WebServer的配置和管理。本手册的组织结构与管理控制台的布局基本对应，每章都以概念性信息开头，随后的部分说明如何使用管理控制台进行特定的操作。

本手册适合的对象

本手册主要适用于使用BES WebServer的系统管理员。

本手册假定您已经具备如下技能：

1. 操作系统的基础操作。
2. JDK的安装。

约定

BES WebServer定义了一些变量来表示BES WebServer目录等信息，本文档中涉及的有：

表 1-1 BES WebServer变量说明

变量	说明
BWS_HOME	文档中采用该值表示BES WebServer的安装目录，该变量实际上并不存在。
BWS	文档中采用该值表示BES WebServer的简称

技术支持

BES WebServer提供全方位的技术支持，获得技术支持的方式有：

网址：www.bessystem.com

Support Email：support@bessystem.com

Support Tel：400 650 1976

在取得技术支持时，请提供如下信息：

1. 姓名
2. 公司信息及联系方式
3. 操作系统及其版本
4. BES WebServer版本
5. 日志等错误的详细信息

第2章 产品介绍

2.1 关于BES WebServer

BES WebServer是一款高性能、稳定和安全的Web服务器。占用极少的内存资源，支持10万+的高并发连接，处理响应请求的速度非常快。支持主备部署模式，保障服务的高可用性，避免单点故障。提供包括轮询、权重、会话保持等多种负载均衡策略，满足业务的多种负载需求。对常见的安全攻击如SQL注入、DDOS攻击进行防护，保障业务的安全性。使用BES WebServer能帮助企业提高业务系统的可靠性、稳定性和安全性，有效降低运营成本，对外提供更优质的服务。BES WebServer使用常见WEB架构，如下图：

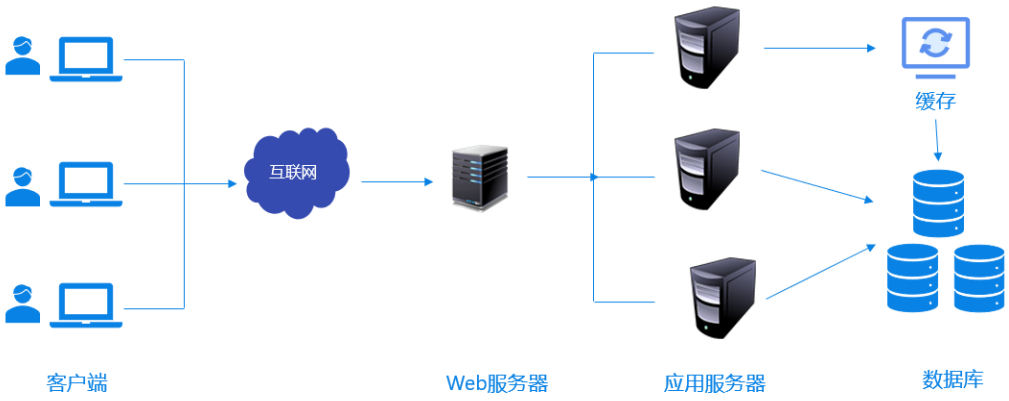


图 2-1 通用WEB架构

2.2 产品架构

产品架构如下：

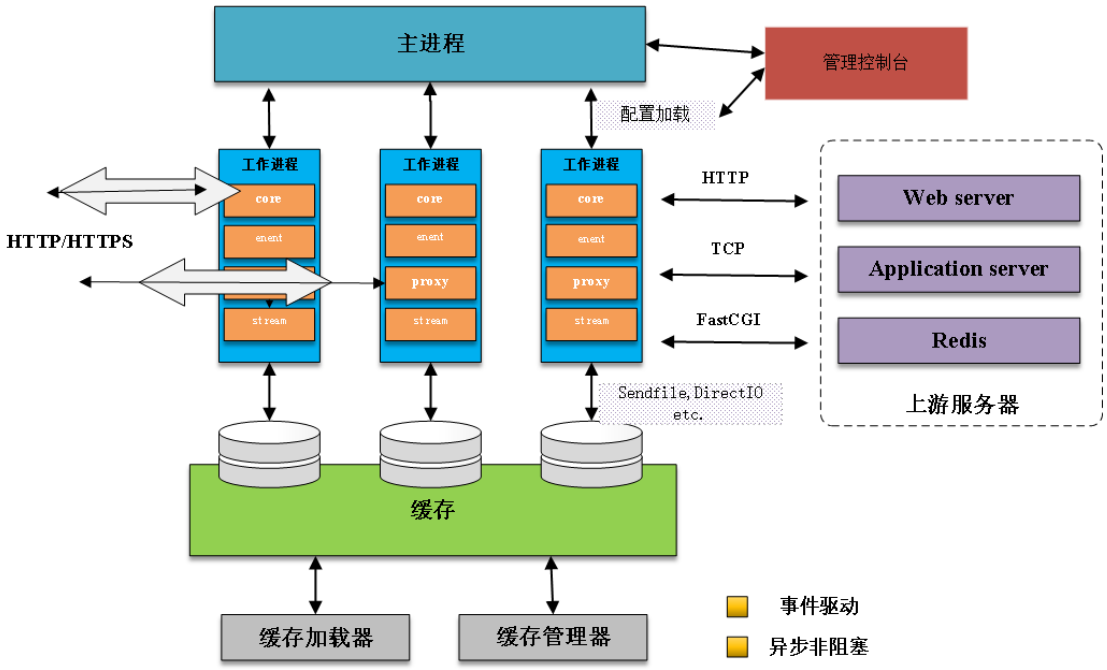


图 2-2 产品架构图

架构中各个组件简介如下:

主进程（Master Processor）

主要负责配置文件解析、维护工作进程和作为BWS整个进程组与用户的交互接口，接收和翻译用户的指令。不需要处理网络事件，不负责业务的执行。

工作进程（Worker Processor）

工作进程的主要任务是完成具体的任务逻辑。包含接受连接，解析客户端请求，处理响应内容，以及与上游服务器的交互等。

上游服务器（Upstream Server）

BWS作为反向代理角色时，BWS将客户请求转发给后端应用服务器、Servlet容器等，这些服务器对BWS提供响应内容。这些后端服务器称为上游服务器。

缓存加载器（Cache Loader）

只运行一次，BWS启动后，它将有关以前缓存的数据的元数据加载到共享内存区域。

缓存管理器 (Cache Manager)

周期性地启动，检查高速缓存的状态，负责清理过期缓存。

管理控制台(Admin Console)

主要负责配置BWS以及监控进程状态。

2.3 产品特性

- 基于IP 和名称的虚拟主机服务;
- 支持 keep-alive 和管道连接;
- 灵活便洁的配置方式;

- 更新配置和在线升级，无须中断客户的工作进程；
- 可定制的访问日志，日志写入缓存，以及快捷的日志轮转；
- 4xx-5xx 错误代码重定向；
- 基于 PCRE 的 rewrite 重写模块；
- 基于客户端 IP 地址和 HTTP 基本认证的访问控制；
- PUT、DELETE和MKCOL 方法；
- 支持 FLV （Flash 视频）；
- 支持带宽限制；

2.3.1 支持的平台环境

BES WebServer认证的平台环境如表所示：

表 2-1 BES WebServer认证的平台环境

操作系统	RedHat系列、SUSE系列、银河麒麟操作系统、深度操作系统、麒麟操作系统、一铭操作系统、统信操作系统.....
芯片	支持国内主流X86和ARM架构芯片：海光、华为鲲鹏、龙芯、飞腾、申威、兆芯.....
浏览器	IE：10.0+ Chrome：73.0+ Firefox：60+
国密算法	支持GMTLS1.1 ECC-SM2-WITH-SM4-SM3单向认证及ECDHE-SM2-WITH-SM4-SM3双向认证，支持国密HTTPS/传统HTTPS自适应

第3章 产品安装与注册

请您联系BES WebServer的销售代表，获取适用于您操作系统的安装包。

3.1 安装前检查

1) 卸载老版本的BES WebServer

建议在安装BES WebServer前先卸载老版本的BES WebServer，保留以前版本的BES WebServer可能会造成潜在的版本冲突，影响BES WebServer的正常运行。有关老版本BES WebServer的卸载步骤，请参考相应版本的安装手册。

2) 安装用户权限

在UNIX/Linux平台上安装BES WebServer，确保用户对安装目录具备读写权限。

3.2 解压产品安装包

```
mkdir /home/bes/BWS
tar -zxvf BES-WEBSERVER-STANDALONE-3.1.0.XXX.tar.gz  -C /home/bes/BWS
```

表 3-1 产品核心文件

文件名称	路径	说明
bws	BWS_HOME/bin	BES WebServer主程序
bws.sh	BWS_HOME/bin	主程序启动和停止脚本
keepalived	BWS_HOME/bin	高可用组件启动和停止脚本
lmadm	BWS_HOME/bin	授权文件查看与导入工具
patch	BWS_HOME/bin	补丁查看和安装工具
startconsole	BWS_HOME/bin	管理控制台启动脚本
stopconsole	BWS_HOME/bin	管理控制台停止脚本
bws.conf	BWS_HOME/conf	BES WebServer主程序主配置文件
application.yml	BWS_HOME/conf	管理控制台主配置文件
keepalived.conf	BWS_HOME/conf	高可用组件配置文件
server.key	BWS_HOME/conf/security	普通RSA证书私钥（证书仅测试用）
server.crt	BWS_HOME/conf/security	普通RSA证书
server-sig.key	BWS_HOME/conf/security	国密签名证书私钥
server-sig.crt	BWS_HOME/conf/security	国密签名证书
server-enc.key	BWS_HOME/conf/security	国密加密证书私钥
server-enc.crt	BWS_HOME/conf/security	国密加密证书
bws-*.jar	BWS_HOME/console/lib	管理控制台jar包

3.3 启动与停止

用户在管理控制台进行BES WebServer的启动、停止和重加载操作，也可以使用命令行进行操作。

命令行脚本位于BWS_HOME/bin目录下。支持如下参数：

```
[bes@Linux17209 bin]$ ./bws.sh -h
BES WebServer version: 3.1.0.1099
Usage: BES WebServer [-?hvVtTq] [-s signal] [-p prefix]
             [-e filename] [-c filename] [-g directives]

Options:
  -?,-h         : this help
  -v            : show version and exit
  -V            : show version and configure options then exit
  -t            : test configuration and exit
  -T            : test configuration, dump it and exit
  -q            : suppress non-error messages during configuration testing
  -s signal     : send signal to a master process: stop, quit, reopen, reload
  -p prefix     : set prefix path (default: ./bes-webserver/)
  -e filename   : set error log file (default: logs/error.log)
  -c filename   : set configuration file (default: conf/bws.conf)
  -g directives : set global directives out of configuration file
```

参数说明：

```
-?或-h              ----打印命令行参数帮助信息
-c file             ----使用指定的配置文件来替代默认的配置文件bws.conf
-v                  ----打印BES WebServer版本。
-V                  ----打印BES WebServer版本，编译器版本和配置参数。
-t                  ----测试配置文件：检查配置文件的语法正确性，
↪  之后尝试打开配置中引用到的文件。
-T                  ----与-t一样，但另外将配置文件转储到标准输出。
-q                  ----在测试配置文件时禁止非错误信息。
-s  signal          ----向 Master进程发送信号，信号参数如下：
    stop            ----快速关闭，不等待工作进程处理完当前请求，
↪  强制停止BES WebServer进程。
    quit            ----正常关闭，等待工作进程处理完当前请求之后，
↪  再停止BES WebServer进程。
    reload          ----重新加载配置，
↪  使用新配置后启动新的Worker进程并正常退出旧的工作进程。
    reopen          ----重新打开日志文件。
-p  prefix          ----设置BES WebServer路径前缀，比如一个存放着服务器文件的目录
↪  （默认是BWS_HOME）。
-e  filename        ----设置BES WebServer错误日志位置，默认是logs/error.log。
-c  filename        ----设置BES WebServer配置文件路径，默认是conf/bws.conf。
-g  directive       ----设置全局配置指令，例如：
↪  ./bws.sh  -g  "pid    BWS_HOME/bin/.pid;"
```

使用场景如下：

（1）停止BES WebServer进程，执行命令：

```
./bws.sh -s quit
```

quit表示用户要等待工作进程处理完当前的请求才停止进程，stop 表示不需要等待工作进程处理完当前的请求，强行停止BES WebServer进程。执行此命令时应确保执行用户和启动BES WebServer的用户一致。

（2）重新加载配置文件，在没有执行重加载命令之前，配置文件所做的内容更改将不会生效，要重新加载配置，需要执行命令:

```
./bws.sh -s reload
```

一旦主进程（Master）收到要重新加载配置（reload）的信号，它将检查新配置文件的语法有效性，并尝试应用新的配置。如果成功，主进程将启动新的工作进程（Worker），并向旧工作进程发送消息，请求它们关闭。否则，主进程回滚更改，并继续使用旧配置。旧工作进程接收到关闭命令后，停止接受新的请求连接，并继续维护当前请求，直到这些请求都被处理完成之后，旧工作进程将退出。

用户也可以使用kill命令将信号发送到BES WebServer进程，信号直接发送到指定进程PID。默认情况下，BES WebServer主进程的PID在BWS_HOME/logs/bws.pid文件中。例如，发送QUIT信号给主进程，让BES WebServer正常平滑关闭，执行命令:

```
kill -s QUIT <main process id>
```

3.4 主配置文件说明

BES WebServer的主配置文件是bws.conf，位于产品的BWS_HOME/conf目录，主配置文件中包含注释和指令控制模块。注释以#开头，指令分为简单指令和块指令。简单指令是由空格分隔的名称和参数组成，并以分号结尾。块指令具有与简单指令相同的结构，但不是以分号结尾，而是以大括号 {} 包围的一组附加指令结尾。

3.4.1 注释

```
# user nobody
```

3.4.2 简单指令

```
worker_processes 1;
```

3.4.3 块指令

```
server {
    listen 8080;
    server_name localhost;
}
```

3.4.4 整体结构

主配置文件整体结构如下:

```
...                                     #全局块

events {                                #events块
     ...
}


http {                                  #http块
    ...                                 #http全局块
    server {                            #server块
         ...                            #server全局块
        location / {                    #location块
            ...
            proxy_pass http://bes;
        }
    }
    upstream bes {                      #upstream块
        ...
    }

}
stream {                                #stream块
    ...                                 #stream全局块
    server {                            #server块
        ...
        proxy_pass mysql;
    }
}
```

全局块：配置影响BES WebServer全局的参数。包括运行BES WebServer的用户组，进程pid存放路径，日志文件存放路径，配置文件引入等。

events块：配置影响BES WebServer服务器与用户的网络连接。每个进程的最大连接数，选取哪种事件驱动模型处理连接请求，允许同时接受多少个连接，开启多个网络连接串行化等。

http块：HTTP代理服务相关配置模块，可以嵌套多个server、upstream，配置代理、缓存、日志等绝大多数功能和第三方模块的配置。如文件引入，mime-type定义，日志自定义，是否使用sendfile传输文件，连接超时时间等。

stream块：TCP、UDP四层代理服务配置模块，同http模块一样，可嵌套配置多个server、upstream。

server块：配置虚拟主机相关参数，可配置在http和stream模块下，支持配置多个server。

upstream块：http、stream中负载均衡相关配置模块

location块：配置请求的路由，及各种页面的处理情况。

3.4.5 指令继承与覆盖

通常子指令继承其父级指令块包含的指令设置。部分指令可以出现在多个指令块当中，在这种情况下，可以通过在子指令块中增加相同指令来达到覆盖父级指令的目的，例如：

```
http {
    include mime.types;
    sendfile on;
    keepalive_timeout 65;
    server {
        listen 8080;
        server_name server1;
        keepalive_timeout 30;
    }
    server {
        listen 8089;
        server_name server2;
    }
}
```

上述配置中，虚拟主机server1，配置了keepalive_timeout为30秒，覆盖了继承自http块中配置的keepalive_timeout配置。虚拟主机server2中继承了http块的配置，keepalive_timeout生效的依然是65s。

3.5 产品注册

BES WebServer安装完成后自带的License是试用版本，用户必须申请正式的产品许可来激活产品，才能正常使用BES WebServer。

1. BES WebServer提供的License包括以下几种：

1) 试用版本：在使用一段时间后自动过期，用户将无法使用产品。

2) 正式版本：不过期，支持用户永久使用的版本。

2. 产品注册具体步骤为：

1) 用户必须向北京宝兰德软件股份有限公司申请License。

2) 用户运行lmadm import-lic命令将收到的License文件导入**BES WebServer**中，假设用户将License文件放在/home/bes/目录下，在BWS_HOME/bin下通过下列命令导入License：

```
[bes@Linux17209 bin]$ ./lmadm import-lic --sourcepath=/home/bes/bws.lic.txt
License has been imported into /home/bes/bws/license/bws.lic successfully.
```

3) License导入成功后，通过lmadm view-lic命令读取License文件：

```
[bes@Linux17209 bin]$ ./lmadm view-lic
Customer.name=BES TRIAL
Product.name=BES WebServer
Product.version=3.1.0
Register.type=DEVELOPMENT
Expire.date=2022-10-08
Expire.interval=183 days
Serial.number=LWMX-88A8UB-L63KM4-2L2M
Register.description=
```

第4章 管理中心

4.1 关于管理中心

BES WebServer管理中心是一个基于WEB浏览器的图形化管理工具，用户通过管理中心对BES WebServer提供的虚拟主机、HTTP参数、上游、区域、证书等进行配置和管理。

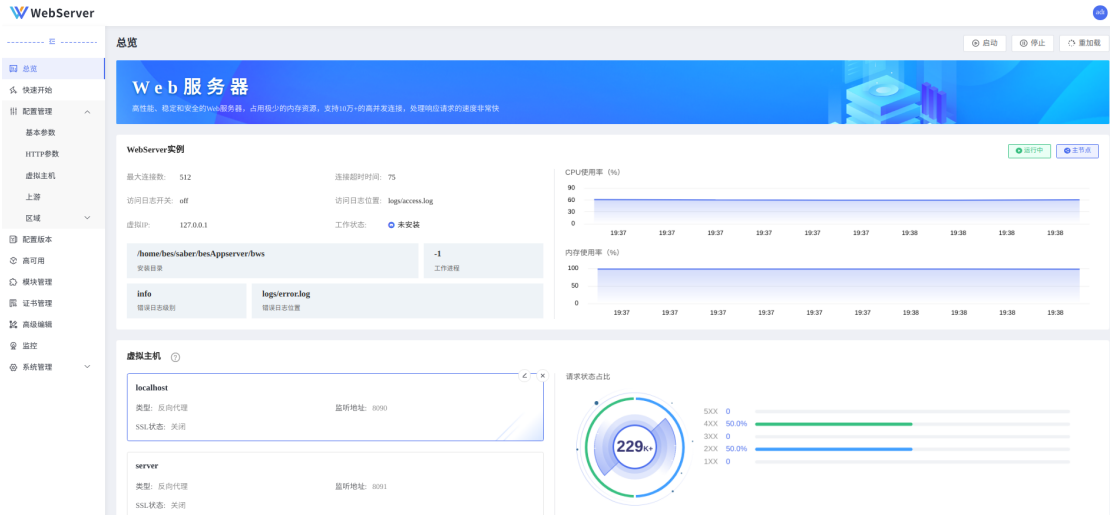


图 4-1 管理控制台

4.1.1 启动管理中心

BWS_HOME/bin下执行startconsole启动管理控制台。

```
[bes@Linux17209 bin]$ ./startconsole
Starting bws...
More information see /home/bes/bws/logs/bws-console.log
```

日志中出现“Started BWSApplication in XXX seconds”即代表管理中心启动成功。

4.1.2 登录管理中心

BES WebServer管理中心提供用户登录，对用户身份进行验证。

管理中心访问方式：

在浏览器中输入http://hostName:port/console以访问管理中心。其中hostName是服务器所在机器的主机名，或是服务器所在机器的IP地址；port是管理中心的管理端口，默认为5900。

1. BES WebServer提供的初始用户名和密码分别是admin、B#2008_2108#es。
2. 建议在初次登录成功后及时修改密码，或是按需新建用户。
3. 用户管理的具体操作请参看“用户管理”章节。

管理中心同时支持安全和国密访问，用户只需要修改BWS_HOME/conf/application.yml文件中下述配置，然后使用支持国密的安全浏览器访问即可。

```
ssl:
  enabled: false                                          --->是否开启安全
  clientAuth: want                                        --->是否开启客户端验证,
  ↪  可选值need、want、none
  ciphers: HIGH:!aNULL:!eNULL:!EXPORT:!DES:!RC4:!MD5
  keyAlias: bes                                           --->证书昵称
  keyPassword: '{AES}PrzBD+FLE0Wheq7AAaghXw=='            --->密钥密码
  keyStore: ${com.bes.installRoot}/conf/security/keystore.jks --->密钥库路径
  keyStorePassword: '{AES}PrzBD+FLE0Wheq7AAaghXw=='               --->密钥库密码
  keyStoreType: jks                                               --->密钥库类型
  trustStore: ${com.bes.installRoot}/conf/security/cacerts.jks--->信任库路径
  trustStorePassword: '{AES}PrzBD+FLE0Wheq7AAaghXw=='             --->信任库密码
  trustStoreType: JKS                                             --->信任库类型
  trustStoreProvider: SUN
  keyStoreProvider: SUN
  protocol: TLSv1                                                 --->安全协议
gmssl-enabled: false                                              --->是否开启国密
 bes:
  gmssl:
    key-alias: bes                                                --->国密证书昵称
    key-password: '{AES}PrzBD+FLE0Wheq7AAaghXw=='                 --->国密证书密码
    key-store-file: ${com.bes.installRoot}/conf/security/keystore.bks  --->国」
    ↪  密证书库路径
    key-store-password: '{AES}PrzBD+FLE0Wheq7AAaghXw=='           --->国密证书库密码
```

以360安全浏览器为例，如果用户要使用国密模式访问，按照如下步骤操作，此处使用360国密专版浏览器作为说明：

1) 修改产品BWS_HOME/conf/application.yml文件中gmssl-enabled为true，即可启用国密模式。
2) 在BWS_HOME/bin下执行gmkeytool –exportpem生成证书。新建一个ctl.dat的文件，将证书内容复制到ctl.dat文件中。
3) 使用国密360专版浏览器，将ctl.dat文件放到360浏览器安装路径的Application/User Data/Default/ctl目录下。（没有目录的话，需要手动建立）
4) 使用360国密专版浏览器访问管理中心https://IP:5900/console。

4.2 快速开始

为了方便用户快速配置并使用产品，BES WebServer提供了快速开始功能，只保留核心的、最常用的一些属性，快速创建出一个虚拟主机和多个代理目标，在简单场景下，客户可以快速完成配置定义工作。

快速开始模块大部分内容都和配置管理重合，只是更简洁，所以详细解释请参考“配置管理”。

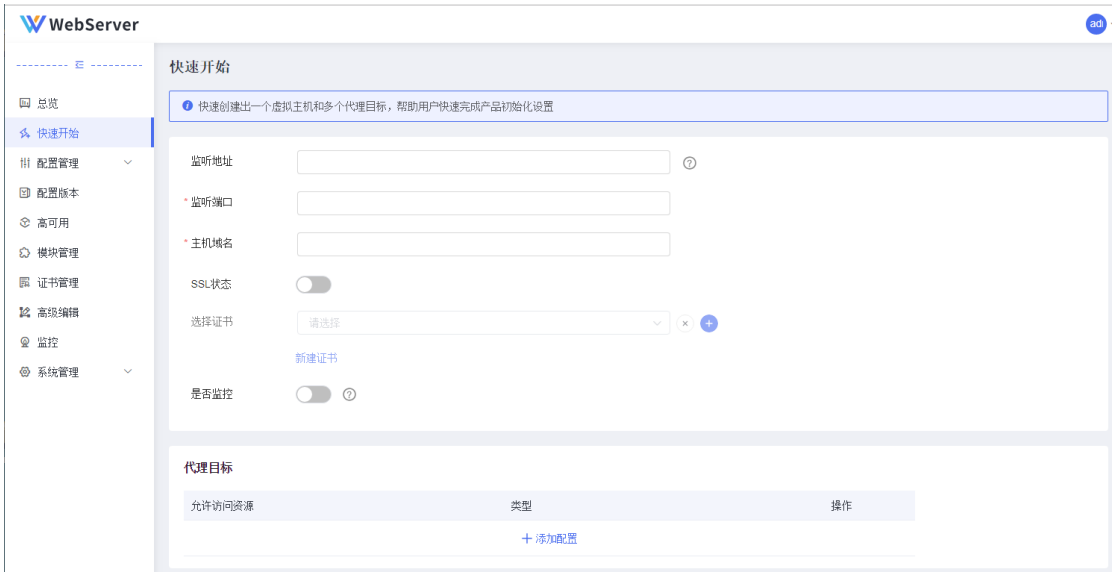


图 4-2 快速开始

用户在快速开始创建好虚拟主机和代理目标之后，可以在“**配置管理**”->“**虚拟主机**”查看新建好的资源。

4.2.1 运行BES WebServer实例

用户可以在管理中心操作BES WebServer实例。在管理中心左侧导航区点击“**总览**”，如下图所示，在右侧进行实例的启动、停止和重加载操作。

图 4-3 基本参数

4.3 配置管理

配置管理模块是将BES WebServer产品的配置文件bws.conf属性界面化，按照功能用途分区域展示，简化用户使用难度和学习成本，更快上手。主要划分了基本参数、HTTP参数、虚拟主机、上游和区域模块。

4.3.1 基本参数

基本参数定义了BES WebServer的一些基本属性，包括工作进程，错误日志位置，错误日志级别，工作模式和最大连接数。

在管理中心左侧导航区点击“**配置管理**”–>“**基本参数**”，进入“**基本参数**”编辑页面。可配置项如下：

图 4-4 基本参数

表 4-1 基本参数

配置项名称	属性名称	说明
工作进程	worker_processes	工作进程的数量，建议等于CPU数量或者2倍于CPU。合法值：-1或者1-2147483647之间的整数，-1表示与CPU数量相同，默认值：-1。
错误日志位置	error_log	存放错误日志的目录。
错误日志级别	error_log	错误日志的级别。可选值有debug、info、notice、warn、error、crit、alert、emerg。日志中将会输出指定级别的或者更高级别的日志。

配置项名称	属性名称	说明
工作模式	use	事件驱动模型，默认值：epoll。select、poll：属于标准事件模型，如果当前系统不存在更有效的方法，默认会选择select或poll； kqueue：适用于FreeBSD 4.1+, OpenBSD 2.9+, NetBSD 2.0 和 MacOS X；epoll：适用于Linux内核2.6版本及以后的系统； /dev/poll：适用于Solaris 7 11/99+，HP/UX 11.22+ (eventport)，IRIX 6.5.15+ 和 Tru64 UNIX 5.1A+;
最大连接数	worker_connections	单个BWS后台进程允许的最大并发连接数。 合法值：1-65535，默认值：512。

4.3.2 HTTP参数

HTTP参数是处理HTTP请求的核心模块，包括默认MIME类型，连接超时时间，访问日志配置。压缩模块使得文件在传输过程中可以使用gzip压缩，从而提升传输效率。客户端模块可以配置客户端请求头缓存大小以及最大上传文件限制等参数。

在管理中心左侧导航区点击“**配置管理**”->“**HTTP参数**”，进入“**HTTP参数**”编辑页面。可配置项如下：

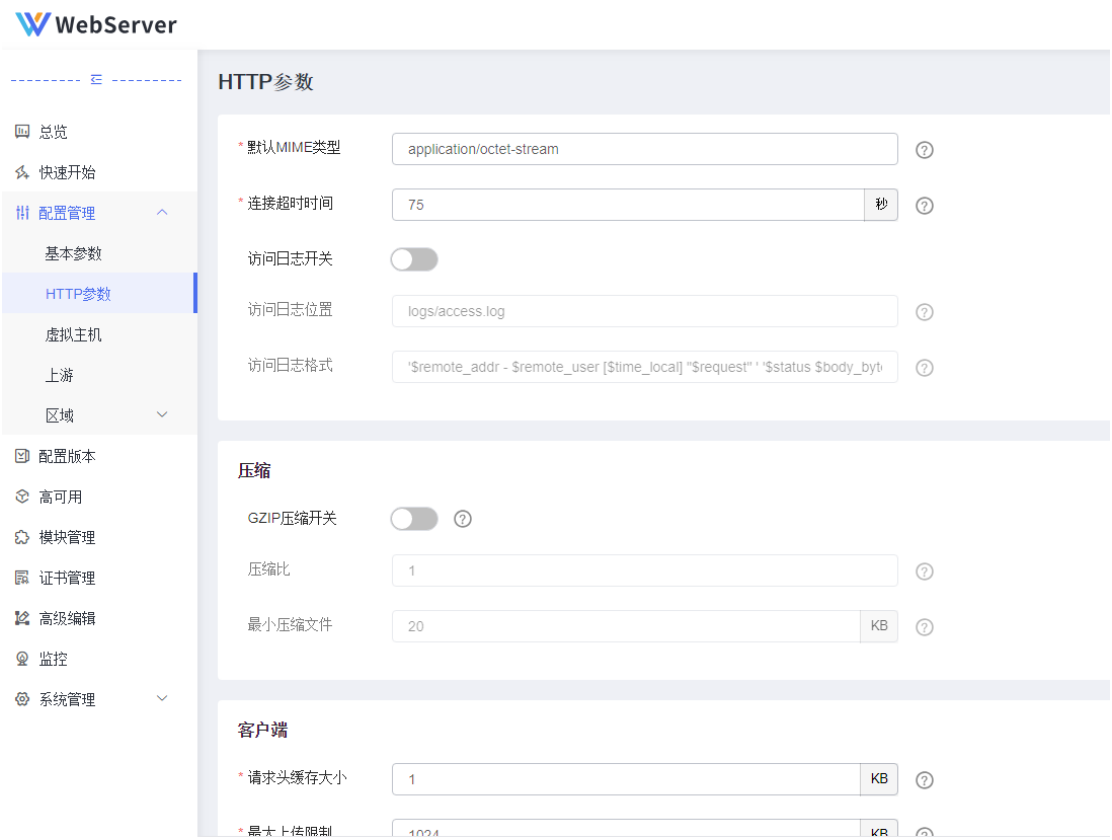


图 4-5 HTTP参数

表 4-2 HTTP参数

配置项名称	属性名称	说明
默认MIME类型	default_type	设定MIME类型，由mime.type文件决定。默认值：application/octet-stream。
长连接超时时间	keepalive_timeout	指定每个TCP连接最多可以保持多长的时间，默认值：75秒，部分浏览器最多只能保持60秒，若设置为0，则禁止保持连接。
访问日志开关	access_log	若开启访问日志功能，则系统会按照下述设置的日志格式记录每次的请求信息。
访问日志位置	access_log	访问日志的存放位置，默认值：logs/access.log。
访问日志格式	log_format	设置日志的打印格式，默认值：`$remote_addr - $remote_user [$time_local] "$request" $status $body_bytes_sent "$http_referer" "$http_user_agent" "$http_x_forwarded_for"`。
GZIP压缩开关	gzip	是否开启gzip压缩。
压缩比	gzip_comp_level	压缩比率，默认值：1，可选值1-9，级别越高压缩率越大，压缩时间越长。
最小压缩文件	gzip_min_length	允许压缩的页面最小字节数，默认值：20，单位为KB，页面字节数从header头中的Content-Length中进行获取。建议设置成大于1k的字节数，小于1k可能会越压越大。
请求头缓存大小	client_header_buffer_size	客户端请求行+请求头的缓冲区大小，默认值：1，单位为KB，如果请求大小小于该值，则请求通过，如果超过则以最大请求头缓存大小large_client_header_buffer_size为准。
最大上传限制	client_max_body_size	客户端请求体允许通过的最大大小，默认值：1024，单位为KB，如果请求超过该大小，客户端会提示413 Request Entity Too Large错误。
请求体缓冲区	client_body_buffer_size	处理客户端请求体的缓冲区大小，默认值：8，单位：KB。如果请求体数据小于该值，数据将在内存中储存，如果大于则会储存在临时文件中。

4.3.3 虚拟主机

虚拟主机根据类型分为正向代理和反向代理，正向代理的上游是客户端，反向代理的上游是服务端，根据URL来匹配，然后分发请求给对应的服务器去进行处理。

在管理中心左侧导航区点击“**配置管理**”->“**虚拟主机**”，操作区域显示“**虚拟主机**”页面，默认显示监听地址和端口为8090的虚拟主机。

图 4-6 虚拟主机页面

4.3.3.1 新建虚拟主机

管理员在管理中心新建虚拟主机：

1. 在管理中心左侧导航区点击“**配置管理**”->“**虚拟主机**”，进入“**虚拟主机**”页面
2. 在“**虚拟主机**”页面，点击“**新建**”按钮，进入“**新建虚拟主机**”页面。

4.3.3.1.1 基本信息

1. 管理员可以在基本信息页面对虚拟主机的基本信息进行相应配置，可配置项如下：

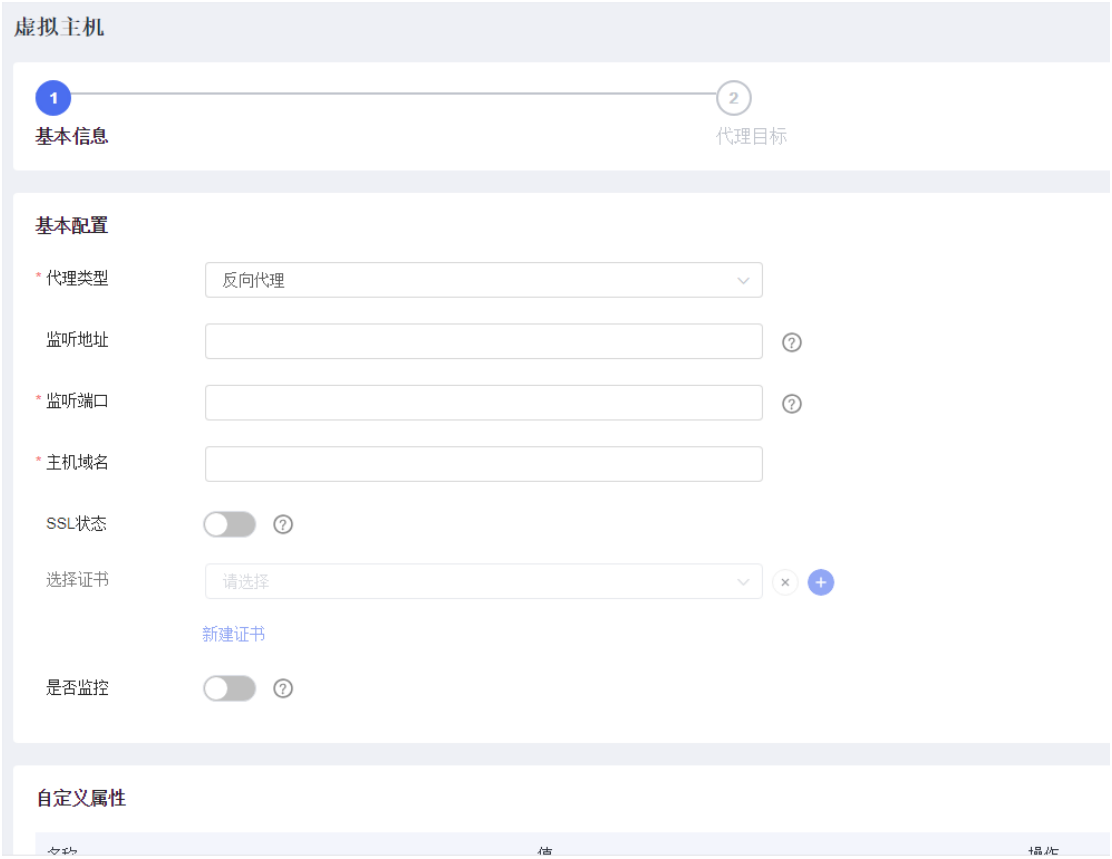


图 4-7 基本信息页面

表 4-3 基本配置

配置项名称	属性名称	说明
代理类型		可选值：正向代理、反向代理。
DNS解析地址	resolver	配置DNS解析IP地址。
监听地址	listen	虚拟主机的监听地址，若为空表示当前机器IP地址，默认值：空。
监听端口	listen	监听端口必须是1-65535的整数。
主机域名	server_name	主机的域名。
SSL状态	listen	是否开启SSL，可选值：启用、禁用，默认是禁用。
是否监控	monitor	控制是否采集该虚拟主机的监控数据，默认关闭，开启后会创建一个默认location进行监控采集。

当BES WebServer的正向代理模式需要使用HTTPS时，需要配置HTTP Connect方法。

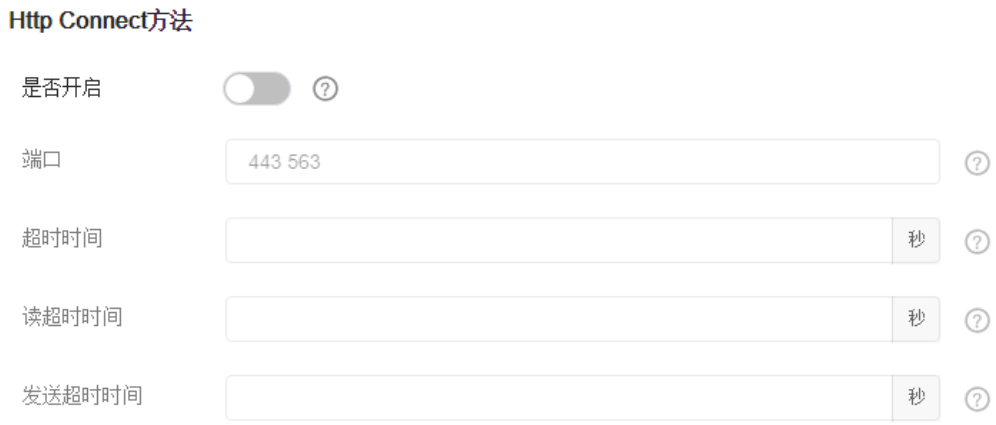


图 4-8 Http Connect

表 4-4 Http Connect方法

配置项名称	属性名称	说明
是否开启	proxy_connect	是否开启对HTTP方法“CONNECT”的支持。
端口	proxy_connect_allow	指定允许开启CONNECT方法的端口，默认只有443和563端口被允许。
超时时间	proxy_connect_connect_timeo	与后端服务器建立连接的超时时间，单位：秒。
读超时时间	proxy_connect_read_timeout	读后端服务器数据的等待时间，仅在两次读数据之间生效，而不是整个应答数据时间。如果后端服务器在超时时间内未发送任何数据，则连接将被关闭，单位：秒。
发送超时时间	proxy_connect_send_timeout	发送数据到后端服务器的等待时间，仅在两次发送数据之间生效，不是整个请求时间。如果后端服务器在等待期间未收取任何数据，则连接将被关闭，单位：秒。

基本信息配置完成后点击下一步，进入“**代理目标**”页面。

4.3.3.1.2 代理目标

管理员可以在代理目标页面对虚拟主机的代理目标进行配置，点击代理目标右侧的”**添加**“按钮，可以添加一条代理目标。

编辑代理目标又分为：基本信息、静态缓存、组合请求、流量控制、系统保护、自定义属性。

基本信息

代理目标的一些基本信息，可配置项如下：

图 4-9 代理目标基本信息页面

表 4-5 基本信息

配置项名称	属性名称	说明
允许访问资源	location	对需要反向代理的URL进行匹配，匹配规则如下：1）=：精确匹配；2）~：正则表达式匹配，匹配时区分字符大小写 3）~*：正则表达式匹配，匹配时忽略字符大小写 4）^~：表示以某个字符串开头。
代理类型	proxy_pass	后端代理目标的类型，分为静态html、动态http和上游，其中：1）静态html：访问的目标是静态页面。2）动态http：转发请求的目标是具体的http地址。3）上游：转发请求的目标是上游地址，需要配合上游使用。
模式	location	指定文件路径的两种方式，区别在于 如何解释location后面的uri。root：指定目录的上级目录，最终的文件目录包含location指定名称的同名目录。
文件路径	location	访问的静态文件路径。
上游名称	proxy_pass	需要提前建好上游配置。
连接协议	proxy_pass	选择连接的协议，可选值：http、https。
代理地址	location	填写处理请求的后端真实地址。
代理服务器	proxy_pass	代理服务器的地址。
缓存区数量	proxy_buffers	从被代理的后端服务器取得的相应内容，会放置在缓冲区，该属性设置缓冲区的个数，和缓存区大小配合使用。

配置项名称	属性名称	说明
缓存区大小	proxy_buffers	每个缓冲区的大小，和缓存数量配合使用，单位：kb。
连接超时时间	proxy_connect_timeout	连接后端服务器的超时时间。

模式举例：

```
location /img/{
  root /var/ww/image
}
#bws会自动去/var/www/image/img/目录下找文件
#alias：指定一个准确目录，例如
location /img/{
  alias /var/ww/image
}
#bws会自动去/var/ww/image/目录下找文件
```

上游举例：

```
location /as2 {
  proxy_pass http://up1;
}
upstream up1{
  server 192.168.5.23:888 weight=1 fail_timeout=10s max_fails=1 ;
}
#在上游配置中定义的内容
```

代理地址举例：

```
location /as1 {
  proxy_pass http://192.168.23.12:8888;
}
```

静态缓存

对静态资源进行缓存，减轻后端服务器的压力，进一步提升访问效率，默认关闭。

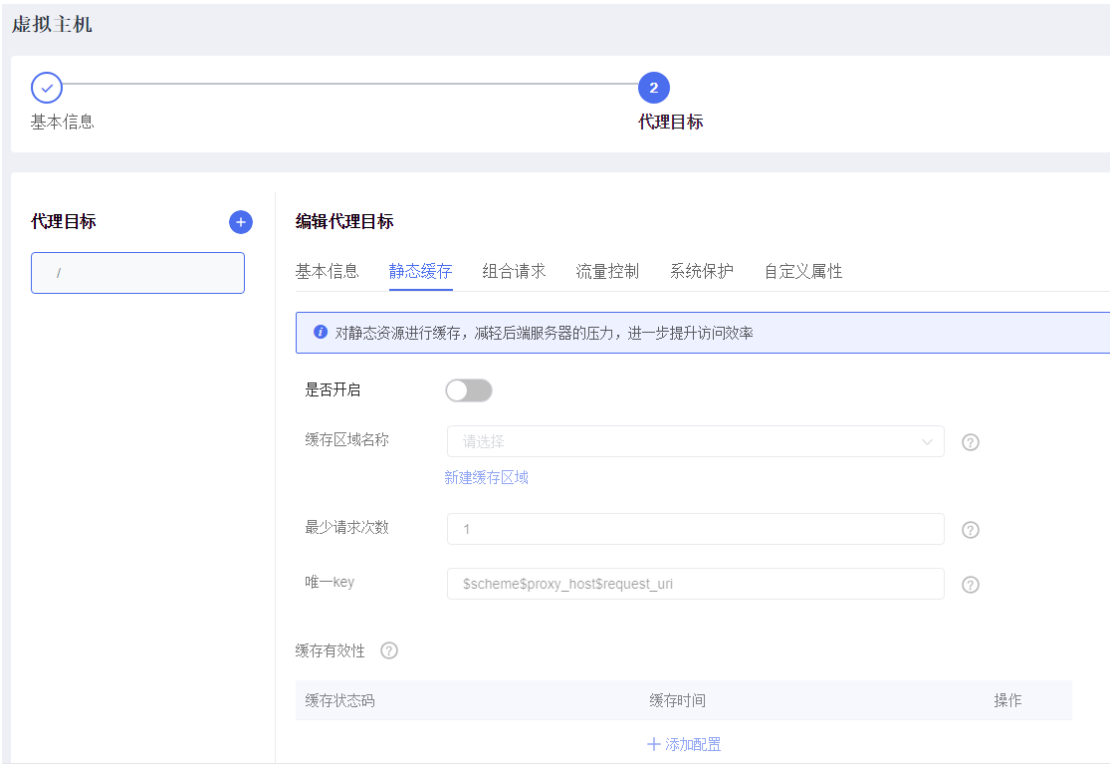


图 4-10 代理目标静态缓存页面

表 4-6 静态缓存

配置项名称	属性名称	说明
是否开启	proxy_cache zone	是否开启静态缓存。
缓存区域名称	proxy_cache **zone**	缓存区域的使用请参考配置管理/区域/缓存区域功能。
最少请求次数	proxy_cache_min_uses	在缓存响应前，必须使用相同密钥请求的最小次数。
唯一key	proxy_cache_key	计算缓存时的密钥唯一key，默认值：`$scheme$proxy_host$request_uri`。
缓存有效性	proxy_cache_valid code time	响应状态码的缓存时间，可以为每个状态码缓存指定时间，也可以使用any进行全部状态码的缓存。
缓存状态码	proxy_cache_valid **code** time	缓存的状态码。
缓存时间	proxy_cache_valid code **time**	缓存的时间。

组合请求

组合多个CSS，JavaScript文件的请求变成一个请求。

图 4-11 代理目标组合请求页面

表 4-7 组合请求

配置项名称	属性名称	说明
是否开启	concat	组合请求仅支持反向代理，且代理类型必须是静态HTML类型。
MIME类型	concat_types	允许被组合成一个请求的资源类型。
文件类型	concat_unique	是否只组合MIME Type相同类型的文件。
最大文件数量	concat_max_files	最大能接受的文件数量。
文件分隔符	concat_delimiter	在文件之间添加的分隔符，例如http://demo.com/??1.js,2.js。

流量控制

使用漏桶算法，限制用户在给定的时间内的HTTP请求数量，控制方式包括同一IP的并发连接数、限制同一IP某段时间的访问量，限制同一IP流量。可配置项如下：

图 4-12 代理目标流量控制页面

表 4-8 并发数

配置项名称	属性名称	说明
流控区域	limit_conn **zone** number	流控区域的使用请参考配置管理/区域/流控区域功能。
最大并发数	limit_conn zone **number**	允许相同标识的客户端同时在处理请求数，超过阈值的请求将返回预定错误（返回码是拒绝响应码设置的值）。
拒绝响应码	limit_conn_status	响应被拒绝的请求返回的状态码。
日志错误级别	limit_conn_log_level	被拒绝的请求会以日志的形式输出到日志文件中，该属性定义输出的错误日志级别。

表 4-9 访问速率

配置项名称	属性名称	说明
流控区域	limit_req zone	流控区域的使用请参考配置管理/区域/流控区域功能。
缓冲区队列大小	limit_req burst	超过访问频次限制的请求，可以先放在缓冲区内等待，超过缓存区设置大小的请求直接错误返回（返回码是拒绝响应码设置的值）。
是否延迟	limit_req	可选值：是、否，默认：否，选择否时，会在瞬间提供处理（速率限制+缓冲区队列大小）个请求的能力，超过的请求直接错误返回（返回码是拒绝响应码设置的值），若选择是，则超过速率限制的请求会依次排队等待，直至延迟超时。

配置项名称	属性名称	说明
延迟处理阈值	limit_req delay	超过（速率限制+缓冲区队列大小）的请求，排队延迟的最长时间，超过后错误返回（返回码是拒绝响应码设置的值。
拒绝响应码	limit_req_status	响应被拒绝的请求返回的状态码。
日志错误级别	limit_req_log_level	被拒绝的请求会以日志的形式输出到日志文件中，该属性定义输出的错误日志级别。

表 4-10 传输速率

配置项名称	属性名称	说明
传输速度	limit_rate	指定向客户端传输传输数据的速度，单位是kb/s，该限制只针对一个连接，如果同时有N个连接，则速度是N倍。
开始限速阈值	limit_rate_after	在未达到阈值前，以最大的速度下载，达到后再进行限速。

系统保护

监控内存、CPU和请求的响应时间，当某些监控指标达到设定的阈值后，跳转到指定的url页面，该模块仅在系统支持sysinfo函数时，才支持基于load与内存信息的包含，以及系统支持loadavg函数时支持基于load的保护，模块需要从/proc文件系统中读取内存信息。可配置项如下：

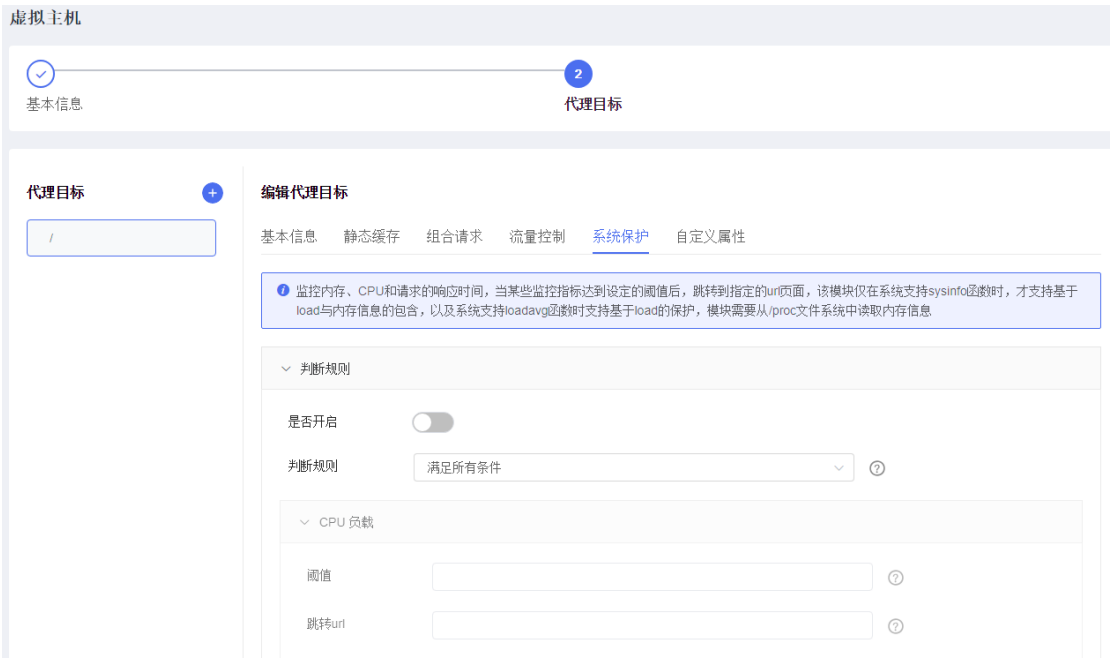


图 4-13 代理目标系统保护页面

表 4-11 判断规则

配置项名称	属性名称	说明
是否开启	sysguard	是否开启系统保护。
判断规则	sysguard_mode	当设置多个监控指标条件时，触发系统保护的规则。

表 4-12 CPU负载

配置项名称	属性名称	说明
阈值	sysguard_load load	当系统1分钟内的load达到阈值时，对请求进行限制，以保护系统。
跳转url	sysguard_load action	达到阈值后，请求会被转到该url上，如果不配置，则直接返回503错误。

表 4-13 CPU使用率

配置项名称	属性名称	说明
阈值	sysguard_cpu usage	当系统在统计周期内的cpu达到阈值时，对请求进行限制，以保护系统。
统计周期	sysguard_cpu period	统计的周期，单位秒。
跳转url	sysguard_cpu action	达到阈值后，请求会被转到该url上，如果不配置，则直接返回503错误。

表 4-14 内存交换区

配置项名称	属性名称	说明
阈值	sysguard_mem swapratio	当前交换空间已使用率超过该阈值时，对请求进行限制，以保护系统。
跳转url	sysguard_mem action	达到阈值后，请求会被转到该url上，如果不配置，则直接返回503错误。

表 4-15 空闲内存

配置项名称	属性名称	说明
阈值	sysguard_mem free	剩余可用的内存小于阈值时，对请求进行限制，以保护系统，单位：MB。
跳转url	sysguard_mem action	达到阈值后，请求会被转到该url上，如果不配置，则直接返回503错误。

表 4-16 请求响应时间

配置项名称	属性名称	说明
阈值	sysguard_rt rt	在统计周期内，请求平均响应时间大于阈值时，对请求进行限制，以保护系统，单位：秒。
统计周期	sysguard_rt period	统计周期，单位：秒。
跳转url	sysguard_rt action	达到阈值后，请求会被转到该url上，如果不配置，则直接返回503错误。

自定义属性

管理员可以自定义一些属性。

图 4-14 代理目标系统保护页面

4.3.3.2 编辑虚拟主机

管理员在管理中心编辑虚拟主机：

1. 在管理中心左侧导航区点击“**配置管理**”->“**虚拟主机**”，进入“**虚拟主机**”页面。
2. 在“**虚拟主机**”页面，点击“**虚拟主机**”链接，进入“**编辑虚拟主机**”，可编辑属性如下：

图 4-15 编辑虚拟主机

3. 修改完毕后点击“**保存**“按钮保存修改的属性。

4.3.3.3 删除虚拟主机

管理员在管理中心删除虚拟主机：

1. 在管理中心左侧导航区点击“**配置管理**”->“**虚拟主机**”，进“**虚拟主机**”页面。
2. 在“**虚拟主机**”页面，勾选要删除的虚拟主机，点击“**删除**”按钮删除即可。

4.3.4 上游

通过上游可以指定后端服务器的列表，实现对服务请求处理的负载均衡，提供多种负载均衡策略，用户可以根据实际业务场景选择具体的负载算法进行使用。

在管理中心左侧导航区点击“**配置管理**”->“**上游**”，操作区域显示“**上游**”页面。

图 4-16 上游页面

4.3.4.1 新建上游

管理员在管理中心新建上游：

1. 在管理中心左侧导航区点击“**配置管理**”–>“**上游**”，进入“**上游**”页面。
2. 在“**上游**”页面，点击“**新建**”按钮，进入“**新建上游**”页面，可配置项如下：

基本信息

上游的基本信息，可配置项如下：

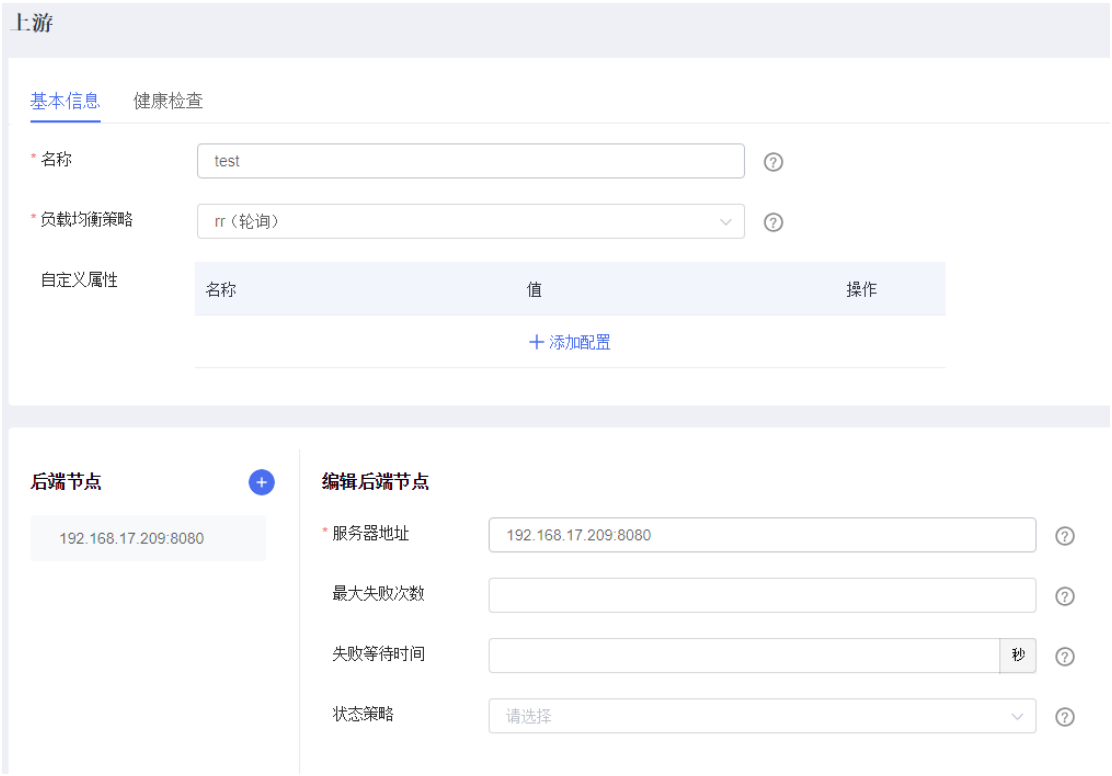


图 4-17 上游基本信息页面

表 4-17 基本信息

配置项名称	属性名称	说明
名称	upstream	上游名称
负载均衡策略		rr（轮询）：将请求按顺序轮流分配到后端服务器。wrr（加权轮询）：根据后端服务器的权重分配请求。ip_hash（源地址哈希）：根据客户端IP地址计算处理该请求的后端服务器。fair（平衡）：根据后端服务器的响应时间分配请求，响应时间短的优先分配。url_hash(url哈希)：根据客户端URL地址计算处理该请求的后端服务器序号。sticky（会话保持）:根据客户端的Cookie计算处理该请求的后端服务器。least_conn（最小连接）：根据后端服务器的活跃连接数与权重比值分配请求。

表 4-18 后端节点

配置项名称	属性名称	说明
服务器地址	server	后端服务器对应的IP和端口信息。
权重	weight	权重越大，分配的请求越多。
最大失败次数	max_fails	允许请求失败的次数，超过后返回错误。
失败等待时间	fail_timeout	达到最大失败次数后，暂停的时间，单位毫秒。
状态策略		无：正常参与负载。停用：不参与负载。备用：其他服务器不可用时，参与负载。

健康检查

上游的健康检查，配置项如下：

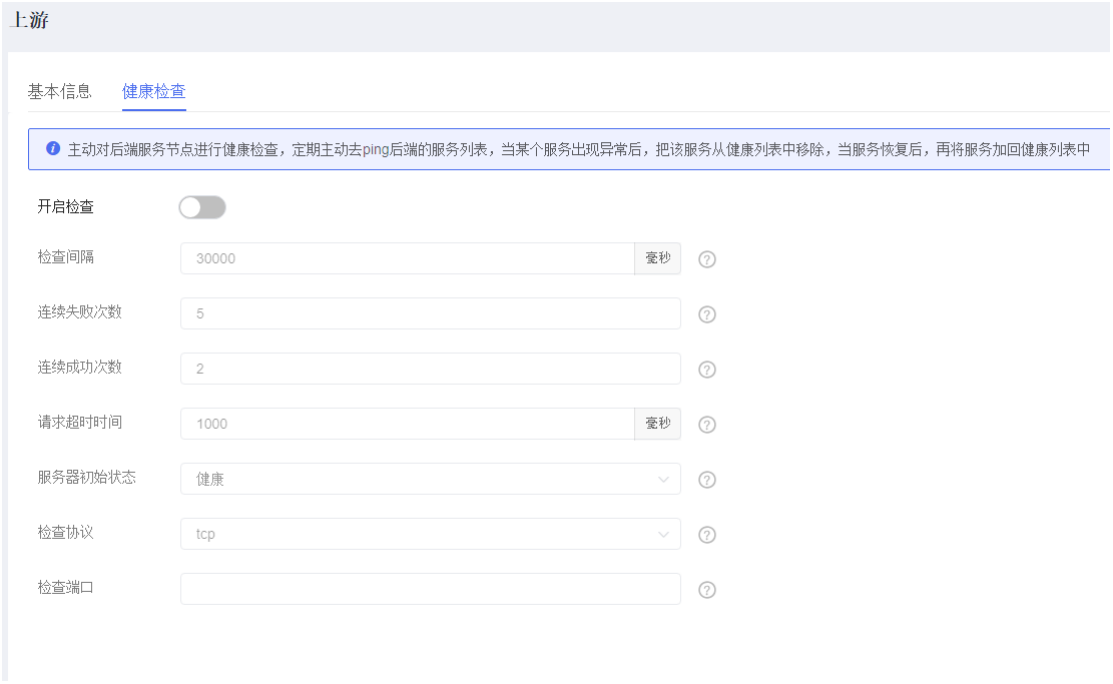


图 4-18 上游监控检查页面

表 4-19 健康检查

配置项名称	属性名称	说明
开启检查	check	是否开启检查。
检查间隔	check interval	向后端发送的健康检查包的间隔。默认值：30000，单位：毫秒。
连续失败次数	check fall	如果连续失败次数达到所指定的值，服务器就被认为是不健康的。默认值：5。
连续成功次数	check rise	如果连续成功次数达到所指定的值，服务器就被认为是健康的。默认值：2。
请求超时时间	check timeout	后端健康请求的超时时间。默认值：1000，单位：毫秒。
服务器初始状态	check default_down	设定初始时服务器的状态。默认值：健康。
检查协议	check type	发送健康检查包的类型。默认值：tcp。tcp：简单的TCP连接，如果连接成功，说明后端正常；http：发送http请求，通过后端的回复判断是否存活。
检查端口	check port	指定后端服务器的检查端口。默认为0，表示跟后端服务器提供真实服务的端口一样。
HTTP报文	check_http_send	配置http健康检查包发送的请求内容。默认值：`GET / HTTP/1.0\r\n\r\n`。

4.3.4.2 编辑上游

管理员在管理中心编辑上游:

1. 在管理中心左侧导航区点击“**配置管理**”->“**上游**”，进入“**上游**”页面。

2. 在“**上游**”页面，点击“**上游名称**”链接，进入“**编辑上游**”页面，可编辑属性如下:

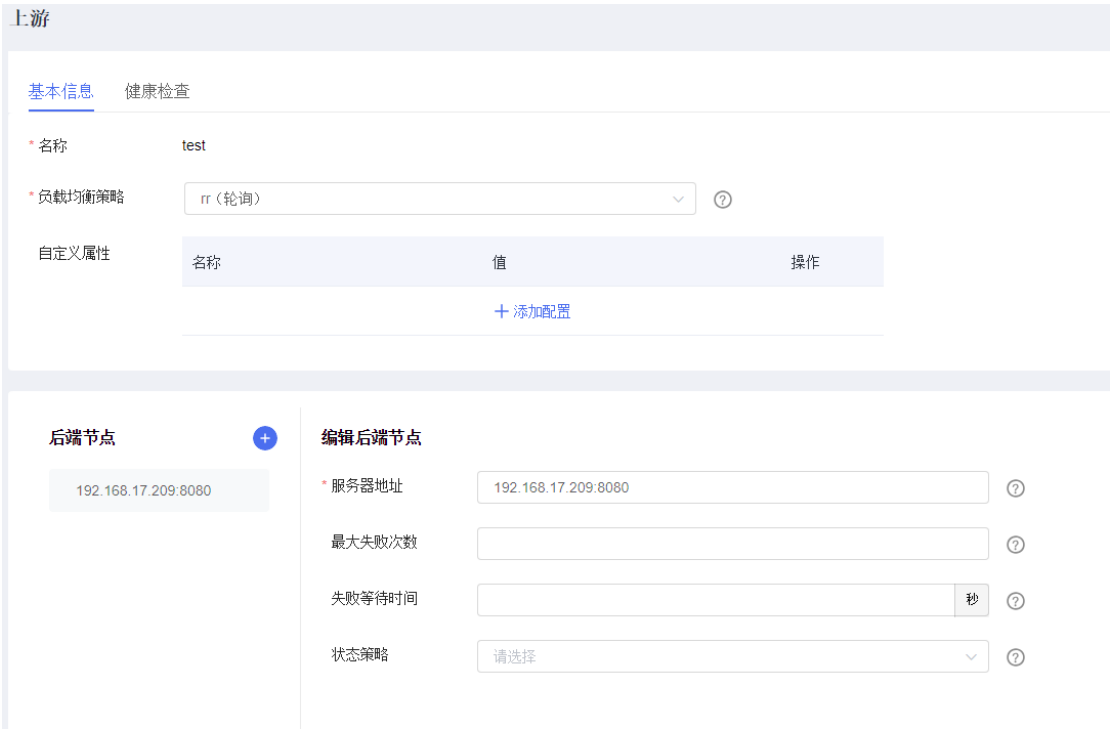


图 4-19 编辑上游

3. 修改上游的属性，点击“**保存**“按钮保存修改的属性。

4.3.4.3 删除上游

管理员在管理中心删除上游:

1. 在管理中心左侧导航区点击“**配置管理**”->“**上游**”，进入“**上游列表**”页面。
2. 在“**上游列表**”页面，勾选要删除的上游，点击“**删除**”按钮删除即可。

4.3.5 区域

4.3.6 流控区域

用来限制用户在给定时间内HTTP请求的数量，保护上游应用服务器不被同时过多用户请求压垮。

图 4-20 流控区域

4.3.6.0.1 新建流控区域

管理员管理中心新建流控区域:

1. 在管理中心左侧导航区点击“**配置管理**”->“**区域**”->“**流控区域**”，进入“**流控区域**”页面。
2. 在“**流控区域**”页面，点击“**新建**”按钮，进入“**新建流控区域**”页面。可配置项如下:

图 4-21 新建流控区域页面

表 4-20 流控区域

配置项名称	属性名称	说明
共享域名称	limit_req_zone zone	所创建的共享域名称。
限制指标	limit_req_zone limit_conn_zone	选择限制指标，可选值：访问速率，并发数。 默认值：访问速率。
KEY	limit_req_zone	指定一个key，标识是否是同一个客户端。 默认值：$binary_remote_addr。

配置项名称	属性名称	说明
共享域大小	limit_req_zone zone	生成一个指定大小的共享区域，用来储存客户端的连接信息，合法值：32-2147483647。单位：KB。
最大请求速率	limit_req_zone rate	允许相同标识的客户端的访问频次，合法值：1-2147483647。单位：次/每秒，次/每分钟。

4.3.6.0.2 编辑流控区域

管理员在管理中心编辑流控区域：

1. 在管理中心左侧导航区点击“**配置管理**”–>“**区域**–“**流控区域**”，进入“**流控区域**”页面。
2. 在“**流控区域**”页面，点击“**区域名称**”链接，进入“**编辑流控区域**”页面，可编辑属性如下：

图 4-22 编辑流控区域

3. 修改流控区域的属性，点击“**保存**”按钮保存修改的属性。

4.3.6.0.3 删除流控区域

管理员在管理中心删除流控区域：

1. 在管理中心左侧导航区点击“**配置管理**”–>“**区域**”–>“**流控区域**”，进入“**流控区域**”页面。
2. 在“**流控区域**”页面，勾选要删除的流控区域，点击“**删除**”按钮删除即可。

4.3.6.1 缓存区域

作为性能优化的一个重要手段，可以极大减轻后端服务器的负载。通常对于静态资源，即较少经常更新的资源，如图片，css或js等进行缓存，从而在每次刷新浏览器的时候，不用重新请

求，而是从缓存里面读取，这样就可以减轻服务器的压力。

图 4-23 缓存区域

4.3.6.1.1 新建缓存区域

管理员在管理中心新建缓存区域：

1. 在管理中心左侧导航区点击“**配置管理**”–>“**区域**”–>“**缓存区域**”，进入“**缓存区域**“页面
2. 在“**缓存区域**”页面，点击“**新建**”按钮，进入“**新建缓存区域**”页面。可配置项如下：

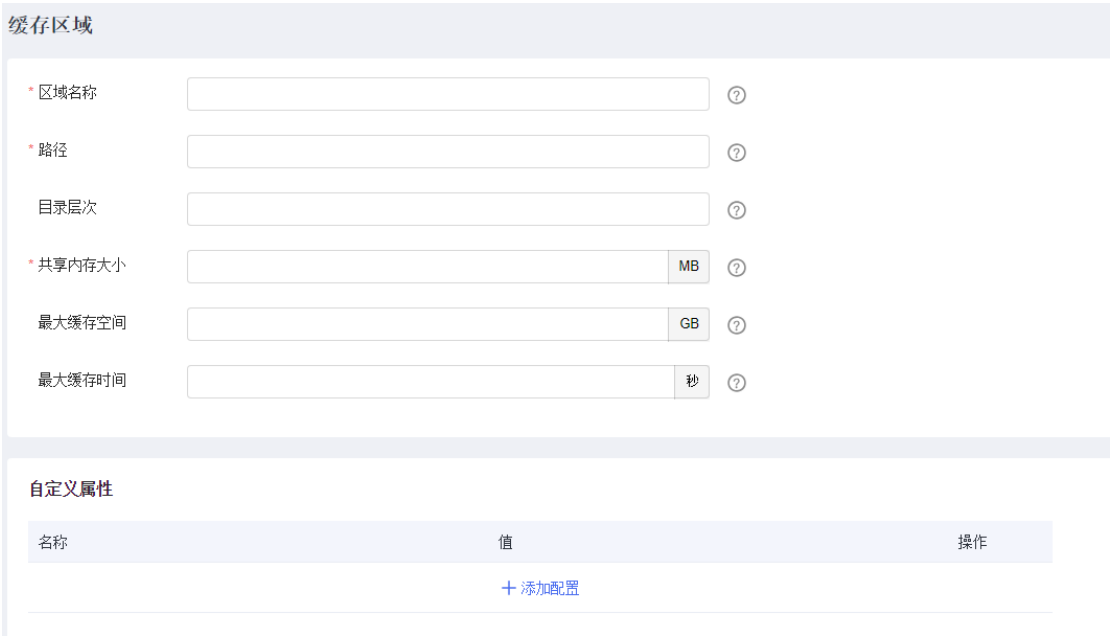


图 4-24 新建缓存区域页面

表 4-21 缓存区域

配置项名称	属性名称	说明
区域名称	proxy_cache_path keys_zone	设置缓存区域的名称。
路径	proxy_cache_path	缓存文件的本地存放路径。
目录层次	proxy_cache_path levels	设置缓存文件的目录层次，1:2代表有2级目录。
共享内存大小	proxy_cache_path keys_zone	共享内存大小，合法值：1-2147483647。单位：MB。
最大缓存空间	proxy_cache_path max_size	最大缓存空间，合法值：1-2147483647。单位：GB。
最大缓存时间	proxy_cache_path inactive	最大缓存时间，合法值：1-2147483647。单位：秒。

4.3.6.1.2 编辑缓存区域

管理员在管理中心编辑缓存区域：

1. 在管理中心左侧导航区
2. 点击“**配置管理**”–>“**区域**”–“**缓存区域**”，进入“**缓存区域**”页面。
3. 在“**缓存区域**”页面，点击“**区域名称**”链接，进入“**编辑缓存区域**”页面，可编辑属性如下：

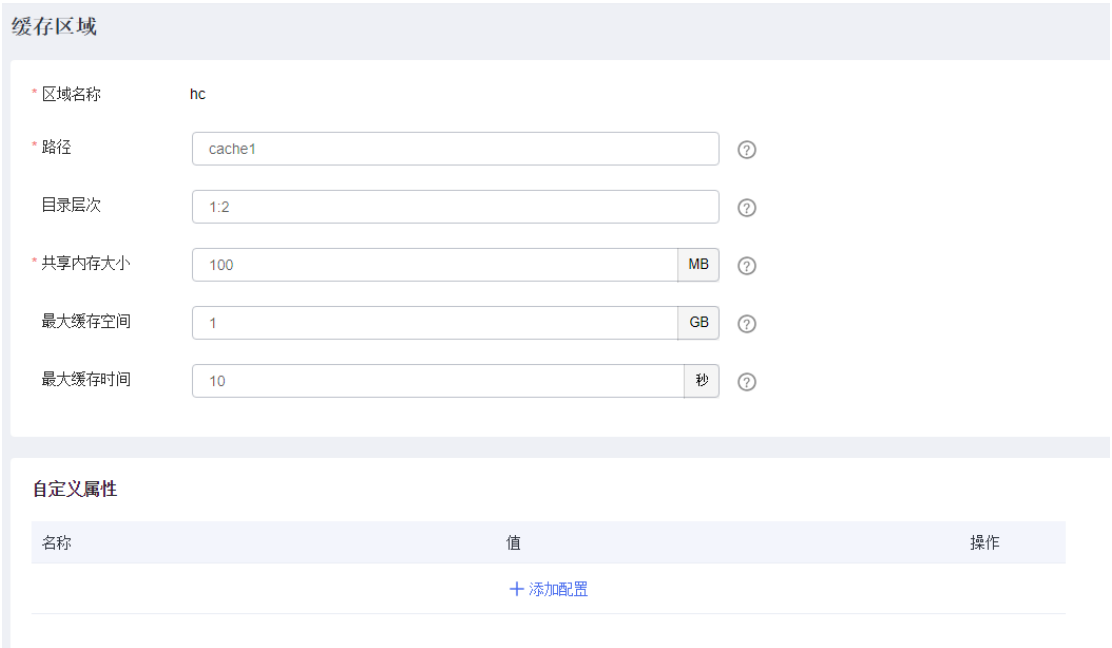


图 4-25 编辑缓存区域

4. 修改缓存区域的属性，点击“**保存**”按钮保存修改的属性。

4.3.6.1.3 删除缓存区域

管理员在管理中心删除缓存区域:

1. 在管理中心左侧导航区点击“**配置管理**”–>“**区域**”–>“**缓存区域**”，进入“**缓存区域**”页面。
2. 在“**缓存区域**”页面，勾选要删除的缓存区域，点击“**删除**”按钮删除即可。

4.4 配置版本

对配置属性进行备份，支持恢复到指定版本的配置，完成对配置的回退，避免误操作导致的配置损坏，通过建立基线的方式建立配置的标准版本。

生成版本：备份当前的配置，作为一个版本记录到版本列表中。

查看：查看该版本备份的配置信息。

恢复：用选中的版本覆盖线上环境的配置，该操作不可逆，请谨慎选择。

删除：删除该版本备份的配置信息。

在管理中心左侧导航区点击“**配置版本**”，进入“**配置版本**”页面，显示所有已生成到管理中心的版本。

图 4-26 配置版本

4.5 高可用

将多个BWS实例组成高可用组，在主节点故障后，由备用节点接管，对外提供服务，保证系统服务的可用性。

在管理中心左侧导航区点击“**高可用**”，进入“**高可用**”页面，可配置项如下:

图 4-27 高可用

表 4-22 高可用

配置项名称	配置项说明	默认值
主备类型	选择节点的初始状态，存在主节点和备用节点两种状态，主节点是工作节点，在主节点故障时，从备用节点选举出一个转为主节点，主节点优先于备用节点获得虚拟IP的使用权。	主节点
网卡	机器所配置的网卡名称。	
虚拟IP	在主备机器上来回切换的IP，如果主节点故障，虚拟IP会切换到备用节点上，待主节点恢复正常后，再切换回去，可同时绑定多个虚拟IP，以逗号分隔。	

高可用使用步骤:

1、用户需要自行准备高可用介质,并将其解压到产品的 BWS_HOME/modules/keepalived 目录下。

2、在控制台配置好高可用的网卡和IP信息。

3、使用root用户在BWS_HOME/bin目录下执行:

```
[root@Linux17209 bin]$ ./keepalived  start
```

4、启动完后使用root用户执行如下命令，如果虚拟机IP已经存在，则证明高可用服务已经启动成功。

```
[root@Linux17209 bin]# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group def
↪  ault qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
```

```
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP 」
↪  group default qlen 1000
    link/ether 00:0c:29:99:ce:19 brd ff:ff:ff:ff:ff:ff
    inet 192.168.17.209/22 brd 192.168.19.255 scope global noprefixroute ens33
       valid_lft forever preferred_lft forever
    inet 16.25.33.98/32 scope global ens33                          --->虚拟IP
       valid_lft forever preferred_lft forever
    inet6 2001::209/64 scope global noprefixroute
       valid_lft forever preferred_lft forever
    inet6 fe80::32e8:af23:97e1:a784/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

4.6 模块管理

在运行时有选择的加载模块，实现模块的动态增减，节省系统的资源占用。

在管理中心左侧导航区点击“**模块管理**”，操作区域显示“**模块管理**”页面，显示所有已添加到管理中心的模块。

图 4-28 模块管理

4.6.1 新建模块

管理员在管理中心新建模块：

1.在管理中心左侧导航区点击“**模块管理**”，进入“**模块管理**”页面。

2.在“**模块管理**”页面，点击新建按钮，进入“**新建模块**”页面，可配置项如下：

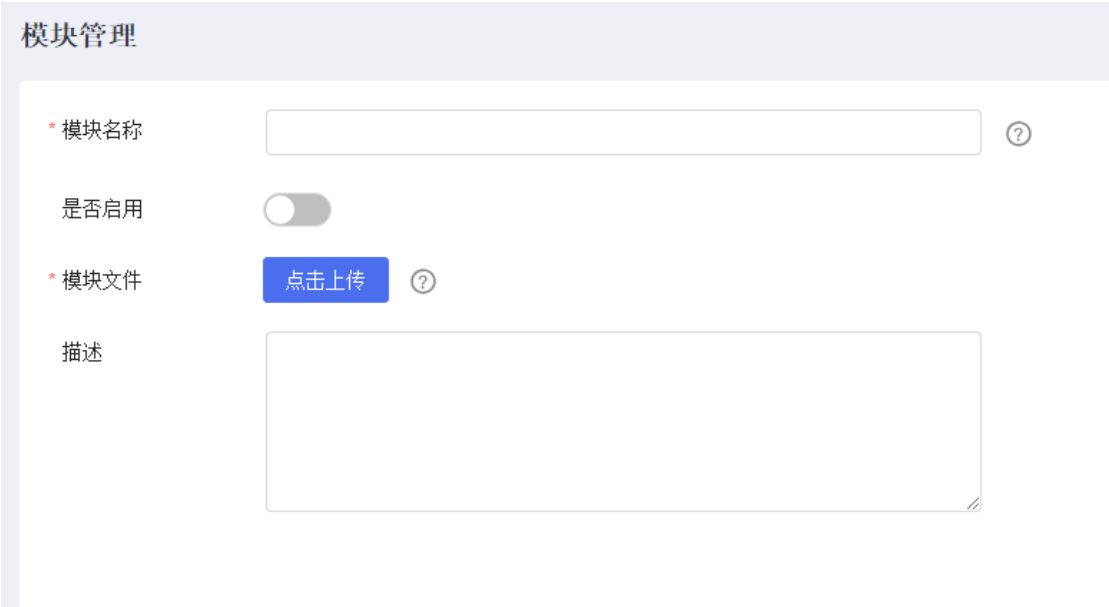


图 4-29 新建模块

表 4-23 新建模块

配置项名称	配置项说明	默认值
模块名称	模块的名称。	无
是否启用	该模块是否启用。	禁用
模块文件	模块文件，请上传so类型文件。	无
描述	关于模块的描述。	无

4.6.2 编辑模块

管理员在管理中心编辑模块：

1. 在管理中心左侧导航区点击“**模块管理**”，进入“**模块管理**”页面。
2. 在“**模块管理**”页面，点击“**模块名称**”按钮，进入“**编辑模块**”页面，可编辑属性如下：

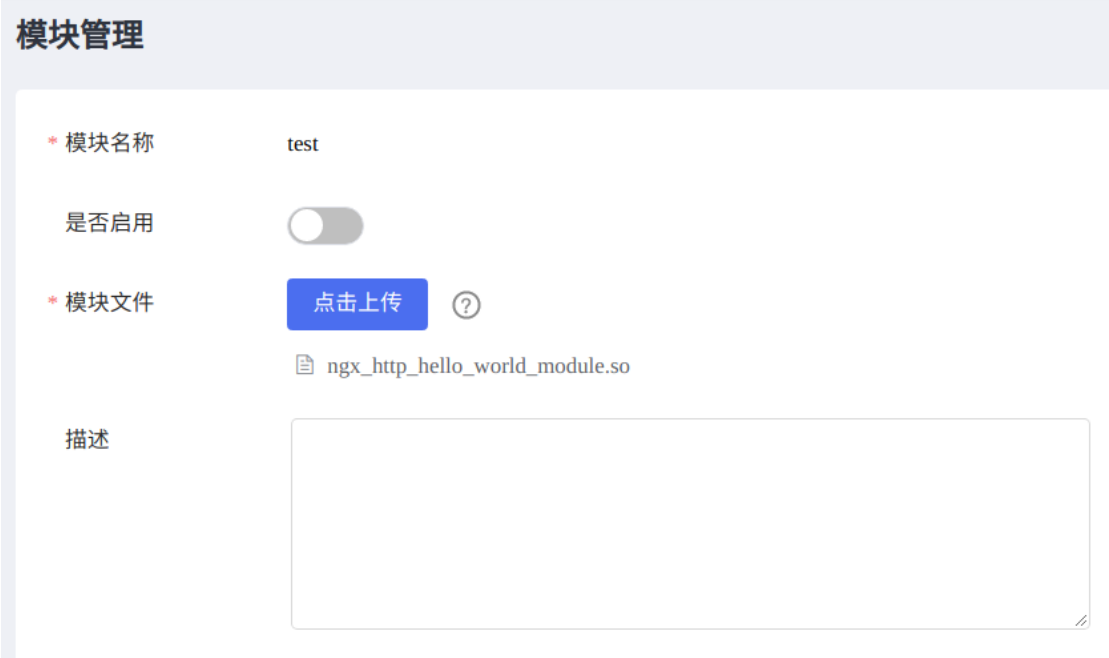


图 4-30 编辑模块

3. 修改模块的属性，点击“**保存**”按钮保存修改的属性。

4.6.3 删除模块

管理员在管理中心删除模块：

1. 在管理中心左侧导航区点击“**模块管理**”，进入“**模块管理**”页面。
2. 在“**模块管理**”页面，勾选要删除的模块名称，点击“**删除**”按钮删除即可。

4.7 证书管理

4.7.1 证书列表

在管理中心左侧导航区点击“**证书管理**”，操作区域显示“**证书列表**”页面，显示所有已添加到管理中心的证书。

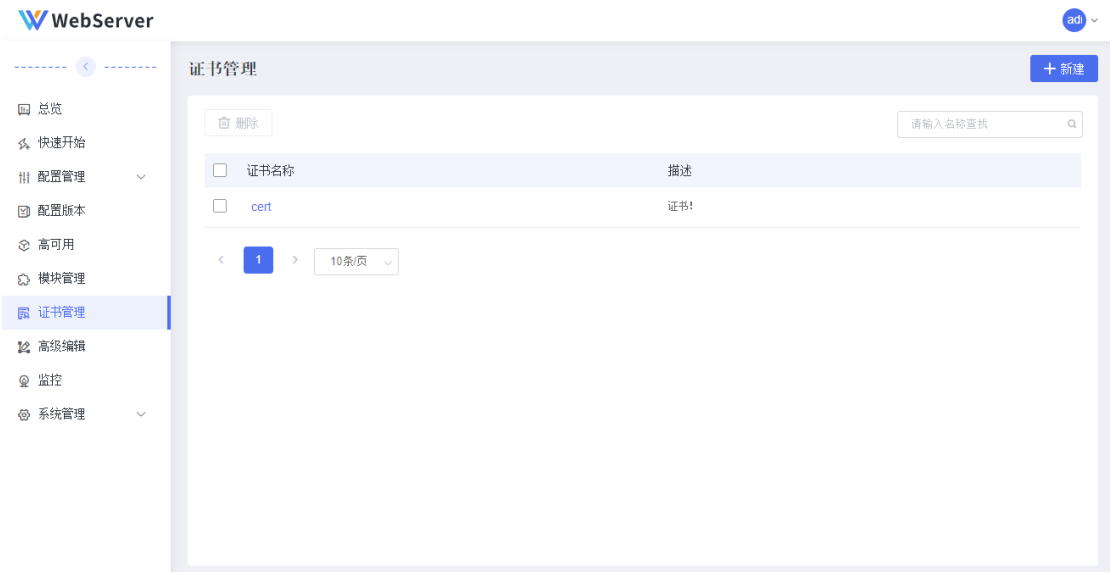


图 4-31 证书列表页面

4.7.2 新建证书

管理员在管理中心新建证书:

1. 在管理中心左侧导航区点击“**证书管理**”，进入”**证书列表**“页面。
2. 在“**证书列表**”页面，点击“**新建**”按钮，进入“**新建证书**”页面，可配置项如下:

（1）选择方式为“**上传证书**”新建证书。

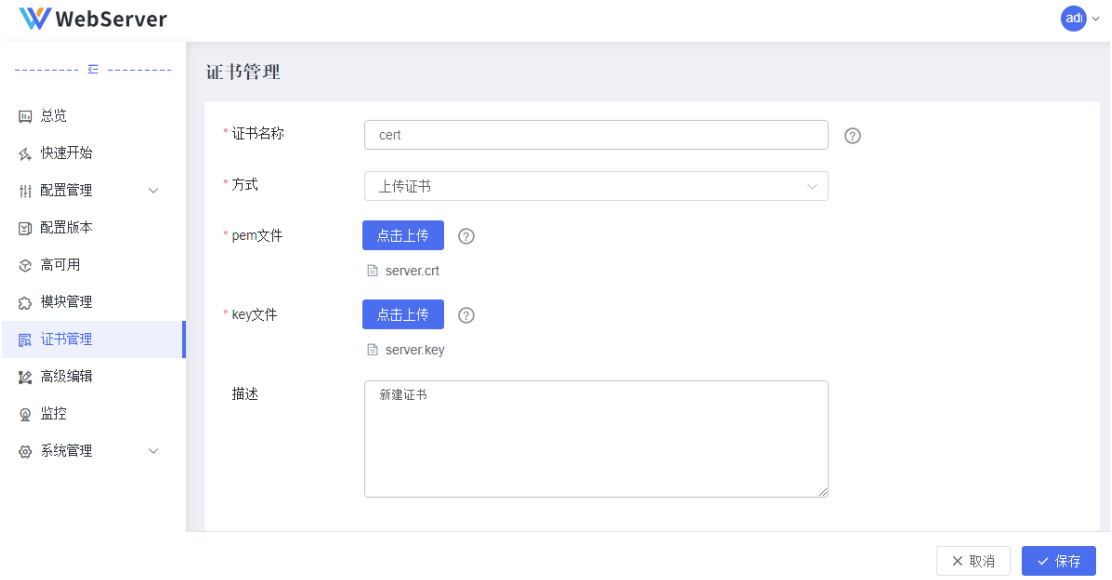


图 4-32 上传证书

（2）选择方式为“**输入本地路径**”新建证书。

图 4-33 输入本地路径

表 4-24 新建证书配置项

配置项名称	说明
证书名称	证书的名称。
方式	可以选择上传证书和输入本地路径两种方式上传证书。
pem文件	证书文件，只能上传crt、pem类型的文件。
key文件	证书密钥文件，只能上传key、pem类型的文件。
描述	证书的描述信息。

3. 配置证书的所有属性，点击“**保存**”按钮完成新建证书。

4.7.3 编辑证书

管理员在管理中心编辑证书：

1. 在管理中心左侧导航区点击“**证书管理**”，进入“**证书列表**”页面。

2. 在“**证书列表**”页面，点击“**证书名称**”链接，进入“**编辑证书**”页面，可编辑属性如下：

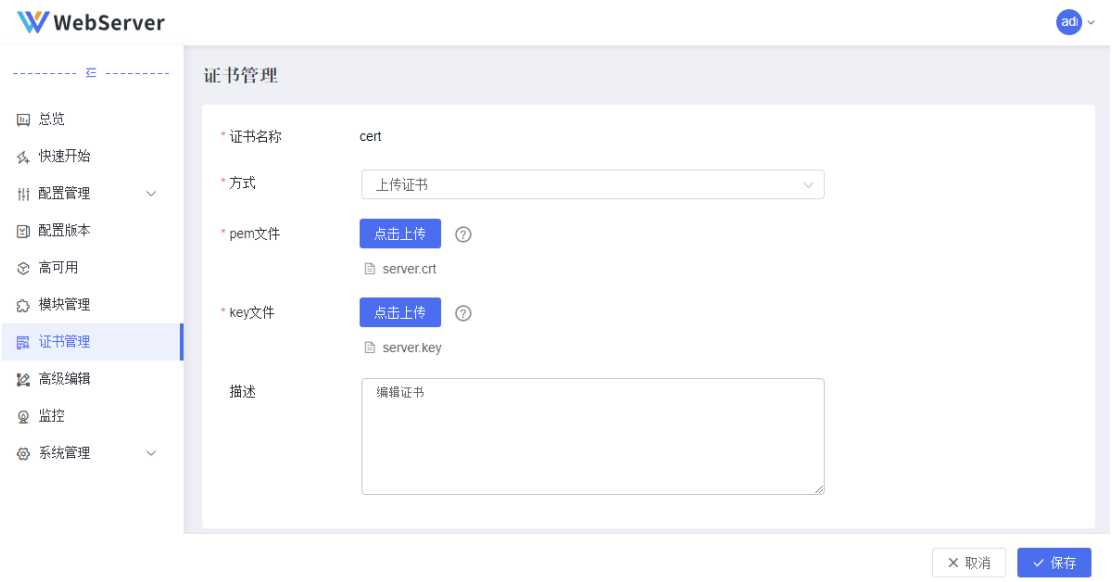


图 4-34 编辑证书页面1

3. 修改证书的属性，点击“**保存**”按钮保存修改的属性。

4.7.4 删除证书

管理员在管理中心删除证书：

1. 在管理中心左侧导航区点击“**证书管理**”，进入“**证书列表**”页面。
2. 在“**证书列表**”页面，勾选要删除的证书，点击“**删除**”按钮删除即可。

4.8 高级编辑

对于用户的定制化需求，BES WebServer提供了高级编辑功能，用户可以直接修改配置文件完成定制化配置。为了方便配置文件的维护，用户也可以创建多个配置文件，然后使用include指令完成引入。

在管理中心左侧导航区点击“**高级编辑**”，操作区域显示“**高级编辑**”页面，显示所有已添加到管理中心的配置文件。

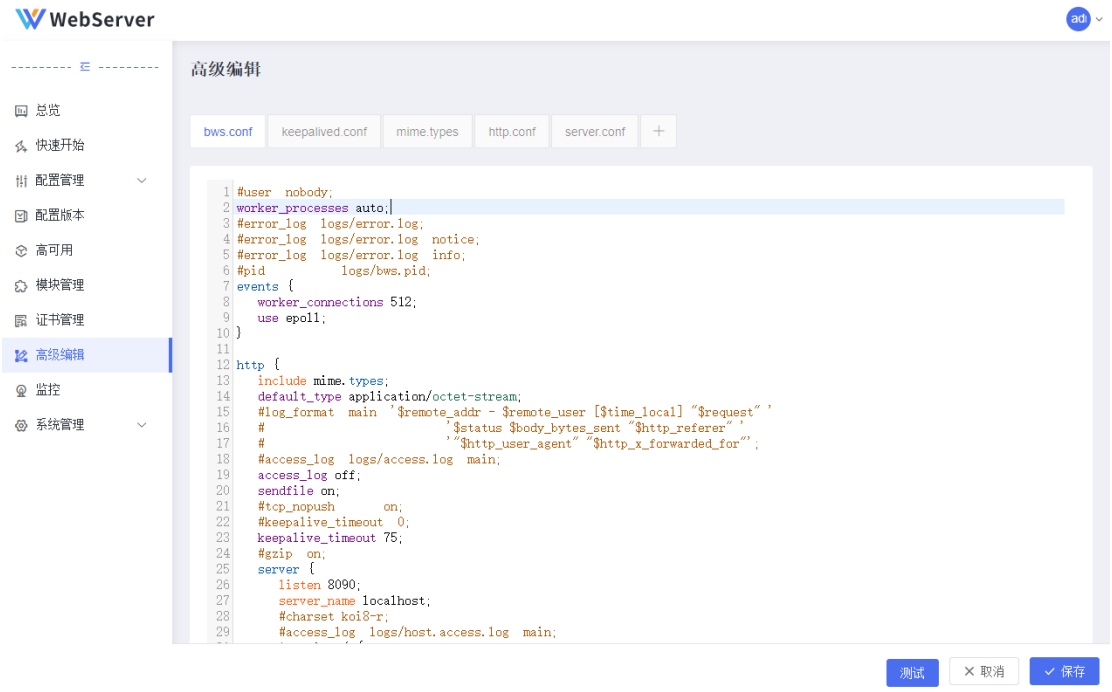


图 4-35 高级编辑页面

用户在修改完配置文件保存之后，可以使用“**测试**”按钮测试配置文件的正确性。

4.8.1 新增配置文件

管理员在管理中心新增配置：

1. 在管理中心左侧导航区点击“**高级编辑**”，进入“**高级编辑**”页面。
2. 点击“+”按钮，弹出“**新增配置**”弹窗，输入配置文件名称，点击“**保存**”按钮，进入新增配置文件页。

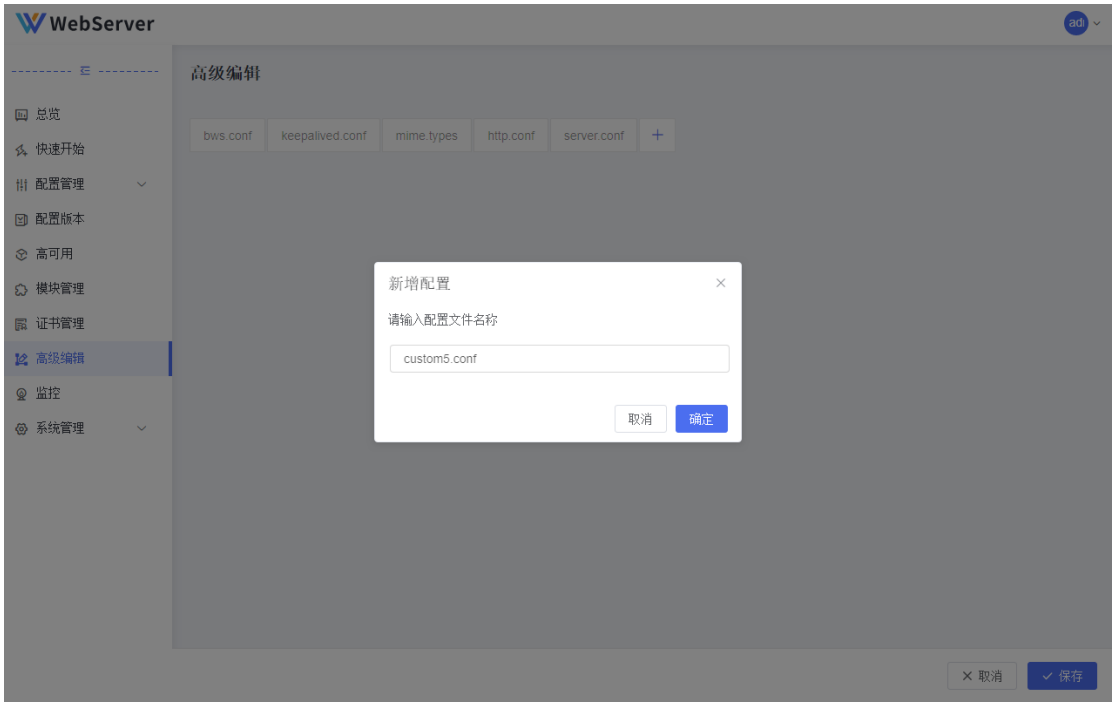


图 4-36 新增配置文件

3. 添加配置信息，点击“**保存**”按钮，新增配置文件成功。

4.8.2 编辑配置文件

管理员在管理中心编辑配置文件：

1. 在管理中心左侧导航区点击“**高级编辑**”，进入“**高级编辑**”页面。
2. 点击需要编辑的“**配置文件名称**”，修改配置信息，点击“**保存**”按钮保存编辑后的配置。

4.8.3 删除自定义配置文件

管理员在管理中心删除自定义配置文件：

1. 在管理中心左侧导航区点击“**高级编辑**”，进入“**高级编辑**”页面。
2. 点击需要删除的自定义配置文件右边的“**X**”按钮，弹出二次确认弹窗，点击“**确定**”按钮删除自定义配置文件成功。

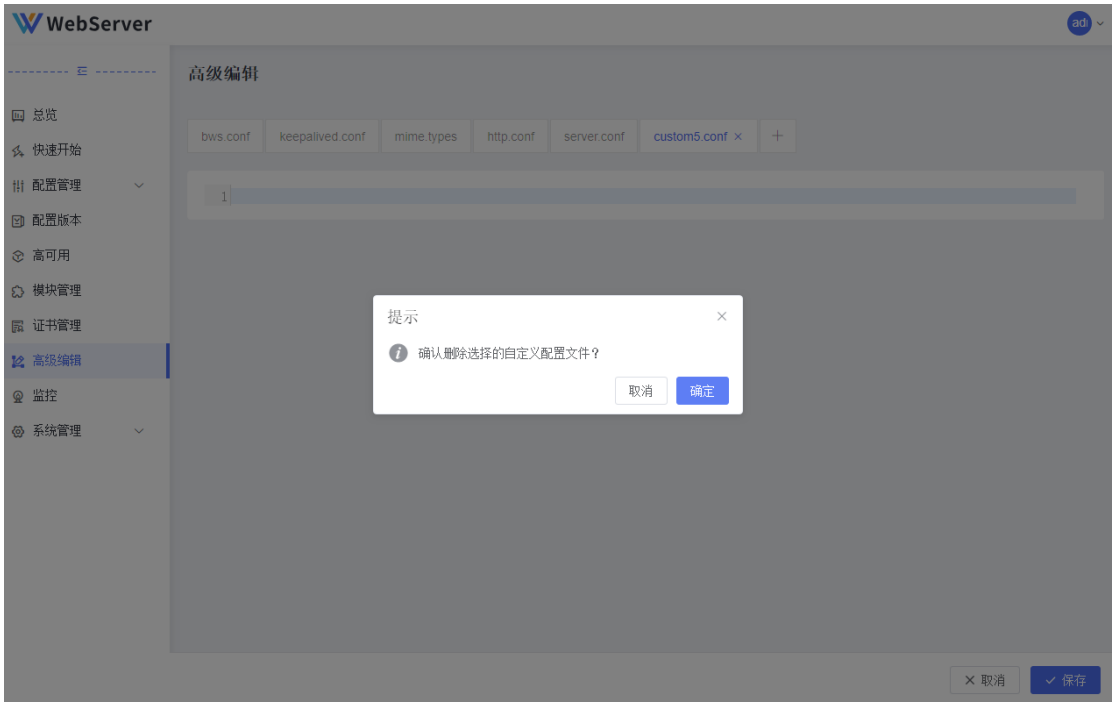


图 4-37 删除自定义配置文件

4.9 监控

4.9.1 开启监控

在管理控制台配置管理–>虚拟主机列表，选择要监控的虚拟主机，然后点击超链接进入虚拟主机编辑页面，将“**是否监控**”选项打开，保存。然后在“**总览**”页面，启动BES WebServer实例。

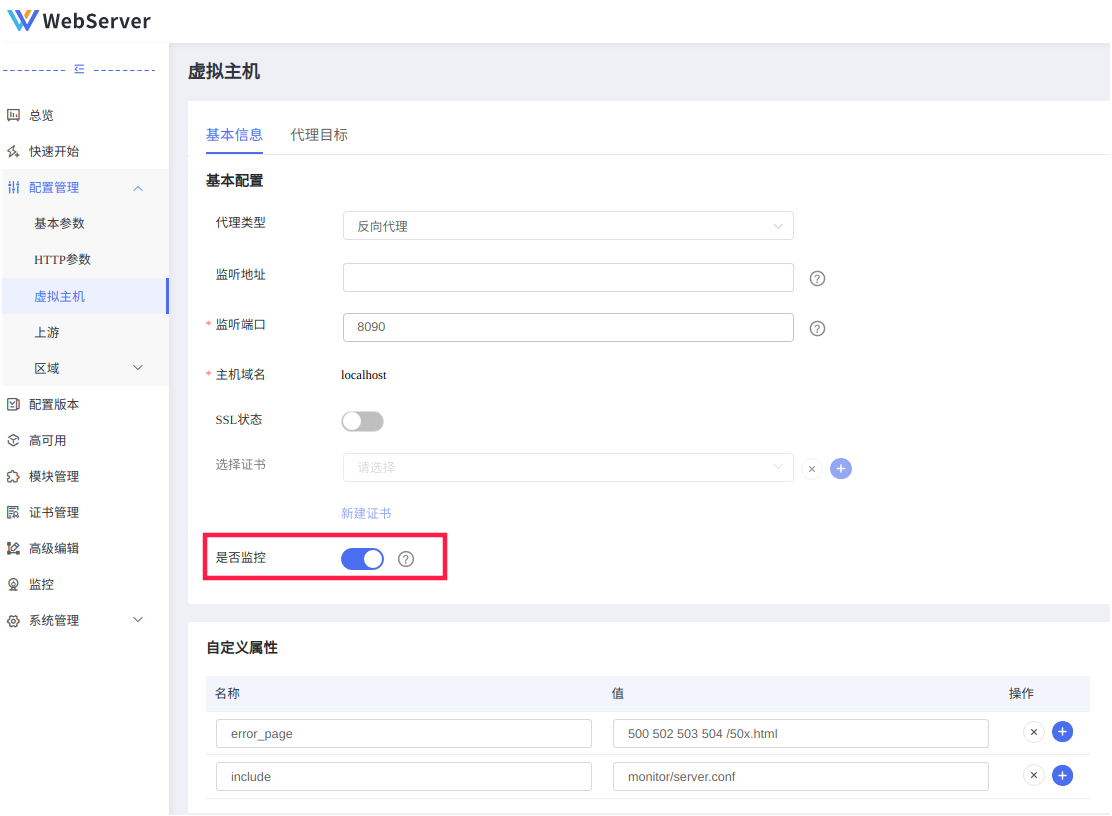


图 4-38 开启监控

4.9.2 虚拟主机监控

在管理中心查看虚拟机监控：

在管理中心左侧导航区点击“**监控**”–>“**虚拟主机**”，进入“**虚拟主机监控**”页面。支持选择“**虚拟主机域名**”查看监控信息，可以查看当前使用（包括活动连接数、空闲状态连接数、读状态连接数和写状态连接数）、总数（包括已接受连接总数、已处理连接总数和请求总数）、请求状态占比、客户端占比、请求数、域名访问量和平均响应时间。

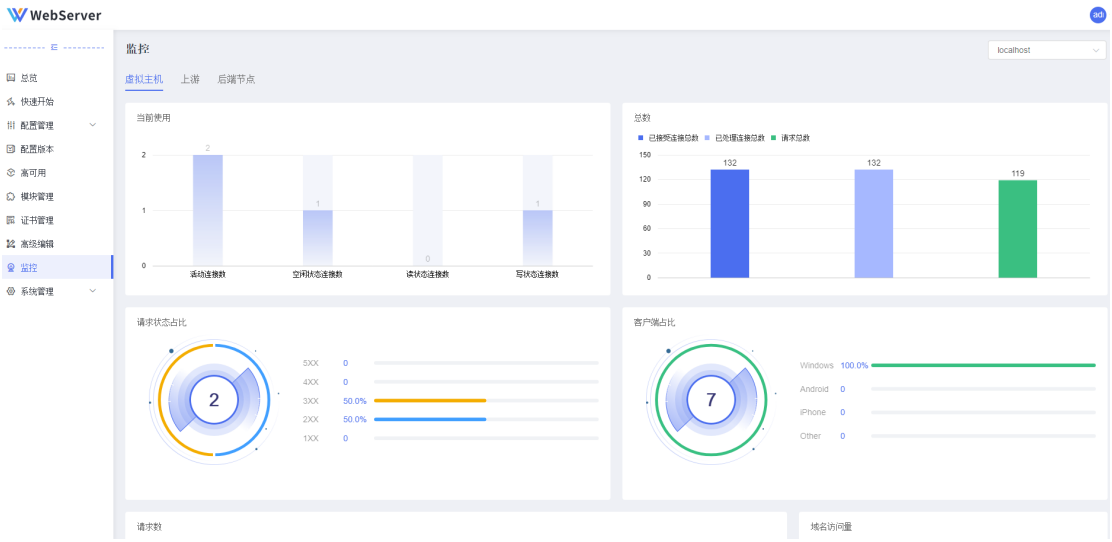


图 4-39 虚拟主机监控

4.9.3 上游监控

在管理中心查看上游监控：

在管理中心左侧导航区点击“**监控**”->“**上游**”，进入“**上游监控**”页面。支持选择“**上游名称**”查看监控信息，可以查看请求状态占比、客户端占比、请求数、域名访问量、平均响应时间、健康检查和后端节点（包括名称、状态、IP、端口、成功次数、失败次数、检查协议、检查端口）。

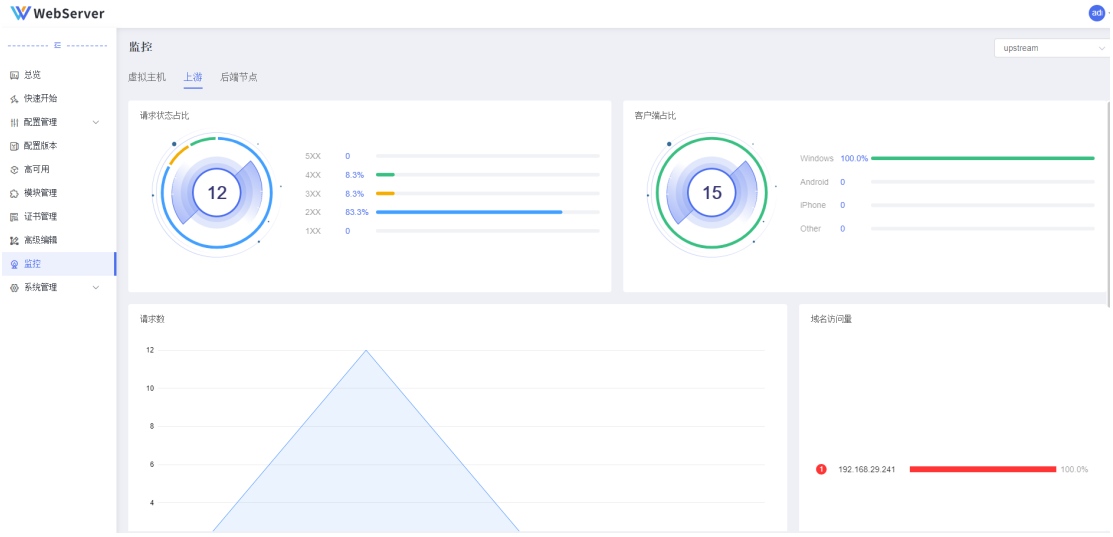


图 4-40 上游监控

4.9.4 后端节点监控

在管理中心查看后端节点监控：

在管理中心左侧导航区点击“**监控**”->“**后端节点**”，进入“**后端节点监控**”页面。支持通过根据虚拟主机查询和根据上游查询选择“**后端节点**”查看监控信息，可以查看请求状态占比、请求数、平均响应时间、健康检查和节点信息。

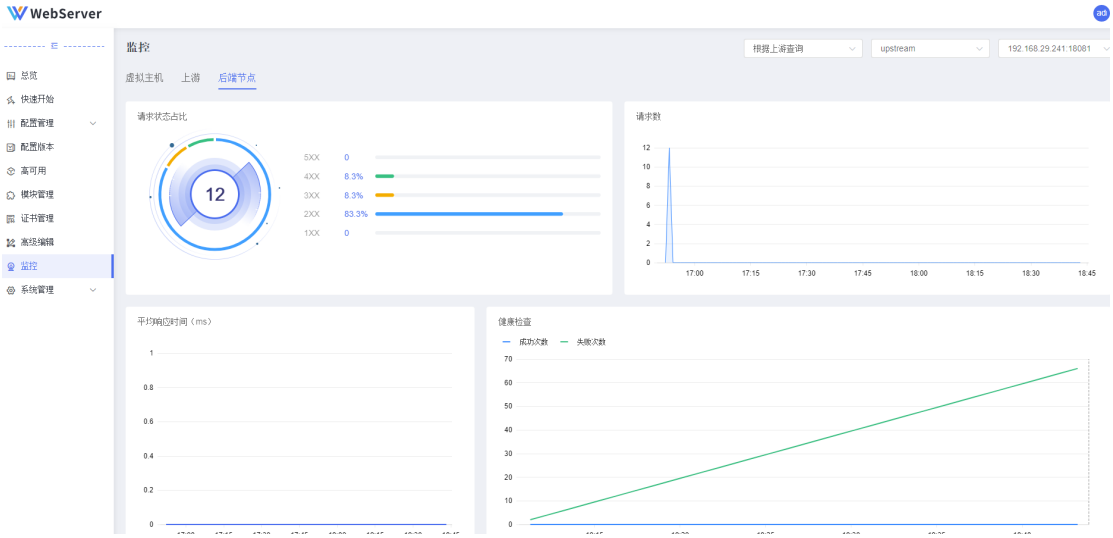


图 4-41 后端节点监控

4.9.5 支持Prometheus监控

（1）在Prometheus的prometheus.yml添加如下内容：

```
#prometheus配置
- job_name: 'bws-prometheus'
  metrics_path: /bws_status/format/prometheus
  static_configs:
      - targets: ['192.168.17.209:8090']     --此处配置BWS实例的地址和端口
```

（2）启动Prometheus，访问Prometheus控制台，即可看到BES WebServer各监控项。

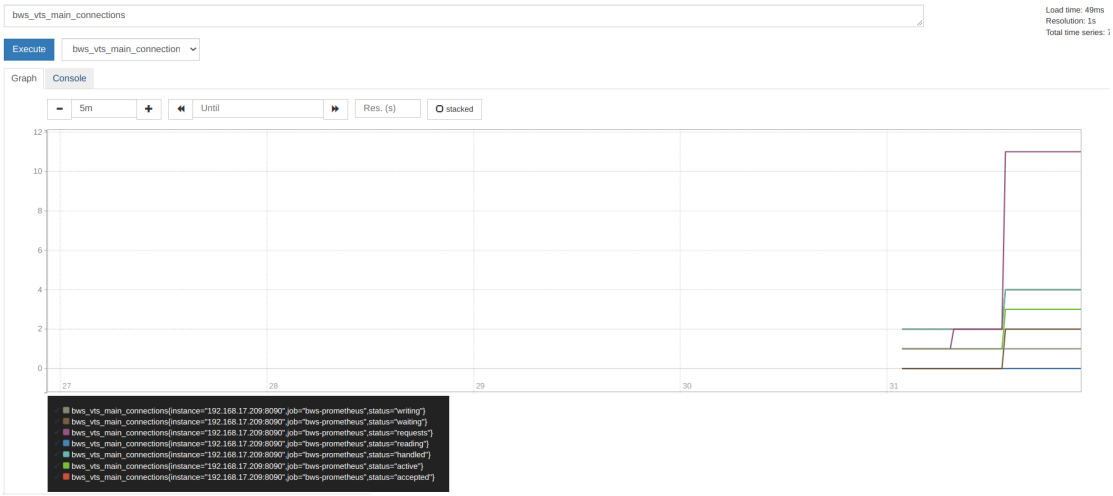


图 4-42 后端节点监控

各指标含义：

bws_vts_filter_bytes_total

根据不同过滤条件分组的字节流量统计，包括接收和发送。其中，direction=“in”表示接收，direction=“out”，表示发送。这里提供了以下过滤条件：

filter=“agent::xxx”，根据客户端进行分组

filter=“domain::xxx”，根据域名进行分组

图 4-43 bws_vts_filter_bytes_total监控项

bws_vts_filter_cache_total

根据不同条件进行分组的缓存统计，filter=“agent::”表示按客户端进行分组，其中各指标如下：

status=“bypass”，缓存绕过的次数

status=“expired”，缓存到期的次数

status=“hit”，缓存命中的次数

status=“miss”，缓存未命中的次数

status=“revalidated”，缓存重新验证的次数

status=“scarce”，缓存告警的次数

status=“stale”，缓存过时的次数

status=“updating”，缓存更新的次数

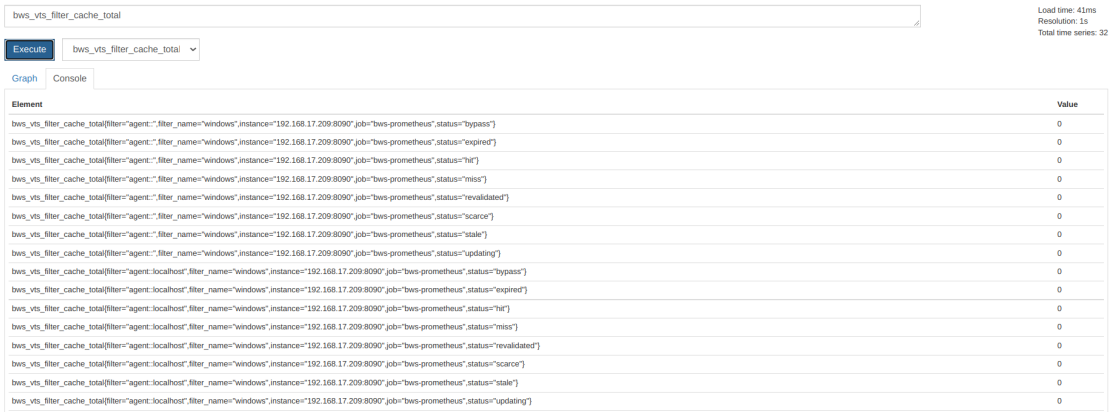


图 4-44 bws_vts_filter_cache_total监控项

bws_vts_filter_request_seconds

根据不同条件分组的请求时间，提供了如下分组条件

filter=“agent::xxx”，根据客户端进行分组

filter=“domain::xxx”，根据域名进行分组

图 4-45 bws_vts_filter_request_seconds监控项

bws_vts_filter_request_seconds_total

根据不同条件分组的请求时间，提供了如下分组条件

filter=“agent::xxx”，根据客户端进行分组

filter=“domain::xxx”，根据域名进行分组

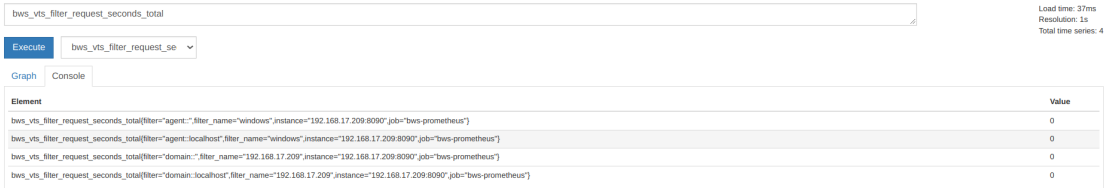


图 4-46 bws_vts_filter_request_seconds_total监控项

bws_vts_filter_requests_total

根据不同条件分组的请求总时间，提供了如下分组条件

filter=“agent::xxx”，根据客户端进行分组

filter=“domain::xxx”，根据域名进行分组

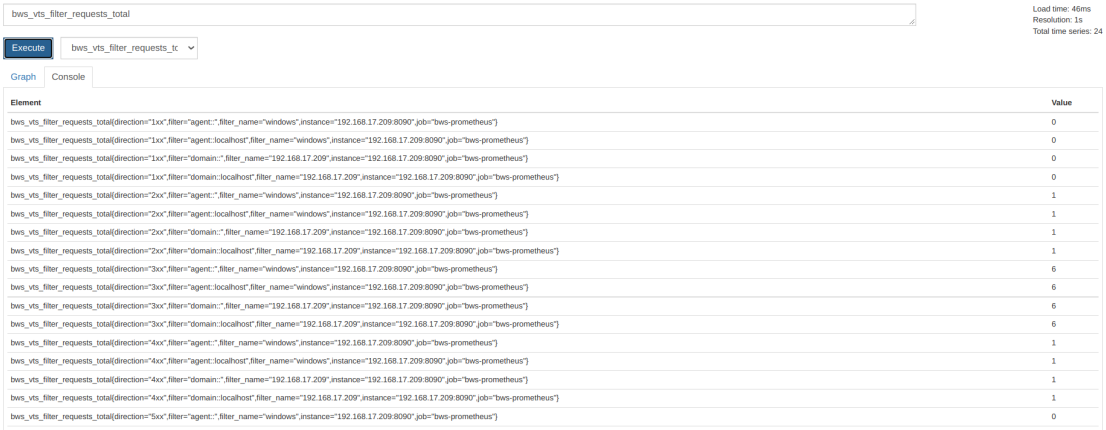


图 4-47 bws_vts_filter_requests_total监控项

bws_vts_info

BES WebServer实例版本信息。

图 4-48 bws_vts_info监控项

bws_vts_main_connections

BES WebServer实例的总连接信息，包括已接受连接、活跃连数、读状态连接数、请求连接数、等待连接数和写状态连接数

图 4-49 bws_vts_main_connections监控项

bws_vts_main_shm_usage_bytes

共享内存使用信息统计，其中

shared=“max_size”，配置中指定的最大共享内存大小限制

shared=“used_node”，当前在共享内存中使用的节点数

shared=“used_size”，当前已使用的共享内存大小

图 4-50 bws_vts_main_shm_usage_bytes监控项

bws_vts_server_bytes_total

根据虚拟主机分组的流量统计，包括接收和发送，其中direction=“in”为接收，direction=“out”为发送。host=“*”表示所有虚拟主机的流量。

图 4-51 bws_vts_server_bytes_total监控项

bws_vts_server_cache_total

根据虚拟主机分组的缓存统计。各个指标如下

status=“bypass”，缓存绕过的次数

status=“expired”，缓存到期的次数

status=“hit”，缓存命中的次数

status=“miss”，缓存未命中的次数

status=“revalidated”，缓存重新验证的次数

status=“scarce”，缓存告警的次数

status=“stale”，缓存过时的次数

status=“updating”，缓存更新的次数

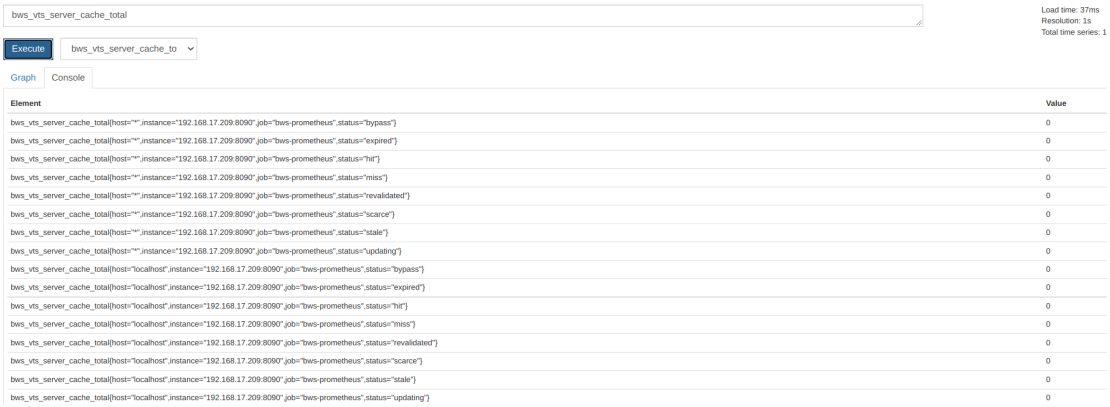


图 4-52 bws_vts_server_cache_total监控项

bws_vts_server_request_seconds

根据虚拟主机分组的请求时间，host=“*”表示所有虚拟主机的总和

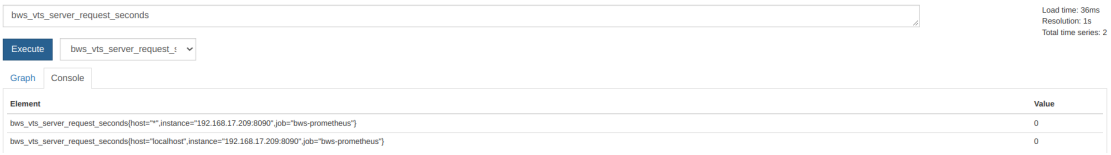


图 4-53 bws_vts_server_request_seconds监控项

bws_vts_server_request_seconds_total

根据虚拟主机分组的总请求时间，host=“*”表示所有虚拟主机的总和

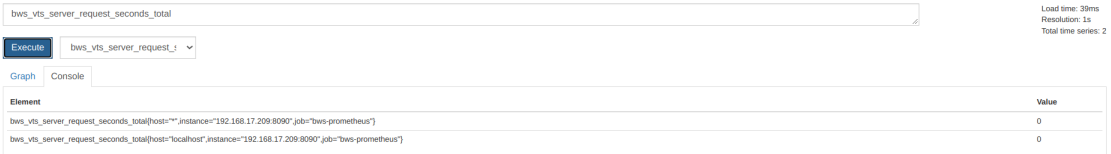


图 4-54 bws_vts_server_request_seconds_total监控项

bws_vts_server_requests_total

统计请求响应码次数，有1xx、2xx、3xx、4xx、5xx和total，total代表所有请求响应码次数之和。

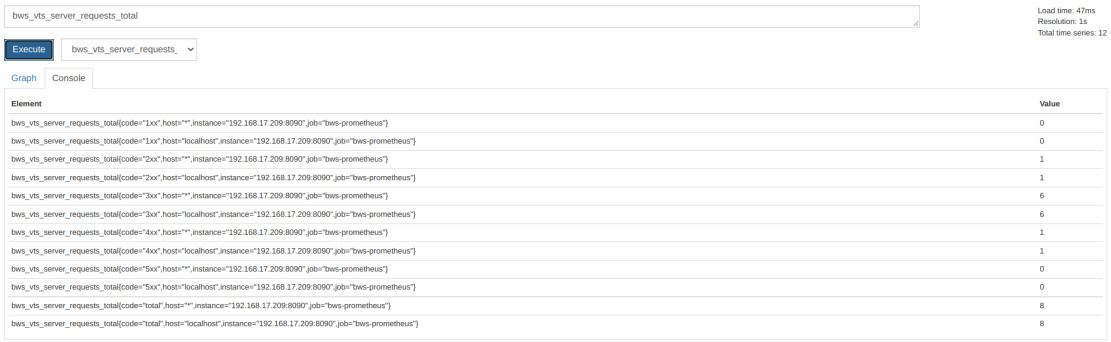


图 4-55 bws_vts_server_requests_total监控项

bws_vts_start_time_seconds

实例启动时间

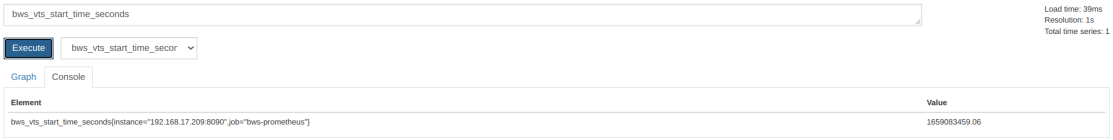


图 4-56 bws_vts_start_time_seconds监控项

4.10 系统管理

4.10.1 用户管理

BES WebServer提供默认的管理员admin（系统组、安全组和审计组）、审计员auditadmin（审计组）和安全管理员secadmin（安全组），这三个用户不可删除。

系统管理用户可以新建用户组，并赋予相应的角色。

为方便管理用户密码，安全管理员secadmin可以在控制台使用“**安全策略**”对登录密码长度、连续登录失败次数、限制登录间隔和用户密码有效时间进行设置。

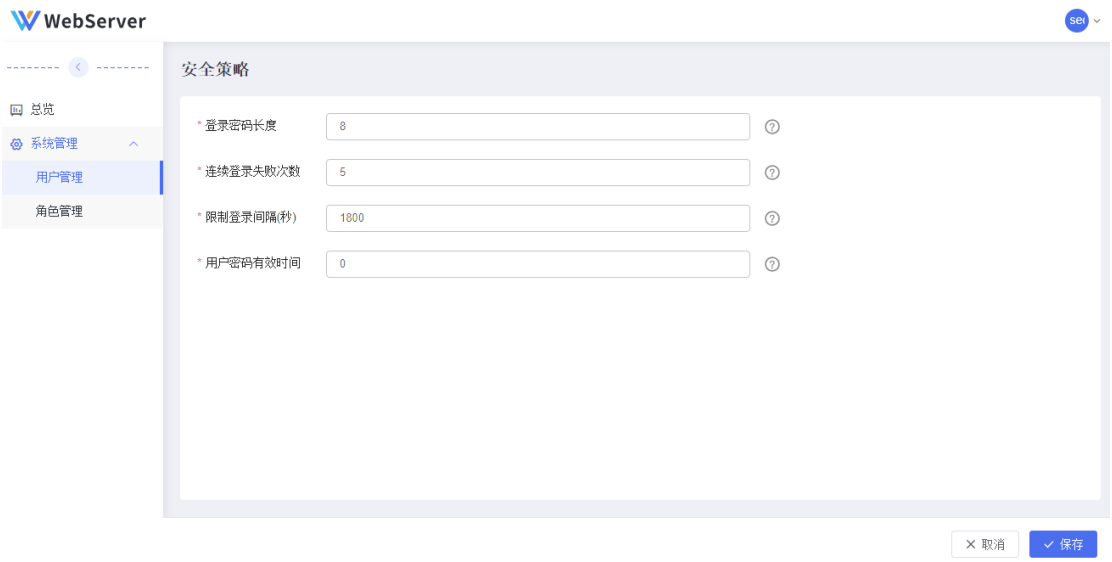


图 4-57 安全策略页面

表 4-25 安全策略配置项

配置项名称	说明	默认值
登录密码长度	用户登录密码的长度，合法值：8-2147483647。登录密码将不小于所配置的长度。	8
连续登录失败次数	用户登录密码鉴权次数，合法值：1到5的整数。用户登录密码错误次数达到鉴权次数将不能继续登录。	5
限制登录间隔（秒）	用户登录密码错误鉴权间隔时间，合法值：1800(30分钟)-86400。用户登录失败间隔时间在此值之内，将会累计鉴权次数。	1800
用户密码有效时间	用户密码有效时间，合法值：0-2147483647，输入0时代表不过期。单位：天。	0

4.10.1.1 用户列表

在管理中心左侧导航区点击“**系统管理**”->“**用户管理**”，操作区域显示“**用户列表**”页面，显示所有已添加到管理中心的用户，默认用户如下。

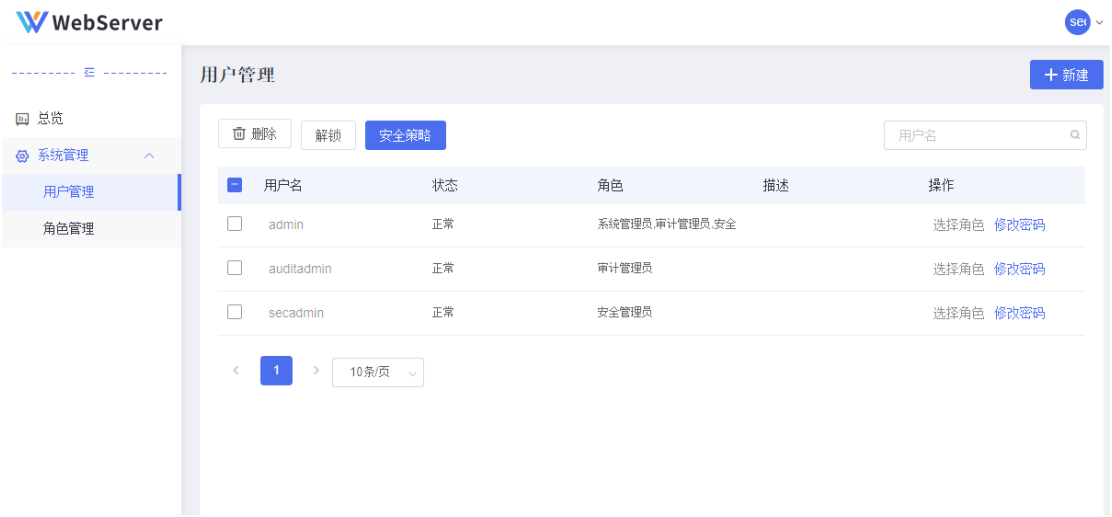


图 4-58 用户列表

4.10.1.2 新建用户

新建用户之前，需要先创建角色，有关角色相关介绍，请参考角色管理章节。

管理员在管理中心新建管理用户：

1) 在管理中心左侧导航区点击“**系统管理**”–>“**用户管理**”，进入“**用户列表**”页面。

2) 在“**用户列表**”页面，点击“**新建**”按钮，进入“**新建用户**”页面，可配置项如下：

图 4-59 新建用户

表 4-26 新建用户配置项

配置项名称	说明
用户名称	用户的名称。
密码	用户密码。
确认密码	再次输入用户密码。
描述	用户的描述信息。

3) 配置上用户的所有属性，点击“**保存**”按钮完成新建用户。

4.10.1.3 分配角色

安全管理员新建完用户之后，需要在用户列表页面进行角色的分配，在“**用户列表**”页面，点击“**选择角色**”超链接。

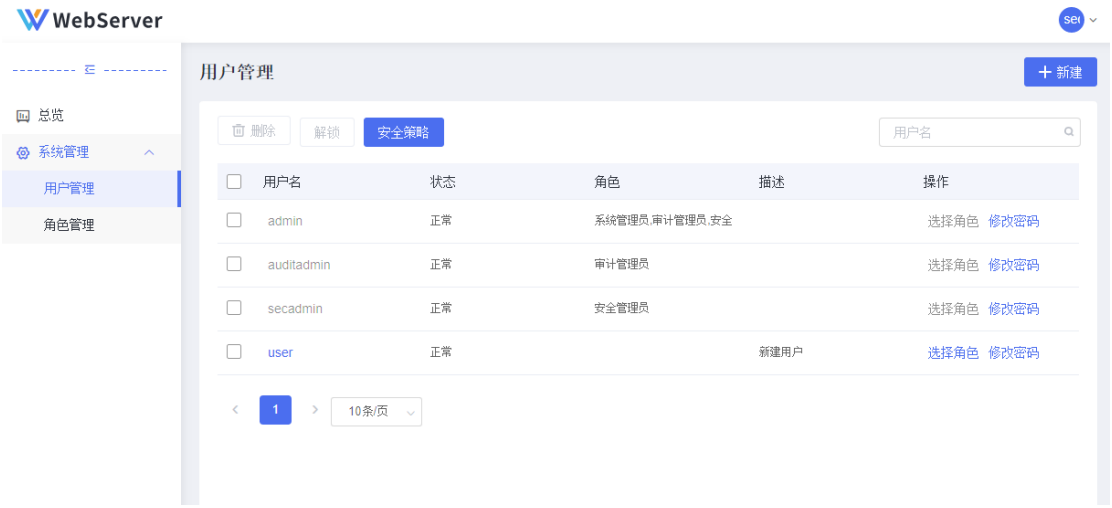


图 4-60 新建用户选择角色

进入“**选择角色**”页面，为当前新建的用户选择角色。

图 4-61 选择角色页面

点击“**保存**”按钮完成角色的选择。

4.10.1.4 编辑用户

管理员在管理中心编辑用户：

1) 在管理中心左侧导航区点击“**系统管理**”–>“**用户管理**”，进入“**用户列表**”页面。

2) 在“**用户列表**”页面，点击“**用户名**”链接，进入“**编辑用户**”页面，可编辑项目如下：

图 4-62 编辑用户页面

管理员可以禁用当前用户，禁用之后，当前用户将无法登录。

3) 修改用户的属性，点击“**保存**”按钮保存修改的属性。

4.10.1.5 解锁用户

当用户登录时，连续输入密码错误超过安全策略设置的次数，用户会被锁定，可以使用安全管理员secadmin来解锁用户

1) 在管理中心左侧导航区点击“**系统管理**”–>“**用户管理**”，进入“**用户列表**”页面。勾选要解锁的用户，点击“**解锁**”按钮即可。

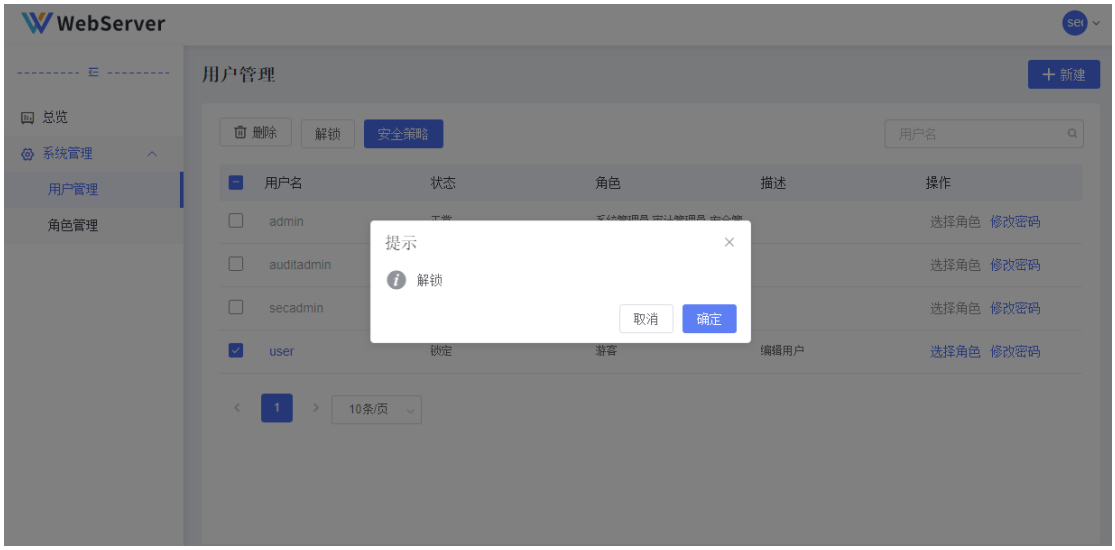


图 4-63 解锁用户

4.10.1.6 修改用户密码

用户可以修改自己的密码，或者由安全管理员修改密码。

1) 安全管理员修改普通用户密码，在“**用户列表**”页面，点击要修改密码用户的操作栏“**修改密码**”超链接，进入修改密码页面，输入新密码和确认密码，点击“**保存**”按钮即可完成密码修改。

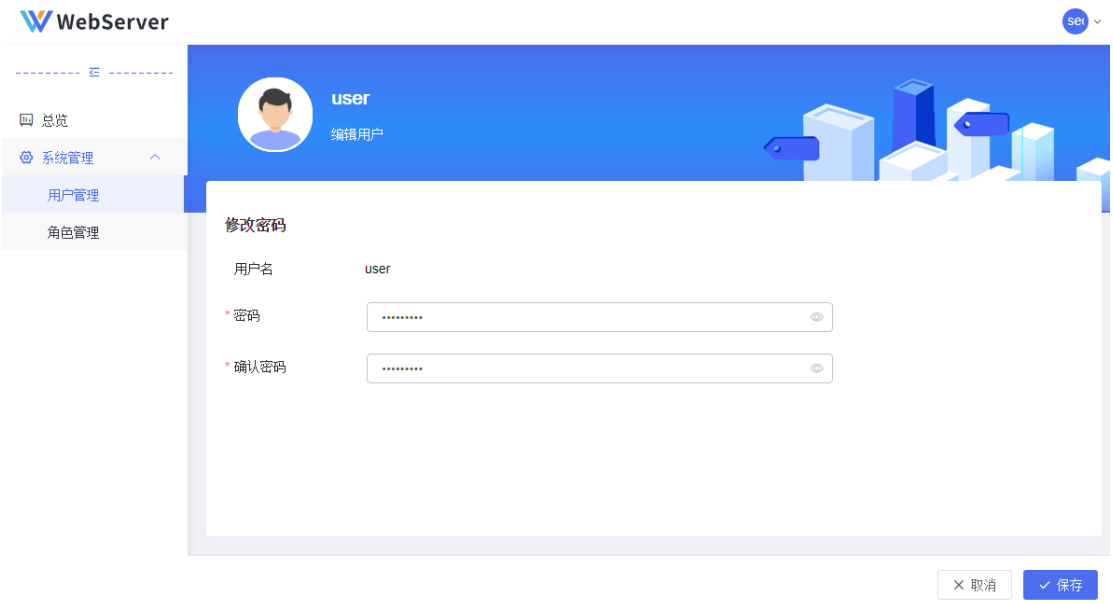


图 4-64 安全管理员修改普通用户密码

2) 普通用户修改自己的密码，可以在管理控制台右上角，点击“**个人信息**”–>“**修改密码**”，进入“**修改密码**”详情页面。输入原始密码、新密码和确认密码完成密码修改。

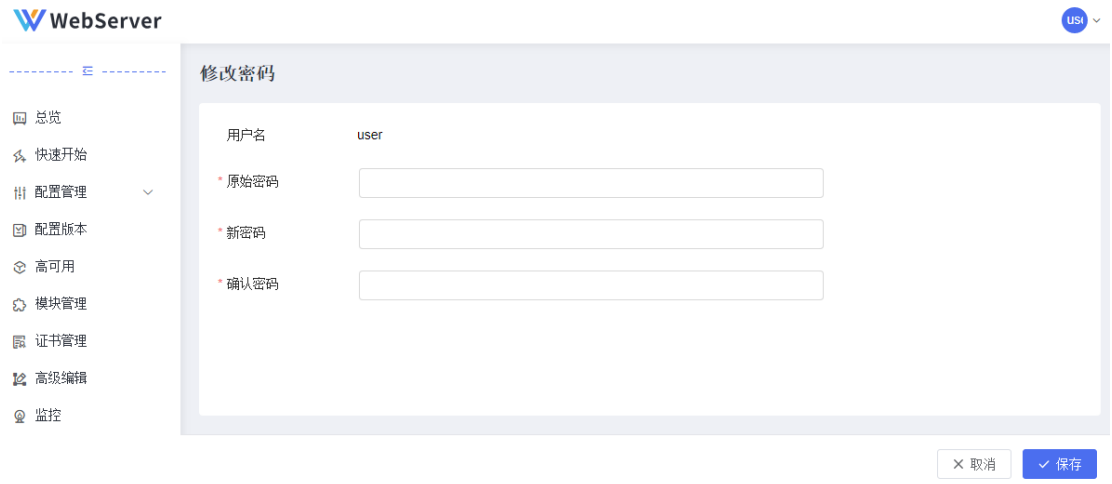


图 4-65 普通用户修改密码详情页

4.10.1.7 删除用户

管理员在管理中心删除用户：

1. 在管理中心左侧导航区点击“**系统管理**”–>“**用户管理**”，进入“**用户列表**”页面。
2. 在“**用户列表**”页面，勾选要删除的用户，点击“**删除**”按钮删除即可。

图 4-66 删除用户

4.10.2 角色管理

使用具有角色管理权限的用户登录管理中心，具有该权限的默认用户为secadmin，密码为B#2008_2108#es。

4.10.2.1 角色列表

在管理中心左侧导航区点击“**系统管理**”–>“**角色管理**”，进入“**角色列表**”页面，显示所有已添加到管理中心的角色名。默认存在如下角色：

图 4-67 角色列表

用户可以根据自己的实际场景选择合适的角色或者新建角色分配相应的权限。

4.10.2.2 新建角色

在管理中心新建角色：

1) 在管理中心左侧导航区点击“**系统管理**”–>“**角色管理**”，点击“**新建**”按钮进入“**新建角色**”页面。

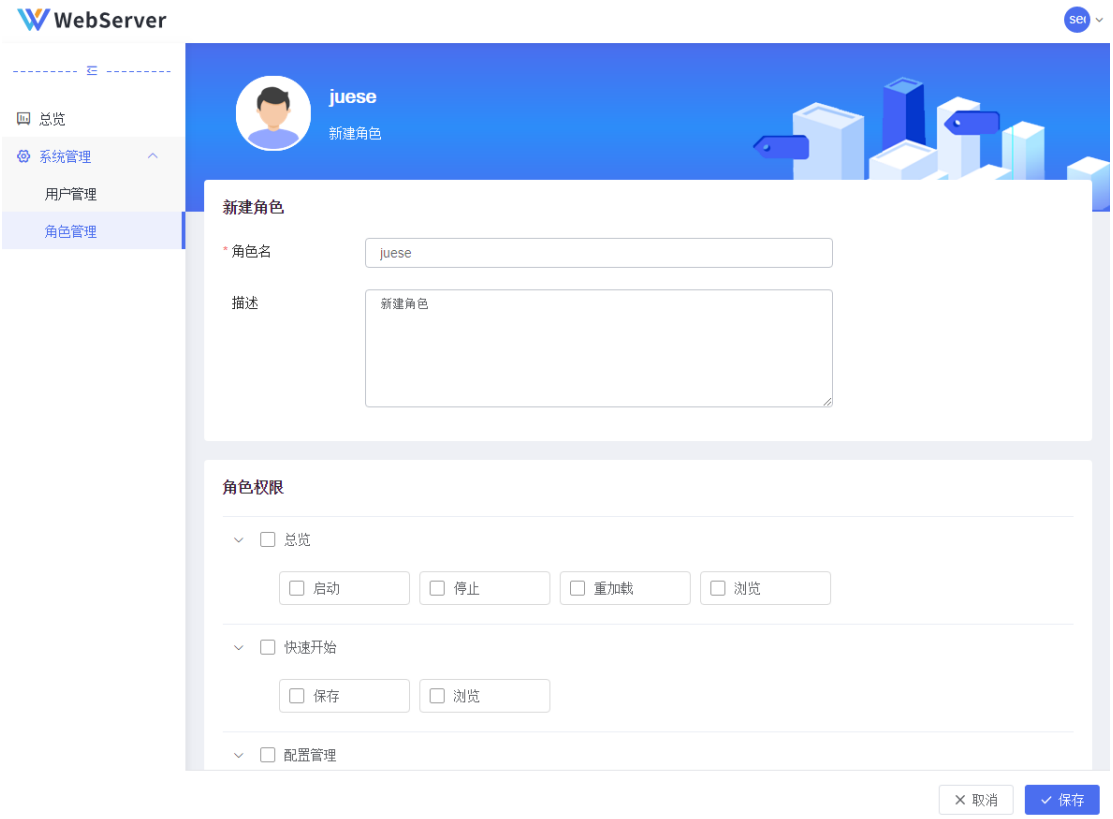


图 4-68 新建角色

2. 角色权限的每一个功能点，都对应有详细的操作权限，用户可以按需选择。

图 4-69 角色权限详情

2) 在“**新建角色**”页面，勾选当前用户要分配的权限，点击保存按钮，完成角色新建。

4.10.2.3 编辑角色

管理员在管理中心编辑角色：

1. 在管理中心左侧导航区点击“**系统管理**”->“**角色管理**”，进入“**角色列表**”页面。
2. 在“**角色列表**”页面，点击“**角色名**”链接，进入“**编辑角色**”页面，可编辑项目如下：

图 4-70 编辑角色

3. 修改角色的属性，点击“**保存**”按钮保存修改的属性。

4.10.2.4 删除角色

管理员在管理中心删除角色：

1. 在管理中心左侧导航区点击“**系统管理**”->“**角色管理**”，进入“**角色列表**”页面。
2. 在“**角色列表**”页面，勾选要删除的角色，点击“**删除**”按钮删除即可。

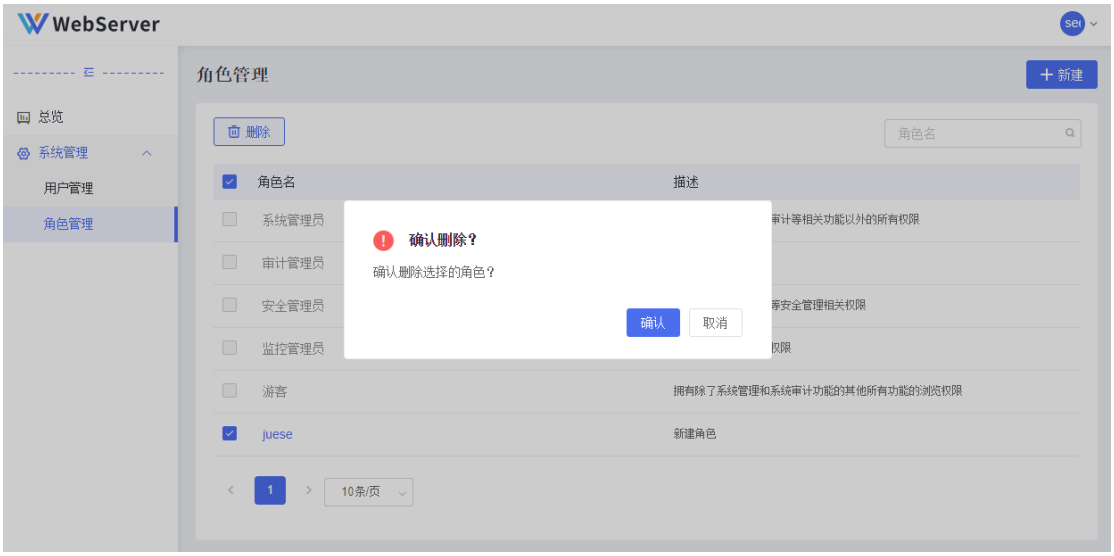


图 4-71 删除角色

4.10.3 系统审计

4.10.3.1 登录审计

BES WebServer管理平台可以对用户登录和注销操作进行审计。主要支持按用户名、按源IP、按操作接口、按操作对象、按状态和按操作时间进行查询，使用具有审计权限的用户登录管理中心，具有该权限的默认用户为auditadmin，密码为B#2008_2108#es。

审计主要内容如下：

操作时间、用户名、源IP、操作动作（登录和注销）、操作接口、操作对象和状态（正常和异常）。

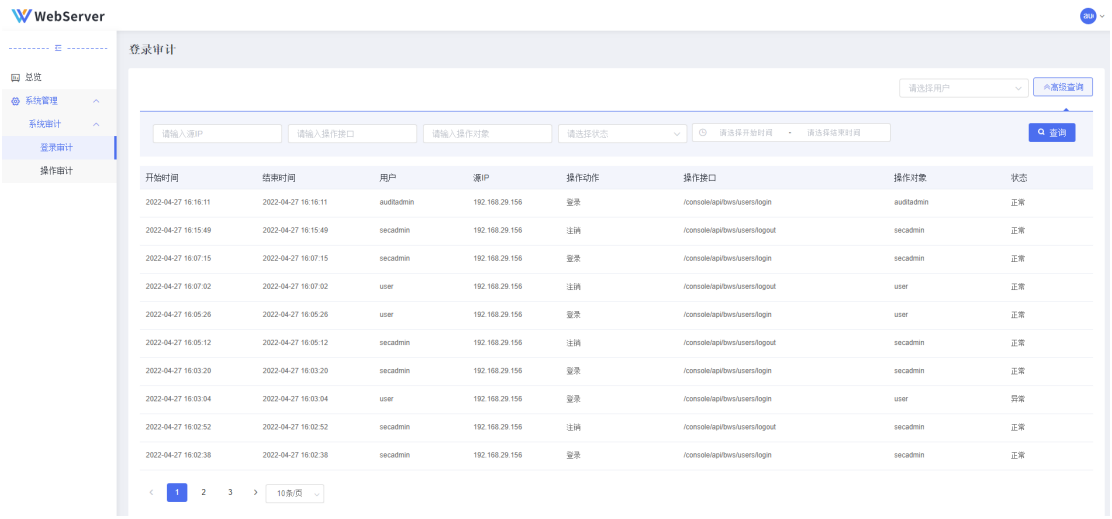


图 4-72 登录审计

4.10.3.2 操作审计

BES WebServer管理平台可以对用户操作进行审计。主要支持按用户名、按源IP、按操作接口、按操作对象、按状态和按操作时间进行查询。

审计主要内容如下：

操作时间、用户名、源IP、操作动作、操作接口、操作对象和状态（正常和异常）。

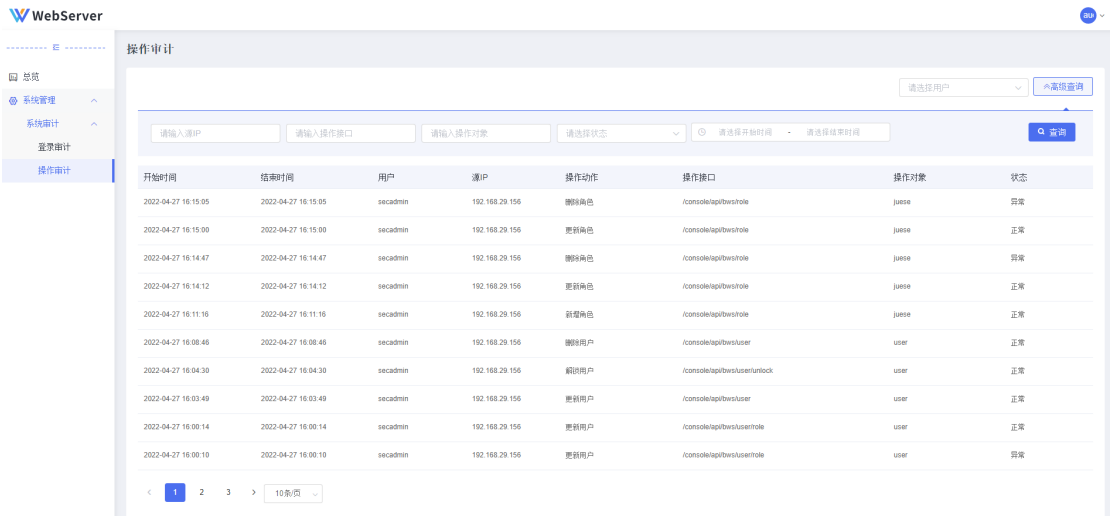

WebServer
操作审计
总览
系统管理
系统审计
登录审计
操作审计
请选择用户
高级查询
请输入源IP
请输入操作接口
请输入操作对象
请选择状态
请选择开始时间
请选择结束时间
查询
开始时间 结束时间 用户 源IP 操作动作 操作接口 操作对象 状态
2022-04-27 16:15:05 2022-04-27 16:15:05 secadmin 192.168.29.156 删除角色 /console/api/bws/role juese 异常
2022-04-27 16:15:00 2022-04-27 16:15:00 secadmin 192.168.29.156 更新角色 /console/api/bws/role juese 正常
2022-04-27 16:14:47 2022-04-27 16:14:47 secadmin 192.168.29.156 删除角色 /console/api/bws/role juese 异常
2022-04-27 16:14:12 2022-04-27 16:14:12 secadmin 192.168.29.156 更新角色 /console/api/bws/role juese 正常
2022-04-27 16:11:16 2022-04-27 16:11:16 secadmin 192.168.29.156 新增角色 /console/api/bws/role juese 正常
2022-04-27 16:08:46 2022-04-27 16:08:46 secadmin 192.168.29.156 删除用户 /console/api/bws/user user 正常
2022-04-27 16:04:30 2022-04-27 16:04:30 secadmin 192.168.29.156 解锁用户 /console/api/bws/user/unlock user 正常
2022-04-27 16:03:49 2022-04-27 16:03:49 secadmin 192.168.29.156 更新用户 /console/api/bws/user user 正常
2022-04-27 16:00:14 2022-04-27 16:00:14 secadmin 192.168.29.156 更新用户 /console/api/bws/user/role user 正常
2022-04-27 16:00:10 2022-04-27 16:00:10 secadmin 192.168.29.156 更新用户 /console/api/bws/user/role user 异常
1 2 3
10条/页

图 4-73 操作审计

第5章 补丁管理

5.1 关于补丁包

当BES WebServer发布新的特性时，有时会以补丁包的形式发布，BES WebServer提供了补丁包的安装工具 Patch，将补丁包安装到BES WebServer上。

5.2 命名规则

BES WebServer补丁包都以zip文件形式存在,并符合固定的命名规则,如BWS3.1.0.9999.001.zip。

补丁的命名规则与补丁包基本一致,但以jar文件形式存在,并以V开头,如V3.1.0.9999.001.jar。

具体命名规则如下:

表 5-1 补丁命名规则

版本号	说明	示例
第一位版本号	代表软件更新换代。	如V**3**.1.0.9999
第二位版本号	软件有计划的大的功能升级。	如V3.**1**.0.9999
第三位版本号	软件有计划的小的功能升级。	如V3.1.**0**.9999
第四位版本号	发布的ID编号。	如V3.1.0.**9999**
在版本号后加小版本加T	临时紧急客户补丁编号。	如V3.1.0.9999.001.**T001**
在版本号后加小版本	正式紧急客户补丁编号。	如V3.1.0.9999.**001**
在版本号后加P	累计补丁。	如V3.1.0.9999.**P1**

5.3 PATCH命令

在BES WebServer的bin目录下，运行patch命令查看帮助信息:

```
[bes@Linux17209 bin]$ ./patch

Usage: patch -jar file[:file...]      多个jar文件之间使用操作系统的系统分隔符分隔
    or
        patch -path path[:path...]    多个指定的目录下的文件一起打补丁,
        ↪  多个目录之间使用操作系统的系统分隔符分隔
    or
        patch -revert version-name    还原补丁
    or
        patch -list                   列出所有的补丁
```

（1）补丁进程如果被意外终止，如：断电、内存溢出等情况，下次运行patch命令时程序会自动检测备份信息，并回退掉上次安装过程被异常中断的补丁。

（2）如果执行patch命令安装补丁过程中，需要安装的补丁包的优先级小于最大已安装的补丁优先级，则该补丁将被过滤而不会被安装。如:

```
[bes@Linux17209 bin]$ ./patch  -jar BWS3.1.0.9999.001.zip
Patch /home/bes/bws/BWS3.1.0.9999.001.zip is applied successfully.
Upgrade successfully, please check directory /home/bes/bws/patch.

再次打相同补丁:
[bes@Linux17209 bin]$ ./patch  -jar BWS3.1.0.9999.001.zip
Higher patches may exist or BWS3.1.0.9999.001.zip has been applied already, w
↪  hich will be ignore.
```

5.3.1 补丁种类

补丁分以下几种:

1) 临时紧急客户补丁：如V3.1.0.9999.001.T001。

2) 正式紧急客户补丁：如V3.1.0.9999.001。

3) 累计补丁：如V3.1.0.9999.P1。

4) 基于累计补丁的临时紧急客户补丁：如V3.1.0.9999.P1.001.T001。

5) 基于累计补丁的正式紧急客户补丁：如V3.1.0.9999.P1.001。

5.3.2 过滤和排序

未来可能会出现很多临时和正式补丁，这些补丁并不需要用户关心和整理，补丁工具会自行判断哪些补丁应该生效。

具体过滤和排序机制如下：(示例中 <= 表示共存但有序 » 表示过滤)

1) 基于同一个正式版本的多个补丁，只有一个补丁生效。

2) 生效序列为：正式补丁 » 高版本临时补丁 » 低版本临时补丁。

如：Vxxx.001 » Vxxx.001.**T002** » Vxxx.001.**T001**

3) 基于不同正式版本的非累计补丁，不互相影响，但要按照正式版本由小到大的顺序应用补丁。

如：Vxxx.**003**.T001 <= Vxxx.**002**.T002 <= Vxxx.**001**

4) 存在累计补丁，只有基于最大的积累版本号的补丁生效。

如：Vxxx.**P2**.002 <= Vxxx.**P2**.001.T001 <= Vxxx.**P2** » Vxxx.**P1** » Vxxx.001

5) 基于累计补丁的临时紧急客户补丁和基于累计补丁的正式紧急客户补丁的关系与普通临时补丁和普通正式补丁的关系相同。

如：Vxxx.P2.**003**.T001 <= Vxxx.P2.**002**.T002 <= Vxxx.P2.**001** » Vxxx.P2.001.**T002** » Vxxx.P2.001.**T001**

第6章 产品优化

BES WebServer使用hash表来协助完成请求的快速处理。

考虑到保存键及其值的hash表存储单元的大小不至于超出设定参数(hash bucket size)，在启动和每次重新配置时，BES WebServer为hash表选择尽可能小的尺寸。直到hash表超过参数(hash max size)的大小才重新进行选择，对于大多数hash表都有指令来修改这些参数。

示例：

保存服务器名字的hash表是由指令 server_names_hash_max_size和

server_names_hash_bucket_size所控制的。参数hash bucket size总是等于hash表的大小，并且是一路处理器缓存大小的倍数。在减少了在内存中的存取次数后，使在处理器中加速查找hash表键值成为可能。如果hash bucket size等于一路处理器缓存的大小，那么在查找键的时候，最坏的情况下在内存中查找的次数为2。第一次是确定存储单元的地址，第二次是在存储单元中查找键值。因此，如果BES WebServer给出需要增大 hash max size 或 hash bucket size的提示，那么首要的是增大前一个参数的大小。

6.1 进程优化

6.1.1 设置进程运行模式

BES WebServer有single process和master-worker两种运行模式，推荐（也是默认）使用后者，即推荐配置daemon on以及 master_process on（以单master+多worker模式运行)。这两个指令缺省值皆为on，也可不配置直接使用它们的缺省值。

6.1.2 设置worker进程数目

worker_processes应配置为不低于CPU核数，推荐配置为CPU核数的1~2倍（这是因为BES WebServer有worker进程级别的负载均衡机制，当某个worker进程负载连接数达到一定阈值，会让其停止接受新的连接），通过“getconf _NPROCESSORS_ONLN”或“nproc”获取CPU核数。

6.1.3 设置worker进程优先级

Linux系统下，可通过worker_priority指令配置worker进程调度优先级，可选数值为-20 ~ 19，数值越低，级别越高。

实际应用中，应配置为一个合适的数值，而不是越低越好。配置的数值过低，可能导致其他的对时延敏感的进程（例如UI交互程序）得不到及时响应，而令使用者感到明显卡顿。

6.1.4 设置worker进程CPU亲缘性

可通过worker_cpu_affinity指令设置各worker进程的CPU亲缘性。worker进程绑定到某个固定CPU核心运行，可以减少因进程切换导致的CPU高速缓存失效，从而提升程序运行性能。

worker_cpu_affinity为掩码形式的配置，掩码长度需要与worker_processes设置值一致。例如，配置worker_processes 4，则可以相应配置worker_cpu_affinity 0001 0010 0100 1000。

6.2 网络优化

6.2.1 TCP连接优化

利用server指令块下的listen指令的一些可选参数，可以控制或优化TCP连接行为。可选参数如下：

fastopen=number

配置为大于0的整数表示打开此选项。此选项开启TCP的TFO（TCP Fast Open）特性。

TFO原理：当client首次访问server，在进行TCP三次握手过程中，发送SYNC包，server根据客户端IP等信息生成cookie，经加密后与sync-ack一同发送给client，client再发送ACK完成三次握手；client在随后新一轮的与server的三次握手过程中，client可以直接发送SYNC+cookie+data，若server校验合法，则回复ACK+数据，否则回归到常规的TCP三次握手过程。可见TFO是对TCP三次握手的优化。

欲使用TFO特性，请确保系统内核支持TCP_FASTOPEN socket选项（Linux下可以通过是否存在/proc/sys/net/ipv4/tcp_fastopen节点确认）。与此相关的两个内核参数数值及含义如下表。

选项	说明
net.ipv4.tcp_fastopen	0 关闭客户端、服务端角色的TFO；1 开启客户端角色TFO；2 开启服务端角色TFO；3 开启客户端、服务端角色TFO。
net.core.somaxconn	影响TFO队列大小，也即fastopen=xxx配置实际生效的是其与net.core.somaxconn配置项的较小值。

backlog=number

设置TCP半连接队列，也即SYN_RECV队列的大小。Linux下此选项作为listen系统调用的第二个参数。

关于TCP半连接、全连接队列以及TCP三次握手过程，Linux有几个相关内核参数：

选项	说明
net.ipv4.tcp_max_syn_backlog	SYN_RECV队列大小。
net.ipv4.tcp_syncookies	可用来设置SYN_RECV队列满时的TCP三次握手处理策略。可配置值：0 关闭；1 SYN_RECV队列溢出时再开启；2 无条件开启。
net.core.somaxconn	TCP全连接队列大小。
net.ipv4.tcp_abort_on_overflow	控制accept队列，即全连接队列溢出时的行为。可配置值：0 丢弃客户端ACK报文；1 发送RST报文关闭连接。

TCP syncookies原理：服务端根据当前状态计算一个cookie，放在SYNC+ACK报文中。收到客户端ACK后，取出此cookie校验，如果合法则连接建立成功。可见纵使SYN_RECV队列已满，仍然可以通过TCP syncookies机制确保TCP连接可以正常建立。

需要注意的是,设置SYN_RECV队列大小,需要同时设置backlog、net.ipv4.tcp_max_syn_backlog以及net.core.somaxconn才能生效;设置全队列大小,需要同时设置net.ipv4.tcp_max_syn_backlog和net.core.somaxconn才能生效。

rcvbuf=size sndbuf=size

设置TCP接收/发送内存大小。Linux下设置时，需要配合内核参使用：

选项	说明
net.ipv4.tcp_rmem	TCP读缓存的最小值、默认值及最大值，单位字节
net.ipv4.tcp_wmem	TCP写缓存的最小值、默认值及最大值，单位字节
net.ipv4.tcp_mem	系统无内存压力、启动压力模式内存阈值、最大值，单位页大小

deferred

延迟处理新连接，即当客户端数据到达时再处理连接。这样可以降低BES WebServer保有的句柄数，从而提高处理性能。

reuseport

该选项允许不同进程或线程侦听同一端口，可在内核层面实现进程或线程级别的负载均衡，并可以解决“惊群”问题。

so_keepalive=on|off

配置TCP协议级别的心跳探测功能。

6.2.2 设置TCP连接池大小

需要根据具体业务场景，通过worker_connections指令设置足够大（根据实际业务场景）的BES WebServer连接池大小。连接池大小为BES WebServer到上游、下游连接的总和。

注意,应配置worker_rlimit_nofile为大于worker_connections的数值,才能让worker_connections配置的数值生效。

6.2.3 控制小报文发送

为了增加网络带宽利用率，需要禁止小报文发送，相关配置如下：

指令	说明
tcp_nodelay	off 启用Nagle算法；on 禁用Nagle算法。
postpone_output	可配置为一个整数值。BES WebServer进程级别的小报文传输控制。
tcp_nopush	开启或关闭CORK算法。仅对senfile on开启时有效。

6.2.4 设置TCP延迟关闭

当有模块销毁HTTP连接，同时该连接又有HTTP请求报文未接受完毕时，配置指令以实现延迟关闭功能。指令如下：

选项	说明
lingering_close	可选值：off 不使能延迟关闭功能；on 根据HTTP请求报文是否接收完毕来决定是否延迟关闭；always 无条件开启。
lingering_time	配置为时间单位，即“整数+[y|M|w|d|h|m|s]”，分别表示年、月、周、日、时、分、秒。此配置含义是延迟关闭开启时，读取HTTP请求报文的超时时间。
lingering_timeout	单位同上。此配置含义是检测HTTP请求报文到达的超时时间。

6.2.5 控制关闭超时TCP连接的方式

当TCP连接有读写事件发生超时，默认情况下，BES WebServer与对端进行四次挥手过程以关闭连接。通过如下指令改变这一行为，即关闭连接时向对端发送RST包代替四次挥手。

指令	说明
reset_timedout_connection	on 开启此功能；off 关闭此功能。

6.3 SSL/TLS连接优化

6.3.1 使用SSL/TLS session缓存

BES WebServer基于openssl实现SSL/TLS功能，支持SSL/TLS session缓存。指令如下：

指令	说明
ssl_session_cache off|none|[builtin[:size]]|[shared:name:size]	off 明确禁用session缓存；none session缓存功能开启，但实际并不缓存session；builtin 使用openssl内置缓存，此情况下仅当客户端命中到同一个worker进程，session缓存才生效；shared:name:size 使用BES WebServer应用层面共享内存中的session缓存，此情况下客户端命中任意worker进程，session缓存都生效。

6.3.2 使用SSL/TLS会话票据

SSL/TLS握手成功后，BES WebServer可以将session加密后作为tickets发给客户端。客户端下次发起连接时，将tickets发给BES WebServer，BES WebServer验证通过后即可复用此session。相关指令如下：

指令	说明
ssl_session_tickets	on 开启SSL/TLS会话票据功能；off 关闭SSL/TLS会话票据功能
ssl_session_ticket_key	配置生成SSL/TLS会话票据的秘钥文件。该指令可以重复使用，即配置多个秘钥文件。

6.4 IO优化

6.4.1 使用direct IO

direct IO即直接IO，一般在BES WebServer作为静态资源server场景下使用。直接IO可以避免系统内核级别的缓存拷贝。当文件较大且不希望生成文件缓存时，可以考虑使用直接IO。相关指令如下：

指令	说明
directio	off 关闭direct IO；字节单位，即“数字+[k|K|m|M|g|G]”，分别表示Byte（没有单位字母后缀）、KB、MB、GB。表示文件大小超过该设定值，再开启direct IO。
directio_alignment	字节单位，同上。跟BES WebServer所在系统使用的文件系统有关，一般配置成512。当使用XFS文件系统时，配置成4K。

6.4.2 使用AIO

AIO即异步IO，顾名思义，表示异步读写操作，一般在BES WebServer作为静态资源server场景下使用。AIO一般搭配directio指令或线程池使用。相关指令如下：

指令	功能
aio	on 开启AIO；off 关闭AIO；threads[=pool] 搭配线程池使用，pool参数省略时，使用“default”名字的线程池。
aio_write	写文件时是否开启AIO。仅在搭配线程池使用时生效。on | off分别表示功能开启、关闭。

6.4.3 使用零拷贝

零拷贝，即消除BES WebServer应用空间buffer到系统内核socket buffer、系统内核socket buffer到网卡驱动buffer的拷贝。一般在BES WebServer作为静态资源server场景下使用。相关指令如下：

指令	说明
sendfile	on 开启零拷贝；off 关闭零拷贝。

若directio、aio、senfile同时开启，如何生效呢？如下例子：

```
location /static_resource{
   sendfile on;
   aio on;
   directio 10m;
}
```

上述配置表示：当文件大小超过10MB时，aio+ directio生效；否则sendfile生效。

6.4.4 压缩HTTP输出报文

可以通过压缩BES WebServer本身或来自上游server的HTTP response报文，来提升网络传输效率。BES WebServer提供gzip压缩方法。相关指令如下：

指令	说明
gzip	可选值on|off，表示使能或禁用gzip压缩功能。
gzip_buffers	语法：gzip_buffers number size。压缩功能使用的buffer数量和大小。一般buffer大小配置成内存页框大小。
gzip_comp_level	压缩比率，可配置成1~9的数字。
gzip_types	对那些MIME类型进行压缩，可配置多个参数。
gzip_window	LZ77算法使用的压缩窗口大小，可选值512、1k、2k、4k、8k、16k、32k。
gzip_hash	压缩算法使用的哈希表元素个数，可选值512、1k、2k、4k、8k、16k、32k, 64k、128k。
postpone_gzipping	设置延迟压缩阈值，即缓存数据大小大于此数值才进行压缩。可配置成为“数字+[k|K|m|M|g|G]”，分别表示Byte（数字后无后缀）、Kbyte、Mbyte、GByte。
gzip_no_buffer	可选值on|off，决定是否对每次压缩的数据不经缓存直接输出。
gzip_min_length	HTTP报文长度大于该设置定才进行压缩。可配置成为“数字+[k|K|m|M|g|G]”，分别表示Byte（数字后无后缀）、Kbyte、Mbyte、GByte。

6.5 超时时间优化

BES WebServer对一些关键操作，比如TCP连接建立、读写事件设定了超时机制。合理配置超时数值，可以减少错误发生。相关指令如下：

指令	说明
worker_shutdown_timeout	worker进程关闭超时时间。
client_header_timeout	读取客户端HTTP请求头的超时时间。
client_body_timeout	两次HTTP请求体读操作间的超时时间。若超过该设定值，向对端返回HTTP 408错误。

指令	说明
send_timeout	两次写操作的超时时间。若超过该设定值没有发生写操作，则连接被关闭。
proxy_connect_timeout	BWS向上游服务建立连接的超时时间。
proxy_send_timeout	与上游间的两次写操作的超时时间。
proxy_read_timeout	与上游间的两次读操作的超时时间。
ssl_session_timeout	SSL/TLS session缓存过期失效时间

第7章 使用场景

7.1 静态内容服务

BES WebServer服务器的一个重要功能是提供文件（比如图片或者静态 HTML 页面）服务。以下我们通过一个示例的方式，根据请求不同目录的文件，来提供服务：/data/www （可能包含 HTML 文件）和 /data/images （包含图片）。

（1）准备静态文件：

创建/data/html（包含HTML 文件）和 /data/images （包含图片）。

（2）在管理控制台–配置管理–虚拟主机，新建虚拟主机，代理类型选择反向代理，端口号为8090，代理目标中配置如下：

添加代理目标1，用于访问HTML文件：

图 7-1 代理目标1配置

添加代理目标2，用于访问图片：

图 7-2 代理目标2配置

（3）启动实例。

访问HTML页面：http://localhost:8090/test.html

访问图片：http://localhost:8090/test.png

7.2 反向代理

BES WebServer可以作为反向代理服务器，接收客户端请求，然后选择一个实际的被代理服务器，转发这个请求，获取到响应后再返回给这个客户端。实现后端多个真实的服务器不直接被外部所访问，而是对用户呈现代理服务器地址。在此示例中，后端服务器使用BES应用服务器搭建环境，并部署简单应用cluster.war展示测试效果。如使用其他服务器请自行搭建环境。

（1）在管理控制台–配置管理–虚拟主机，新建虚拟主机，代理类型为反向代理，代理目标中基本信息的允许访问资源为`/cluster`，代理类型选择动态`HTTP`，代理地址为`http://192.168.29.132:18080`

（2）在客户端访问cluster.war，访问成功。

7.3 正向代理

BES WebServer 的一个常见用途是作为一个代理服务器，作用是接收请求并转发给被代理的服务器，从中取得响应，并将其发送回客户端。在此示例中，服务器使用BES应用服务器搭建环境，如使用其他服务器请自行搭建环境。

前置步骤：在BES应用服务器环境上创建一个实例作为后端服务器处理请求，修改实例虚拟主机配置使得该实例拒绝客户端所在IP访问，然后部署简单应用cluster.war展示访问效果。

（1）配置管理–虚拟主机，新建虚拟主机，代理类型选择正向代理，DNS解析地址填写127.0.0.1，监听端口为8082，代理目标中添加如下内容：

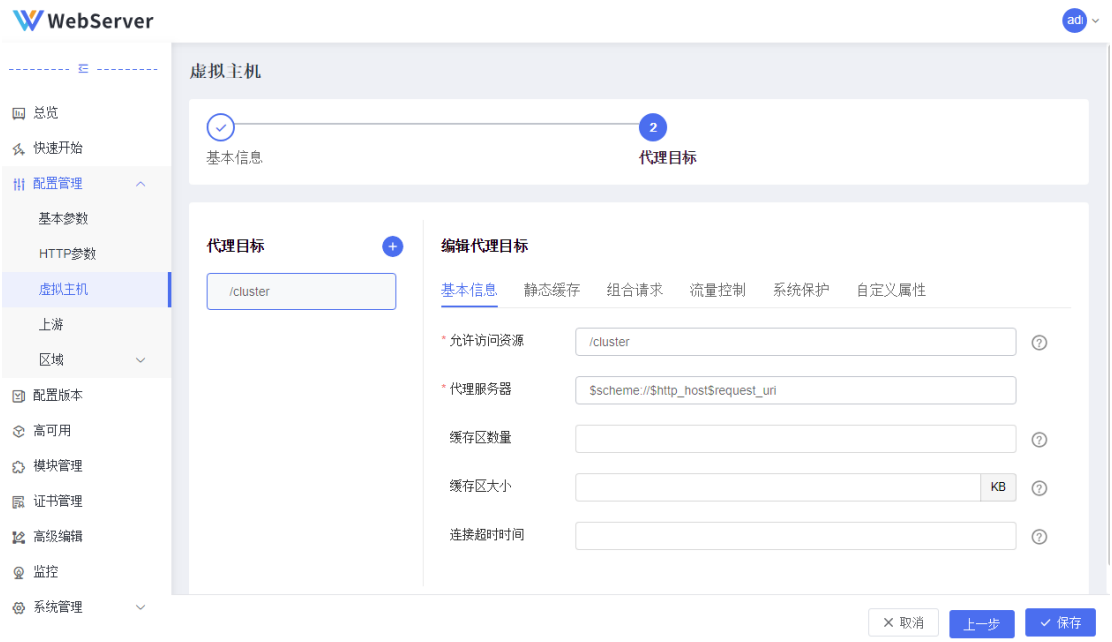


图 7-3 正向代理代理目标配置

（2）启动或重启实例，在客户端直接访问cluster.war，访问被拒绝。

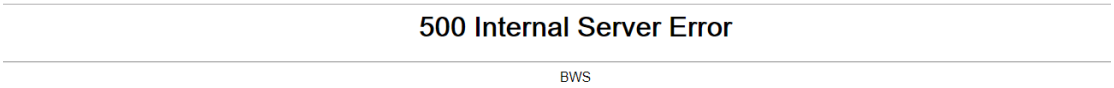


图 7-4 请求拒绝页面

（3）在客户端使用代理访问cluster.war，访问成功。

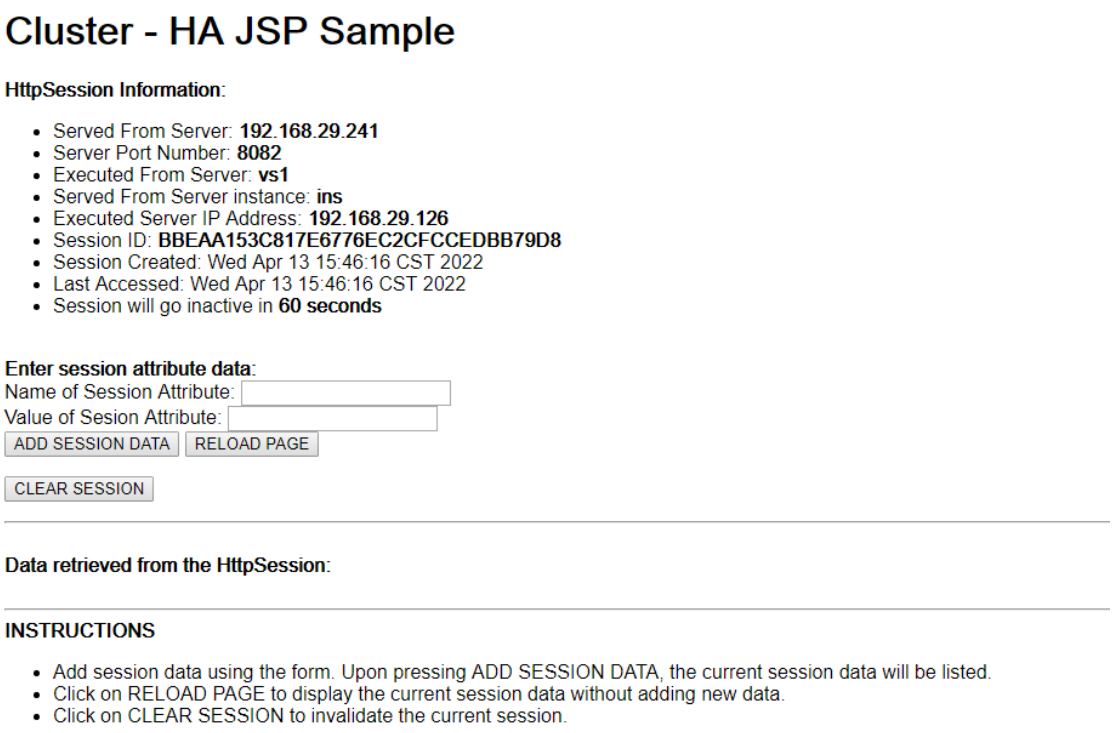


图 7-5 负载均衡测试页面效果.png

7.4 负载均衡

BES WebServer可以创建bws实例向多个后端服务器转发请求、分配工作负载，从而提高系统的整体吞吐量。在此示例中，后端服务器使用BES应用服务器搭建环境，在其环境上创建两个实例作为两个后端服务器处理请求，并部署简单应用cluster.war展示测试效果。如使用其他服务器请自行搭建环境。

图 7-6 负载均衡示意图

（1）在管理控制台–配置管理–上游，新建上游，添加2个后端目标，后端目标的IP端口对应BES实例的监听地址及HTTP端口，如下所示：

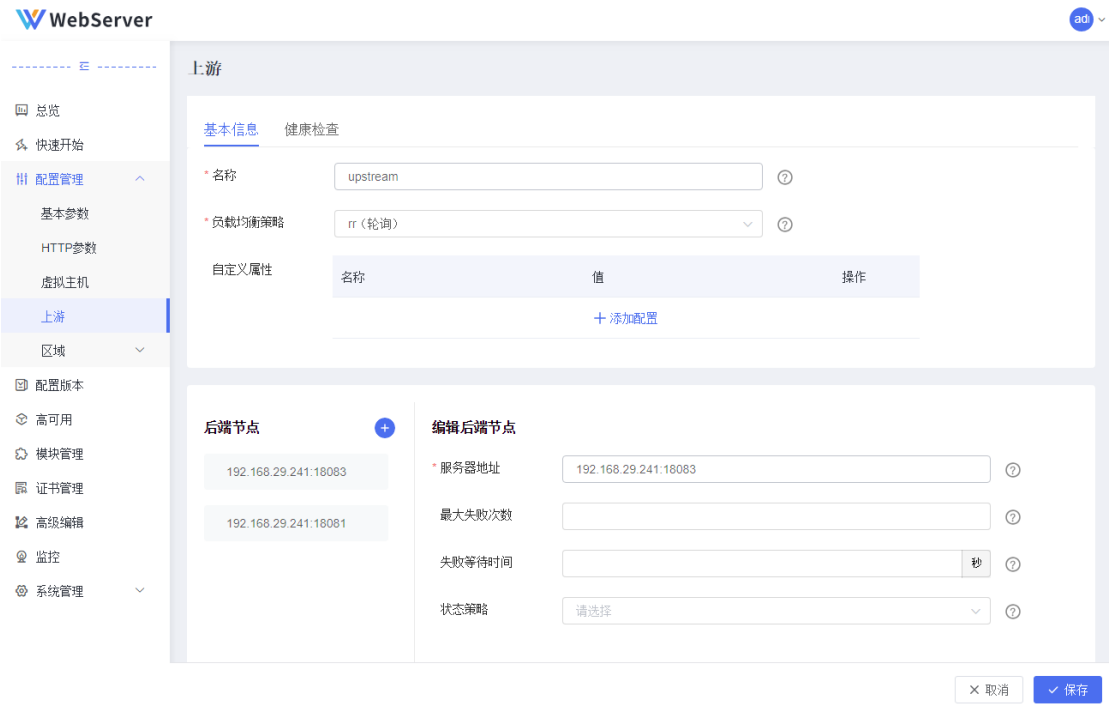


图 7-7 上游后端目标配置

负载均衡策略：

1. rr（轮询）：通过逐一轮询所配置的server列表，将接收到的请求平均的分配到后端服务器。

2. 权重：在后端服务器资源不一样、硬件差别较大的情况下，可以在轮询策略的基础上指定轮询的几率，weight的数值与访问比率成正比，权重越大分配到需要处理的请求越多。
3. ip_hash：每个请求按照发起客户端的IP的哈希结果进行匹配，这样的算法下，一个固定的IP地址的客户端总会访问到同一个后端服务器，在一定程度上，解决了集群环境下session共享问题。
4. url_hash：按照访问的url的哈希结果分配请求，每个请求的url会指向后端固定的服务器，可以在bws作为静态服务器的情况下提高缓存效率。
5. least_conn：该算法自动识别后端服务中，哪些后端服务器占用了较多连接，从而将更多的请求分配到较为空闲且权重较大的后端服务器中。
6. fair：动态的根据后端服务器的请求处理到相应的时间进行均衡分配，响应时间短，处理 效率高的后端服务器分配到请求的概览高，响应时间长，处理效率低的后端服务器分配 到的请求少。
7. sticky：通过cookie实现客户端与后端服务器的会话保持，在一定条件下可以保证同一个 客户端访问的都是同一个后端服务器。

（2）在管理控制台–配置管理–虚拟主机，新建虚拟主机，代理类型为反向代理，代理目标中添加如下内容：

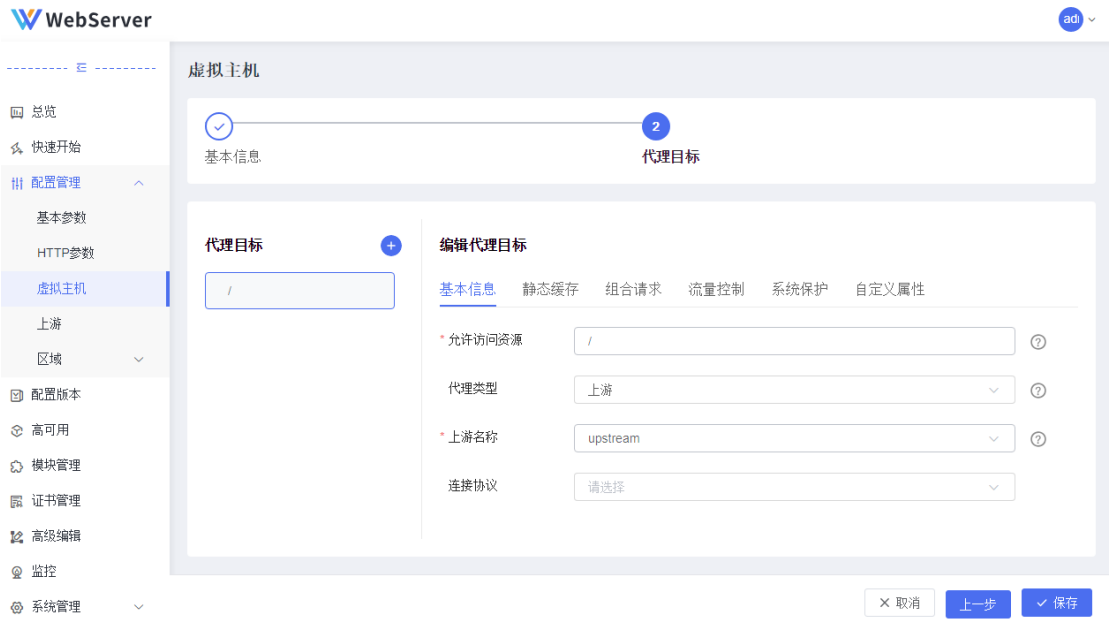
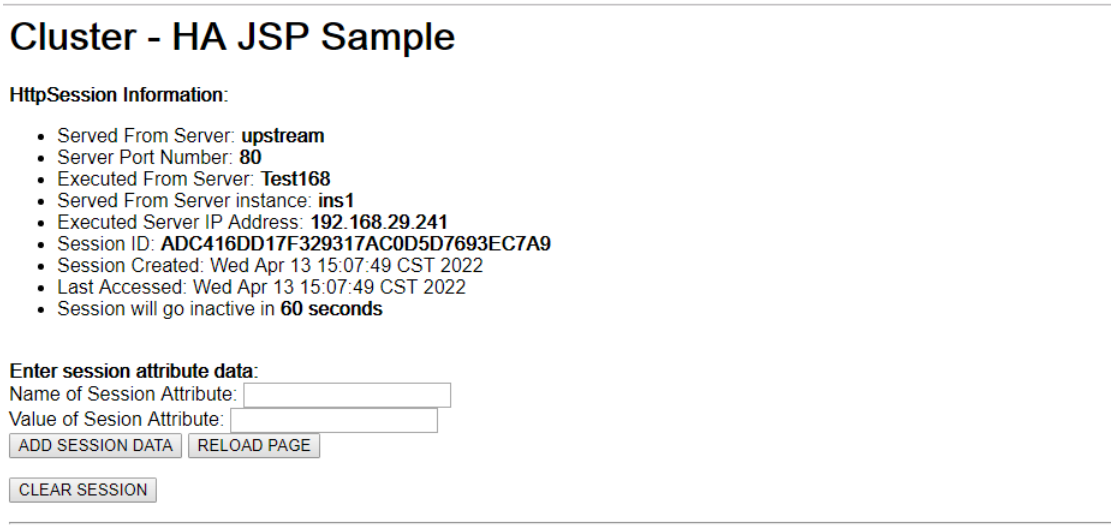


图 7-8 代理目标配置

（3）启动或重启实例，多次访问页面，Served From Server instance后显示的处理请求的服务器在BES的两个实例之间不断切换。



Cluster - HA JSP Sample

HttpSession Information:

- Served From Server: **upstream**
- Server Port Number: **80**
- Executed From Server: **Test168**
- Served From Server instance: **ins1**
- Executed Server IP Address: **192.168.29.241**
- Session ID: **ADC416DD17F329317AC0D5D7693EC7A9**
- Session Created: Wed Apr 13 15:07:49 CST 2022
- Last Accessed: Wed Apr 13 15:07:49 CST 2022
- Session will go inactive in **60 seconds**

Enter session attribute data:
Name of Session Attribute:
Value of Sesion Attribute:
ADD SESSION DATA RELOAD PAGE

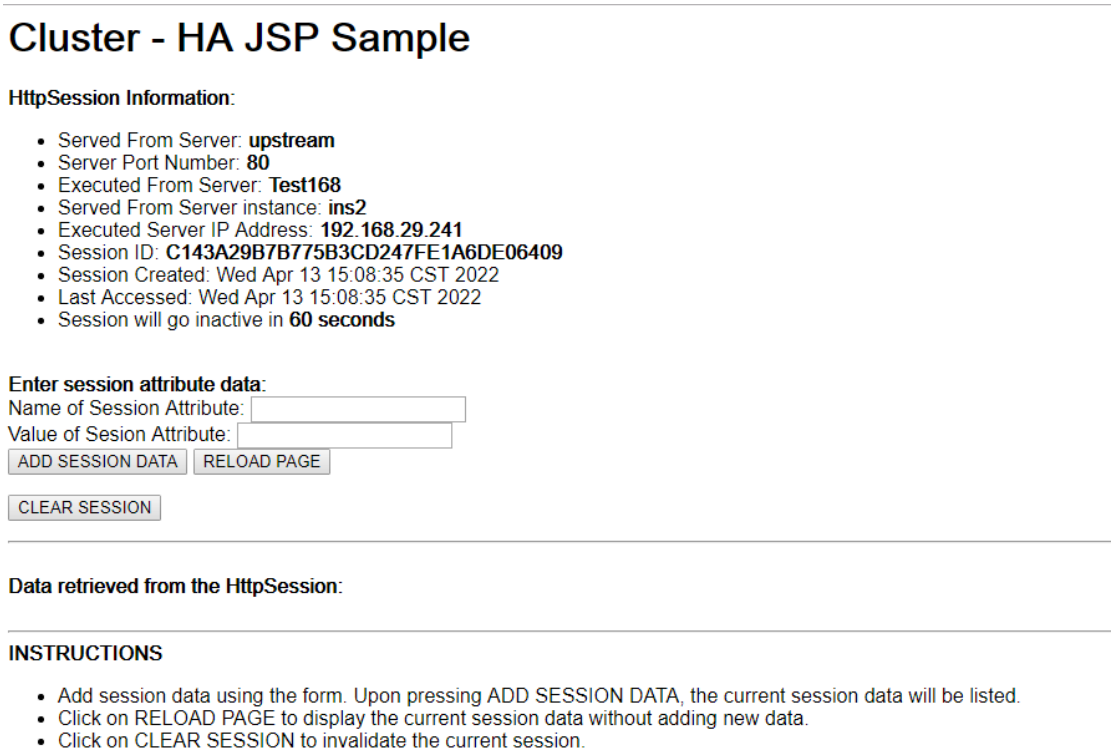
CLEAR SESSION

Data retrieved from the HttpSession:

INSTRUCTIONS

- Add session data using the form. Upon pressing ADD SESSION DATA, the current session data will be listed.
- Click on RELOAD PAGE to display the current session data without adding new data.
- Click on CLEAR SESSION to invalidate the current session.

图 7-9 负载均衡测试页面效果1



Cluster - HA JSP Sample

HttpSession Information:

- Served From Server: **upstream**
- Server Port Number: **80**
- Executed From Server: **Test168**
- Served From Server instance: **ins2**
- Executed Server IP Address: **192.168.29.241**
- Session ID: **C143A29B7B775B3CD247FE1A6DE06409**
- Session Created: Wed Apr 13 15:08:35 CST 2022
- Last Accessed: Wed Apr 13 15:08:35 CST 2022
- Session will go inactive in **60 seconds**

Enter session attribute data:
Name of Session Attribute:
Value of Sesion Attribute:
ADD SESSION DATA RELOAD PAGE

CLEAR SESSION

Data retrieved from the HttpSession:

INSTRUCTIONS

- Add session data using the form. Upon pressing ADD SESSION DATA, the current session data will be listed.
- Click on RELOAD PAGE to display the current session data without adding new data.
- Click on CLEAR SESSION to invalidate the current session.

图 7-10 负载均衡测试页面效果2

用户也可以使用权重方式来实现加权负载均衡。

7.5 PHP站点配置

BES WebServer 可以使用 location 来处理典型的简单 PHP 站点的请求：

（1）确保php-fpm已经启动

```
[BWS@bogon sbin]$ ps aux | grep php-fpm
BWS       11326  0.0  0.1 207944  4036 ?        Ss   14:25   0:00 php-fpm: ma
↪  ster process (/usr/local/php/etc/php-fpm.conf)
BWS       11327  0.0  0.1 207944  3336 ?        S    14:25   0:00 php-fpm: po
↪  ol www
BWS       11328  0.0  0.1 207944  3336 ?        S    14:25   0:00 php-fpm: po
↪  ol www
BWS       11351  0.0  0.0 112728   992 pts/1    R+   14:25   0:00 grep --colo
↪  r=auto php-fpm
```

如果没有启动，则启动php-fpm：

```
/usr/local/php/sbin/php-fpm
```

（2）在实例的反向代理–代理目标中添加如下内容：

图 7-11 PHP配置

自定义属性中添加如下内容：

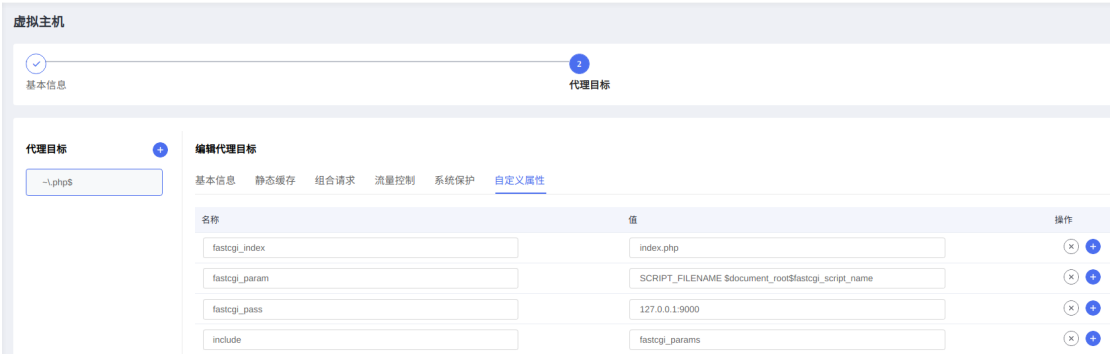


图 7-12 PHP配置

（3）在BES WebServer的html目录中添加index.php文件，内容如下：

```
<?php phpinfo();?>
```

（4）重启实例，访问页面，出现PHP的信息，即代表访问成功！

注意：如果访问时出现：“Primary script unknown”while reading response header from upstream异常，原因是bws.conf和php-fpm.d/www.conf中的user不一致，需要设置成同样的用户。

7.6 支持国密

（1）用户需要自行准备好国密证书或者使用BWS_HOME/conf/security中的证书。

（2）在管理中心左侧导航区点击“**证书管理**”，新建签名证书、加密证书和普通证书，分别对应BWS_HOME/conf/security中：

```
签名证书signCer：server-sign.crt和server-sign.key
加密证书encCer：server-enc.crt和server-enc.key
普通证书normalCer：server.crt和server.key
```

（3）在管理中心左侧导航区点击“**虚拟主机**“，在默认的8090虚拟主机中，启用SSL状态–选择证书，选择刚才新建的三组证书signCer、encCer和normalCer。

（4）在管理中心左侧导航区点击“**虚拟主机**“，默认的8090虚拟主机的自定义属性中添加如下内容：

```
    ssl_protocols TLSv1.2 TLSv1.3; #同时支持tls和gmssl，自适应
    ssl_session_cache shared:SSL:1m;
    ssl_session_timeout 5m;
    ssl_ciphers "ECDHE-RSA-AES256-GCM-SHA384:ECC-SM2-WITH-SM4-SM3:ECDHE-SM2-WI
    ↪  TH-SM4-SM3";
    ssl_prefer_server_ciphers on;
```

（5）启动BWS实例，使用支持国密的浏览器访问即可。

7.7 URL重写

BES WebServer支持对URL进行更改请求uri、返回重定向，详情请参考rewrite模块，可以在高级编辑中添加如下配置：

```
server {
    listen 8090;
    server_name localhost;
    location / {
        rewrite ^(/bes/)(.*) /test/$2 last;
    }
    location /test {
        proxy_pass http://192.168.29.132:8080;
    }
}
```

当访问/bes/开头的url时，将url重写为/test/开头，并重新搜索location来匹配重写后的url。

7.8 WebSocket配置

BES WebServer支持代理WebSocket协议配置，在此示例中，后端服务器使用BES应用服务器搭建环境，并部署WebSocket.war（需修改此应用中IP端口和地址为BWS虚拟主机的IP和端口）。如使用其他服务器请自行搭建环境。

（1）在管理控制台–配置管理–虚拟主机，新建虚拟主机，代理类型为反向代理，代理目标中基本信息的允许访问资源为/WebSocket，代理类型选择动态HTTP，代理地址为http://192.168.29.132:8080，自定义属性添加proxy_set_header Upgrade $http_upgrade以及proxy_set_header Connection "upgrade"。

配置如下：

```
server {
    listen 8090;
    server_name localhost;
    location /WebSocket {
        proxy_pass http://192.168.29.132:8080;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
    }
}
```

7.9 Lua配置

BES WebServer支持用户使用Lua脚本实现自定义的业务逻辑，保证高并发服务能力的同时，极大的降低用户业务逻辑的实现成本。如在高级编辑中添加以下配置：

```
location /lua {
    default_type text/html;
```

```
    content_by_lua_block{
        ngx.say("hello world");
    }
}
```

7.10 反向代理黑白名单

BES WebServer支持定义黑白名单，允许限制特定IP、资源等访问。如在高级编辑中添加以下配置：

```
server {
    allow 192.168.31.1/27;  #允许31.1~31.30范围内的IP访问
    deny all;   #其他IP全禁止

    # 禁止访问后缀名为html的文件
    location ~* \.(html)$ {
        deny all;
    }

    # 若访问不是GET请求，则返回403
    if ($request_method !~ ^(GET)$){
        return 403;
    }
}
```

7.11 gzip压缩

BES WebServer支持使用gzip方法压缩响应，有助于将传输数据的大小减少一半甚至更多，减少网络传输带宽使用。在管理中心左侧导航区点击“**HTTP参数**“，启用gzip压缩，压缩比为1，最小压缩文件为20，详情请参考gzip模块。配置如下：

```
gzip on;
gzip_comp_level 1;
gzip_min_length 20k;
```

7.12 HTTPS配置

BES WebServer支持HTTPS配置。

（1）用户需要自行准备好证书或者使用BWS_HOME/conf/security中的证书。

（2）在管理中心左侧导航区点击“**证书管理**”，新建证书，使用BWS_HOME/conf/security中的server.crt和server.key。

（3）在管理中心左侧导航区点击“**虚拟主机**“，在默认的8090虚拟主机中，启用SSL状态–选择证书，选择刚才新建的证书。

使用https访问即可。

7.13 TCP或UDP负载均衡配置

BES WebServer支持四层协议负载均衡功能，该功能由stream模块和upstream模块实现，通过负载均衡算法将请求分发到不同的后端服务器上。负载均衡策略：

1. 轮询：默认负载均衡策略，配置文件中不需要设置
2. 随机：随机选取一个后端服务器或随机选取两个后端服务器，然后使用指定的方法进行配置
3. 哈希：按指定配置进行哈希选择后端服务器
4. 最小连接：将请求转发到后端连接数最小的后端服务器
5. 权重：设置后端服务器权重，性能高的后端服务器可适当提高权重，默认为 1

在高级编辑中添加如下配置，stream块与http块平级，若要使用udp协议，在listen之后需添加udp。

```
stream {
    upstream backend1 {
        server 192.168.29.132:18080;
        server 192.168.29.132:18081;
    }
    server {
        listen 8092;
        proxy_pass backend1;
    }
}
```

第8章 常见问题

8.1 静态文件404

访问静态文件出现404错误时，首先拿到对应文件的url，然后再对比文件在服务器存放的实际位置。当url文件路径与文件实际路径不符时，可以配置location来解决该问题，配置location可以理解为改变访问路径的指向。如分别访问/data/html/test1下的文件1.html和/data/html/test2下的文件2.html，可以配置以下两种方式来解决:

```
server {
    ...
    location /html/test1 {
        root /data;
        ...
    }

    location /html/test2 {
        alias /data/html/test2;
    }
    ...
}
```

root或alias都可以用来配置指向路径，只不过含义有点区别。

对于root，BES WebServer实际是将root路径 + 匹配的location + 匹配的location之后的路径，比如上述访问的1.html 文件，那么实际访问的路径是/data + /html/test1 + /1.html。

对于alias，BES WebServer实际是将alias路径 + 匹配的location之后的路径，比如上述访问的2.html 文件，那么实际访问的路径是/data/html/test2 + /2.html。

8.2 静态文件403

访问静态文件时出现403错误，通过ls -l BWS_HOME/html/index.html确认静态文件权限，工作进程需要对文件有读权限，为了安全考虑一般不使用777权限。以BWS_HOME/html目录为例修改权限。

```
find BWS_HOME/html -type d -exec chmod 755 {} \;
find BWS_HOME/html -type f -exec chmod 644 {} \;
```

8.3 Request Entity Too Large错误

当出现413（Request Entity Too Large）错误时，一般是上传的文件大小超过了BES WebServer的配置。可通过client_max_body_size指令定义最大大小，根据具体的业务需求设置。

```
http {
    client_max_body_size 8192k;
```

```
    ...
}
```

当使用PHP作为后端服务时，client_max_body_size的值应大于等于php.ini中的upload_max_filesize（允许上传文件大小的最大值）以及post_max_size（通过表单POST给PHP的所能接收的最大值，包括表单里的所有值）。

第9章 常用模块

9.1 主模块

控制 BES WebServer 的基本功能的指令。

9.1.1 daemon

语法:daemon on | off

缺省值:on

```
daemon  off;
```

决定 BES WebServer 是否应该成为守护进程，主要用于开发。

9.1.2 debug_points

语法:debug_points [stop | abort]

缺省值:none

```
debug_points stop;
```

该指令用于调试。当检测到内部错误时，例如，在重新启动工作流程时，发生套接字泄漏，启用 debug_points 会导致核心文件创建（abort ）或停止（stop ）进程以使用系统调试器进行进一步分析。

9.1.3 error_log

语法:error_log file [debug | info | notice | warn | error | crit]

缺省值:${prefix}/logs/error.log

```
error_log LOGFILE [ debug_core | debug_alloc | debug_mutex | debug_event]: | 」
↪  debug_http | debug_imap ;
```

配置日志。可以在同一级别指定几个日志。如果在 main 配置级别将日志写入一个未明确定义文件，则 将使用默认文件。

第一个参数定义了一个将存储日志的 file 。特殊值 stderr 选择标准错误文件。可以通过指定 syslog: 前 缀来配置记录日志到 syslog。可以通过指定 memory: 前缀和缓冲区 size 来配置记录日志到循环内存缓冲 区，此通常用于调试。

第二个参数确定日志记录的 level ，可以是以下之 一：debug 、info 、notice 、warn 、error 、crit 、alert 或 emerg 。上述日志级别按严重性递增排列。设置某个日志级别将造成所

有指定的消息和更严重的日志级别被记录。示例，默认级别 error 将导致 error 、crit 、alert 和 emerg 消息被记录。如果省略此参数，则使用 error 。

9.1.4 include

语法:include file

缺省值:none

你可以在任意地方使用include指令实现配置文件的包含，类似于apache中的include方法，可减少主配置文件。

include 指令还支持像下面配置一样的全局包含的方法，示例包含一个目录下所有以“.conf”结尾的文件:

```
include vhosts/*.conf;
```

9.1.5 lock_file

语法:lock_file file

缺省值:compile-time option

```
lock_file  /var/log/lock_file;
```

BES WebServer 使用锁机制来实现 accept_mutex 并序列化对共享内存的访问。大多数系统是使用原子操作实现锁，并忽略此指令。在其他系统上是使用“锁文件”机制。此指令用于指定一个锁文件名称的前缀。

9.1.6 master_process

语法:master_process on | off

缺省值:on

```
master_process  off;
```

用于决定是否启动工作进程。该指令适用于开发调试。

9.1.7 pid

语法:pid file

缺省值:compile-time option

示例:

```
pid BWS_HOME/logs/bws.pid;
```

进程id存储文件。可以使用 kill -HUP PID进行配置文件重新加载。

9.1.8 ssl_engine

语法:ssl_engine engine

缺省值:system dependent

该指令用于指定openssl使用的引擎。你可以通过下面的命令行获知系统目前支持的openssl引擎

```
openssl engine -t
```

9.1.9 timer_resolution

语法:timer_resolution t

缺省值:none

示例:

```
timer_resolution  100ms;
```

减少工作进程中的计时器分辨率，从而减少 gettimeofday() 的系统调用次数。默认情况下，每次接收到内核事件时都会调用 gettimeofday() 。随着分辨率的降低，gettimeofday() 仅在指定的时间间隔内被调用一次。

9.1.10 user

语法:user user [group]

缺省值:nobody nobody

指定BES WebServer Worker进程运行用户，默认是nobody帐号。

示例:

```
user www users;
```

9.1.11 worker_cpu_affinity

语法:worker_cpu_affinity cpumask [cpumask...]

缺省值:none

只适用于Linux平台。

将工作进程绑定到一组 CPU。每个 CPU 组由允许的 CPU 的位掩码表示。应为每个工作进程定义一个单独的组。默 认情况下，工作进程没有绑定任何特定的 CPU。

示例:

```
worker_proceses     4;
worker_cpu_affinity 0001 0010 0100 1000;
```

分别给每个worker进程绑定一个CPU.

```
worker_proceses     2;
worker_cpu_affinity 0101 1010;
```

将CPU0/CPU2绑定给第一个worker进程，将CPU1/CPU3绑定给第二个worker进程。

9.1.12 worker_priority

语法:worker_priority [-] number

缺省值:on

定义工作进程的调度优先级，与使用 nice 命令完成类似：负数意味着更高的优先级。允许范围通常在 -20 到 20 之间。

9.1.13 worker_processes

语法:worker_processes number

缺省值:none

示例:

```
worker_processes 5;
```

定义工作进程的数量。最优值取决于许多因素，包括（但不限于）CPU 核心数、存储数据的硬盘驱动器数量以及加载模式。当无法确定时，设置为可用 CPU 核心的数量是通常的做法。

9.1.14 worker_rlimit_core

语法:worker_rlimit_core size

缺省值: none

指定工作进程异常退出后生成core文件大小限制。

9.1.15 worker_rlimit_nofile

语法: worker_rlimit_nofile limit

缺省值: none

指定工作进程最大打开文件数（RLIMIT_NOFILE ）的限制。

9.1.16 working_directory

语法:working_directory path

缺省值: none

定义工作进程的当前工作目录，工作进程应有指定目录的写入权限。

9.2 事件模块

9.2.1 accept_mutex

语法:accept_mutex [on | off]

默认值:on

如果启用了 accept_mutex ，则工作进程将轮流接受新连接。否则，将新连接通知给所有工作进程，如果新连接数量较少，某些工作进程可能会浪费系统资源。

9.2.2 accept_mutex_delay

语法:accept_mutex_delay Nms;

默认值:500ms

如果启用了 accept_mutex，则指定如果另一个工作进程正在接受新连接，则工作进程将尝试重新启动以接受新连接 的最长时间。

9.2.3 debug_connection

语法:debug_connection [ip | CIDR]

默认值:none

启用所选客户端连接的调试日志。其他连接将使用由 error_log 指令设置的日志级别。调试连接由 IPv4 或 IPv6 地址或网络指定。也可以使用主机名指定连接。对于使用 UNIX 域套接字的连接，调试日志由 unix参数启用。

示例:

```
error_log BWS_HOME/logs/errors;
events {
debug_connection   192.168.1.1;
}
```

9.2.4 multi_accept

语法:multi_accept [on | off]

默认值:off

如果 multi_accept 被禁用，工作进程将一次接受一个新连接。否则，工作进程将一次接受所有新连接。

9.2.5 use

语法:use [kqueue | rtsig | epoll | /dev/poll | select | poll | eventport]

默认值:none

指定要使用的连接处理方式。通常不需要明确指定它，因为 BES WebServer 将默认使用最有效的方式。

9.2.6 worker_connections

语法:worker_connections number

默认值:10240

设置工作进程可以打开的同时连接的最大数量

通过worker_connections和worker_proceses可以计算出maxclients：

```
max_clients = worker_processes * worker_connections
```

作为反向代理，max_clients为：

```
max_clients = worker_processes * worker_connections/4
```

9.3 http核心模块

9.3.1 alias

语法:alias file-path|directory-path;

默认值:no

定义指定 location 的替换。示例，使用以下配置

```
location /i/ {
 alias /data/w3/images/;
 }
```

/i/top.gif 的请求，将发送 /data/w3/images/top.gif 文件。path 值可以包含变量，除 $document_root 和 $realpath_root 外。如果在使用正则表达式定义的 location 内使用了别名，那么这种正则表达式应该包含捕获，并且别名应该引用这 些捕获，示例：

```
location ~ ^/users/(.+\.(?:gif|jpe?g|png))$ {
   alias /data/w3/images/$1;
}
```

当 location 与指令值的最后一部分匹配时：

```
location /images/ {
   alias /data/w3/images/;
}
```

更好的方式是使用 root 指令：

```
location /images/ {
   root /data/w3;
 }
```

9.3.2 client_body_in_file_only

语法:client_body_in_file_only on|off

默认值:off

上下文:http, server, location

确定 BES WebServer 是否应将整个客户端请求体保存到文件中。可以在调试期间或使用 $request_body_file 变量或模块http_perl_module 的 $r->request_body_file 方法时使用此指令。当设置为 on 值时，临时文件在请求处理后不会被删除。值 clean 会将请求处理后剩下的临时文件删除。

9.3.3 client_body_in_single_buffer

语法:client_body_in_single_buffer

默认值:off

上下文:http, server, location

确定 BES WebServer 是否应将整个客户端请求体保存在单个缓冲区中。在使用 $request_body 变量时，建议使用该指令，用来保存涉及的复制操作次数。

9.3.4 client_body_buffer_size

语法:client_body_buffer_size the_size

默认值:8k/16k

上下文:http, server, location

设置读取客户端请求体的缓冲区大小。如果请求体大于缓冲区，则整个体或仅将其部分写入临时文件。默认情况下，缓冲区大小等于两个内存页。在 x86、其他 32 位平台和 x86-64 上是 8K。在其他 64 位平台上通常为 16K。

9.3.5 client_body_temp_path

语法:client_body_temp_path dir-path [level1 [level2 [level3]

默认值:client_body_temp

上下文:http, server, location

定义用于存储持有客户端请求主体的临时文件的目录。最多可以在指定目录下使用三级子目录层次结构。

示例:

```
client_body_temp_path /spool/bws/client_temp 1 2;
```

临时文件的路径可以如下:

```
/spool/bws/client_temp/7/45/00000123457
```

9.3.6 client_body_timeout

语法:client_body_timeout time

默认值:60

上下文:http, server, location

定义读取客户端请求正文的超时时间。超时设置仅是在两个连续读操作之间的时间间隔，而不是整个请求体的传输过 程。如果客户端在此时间内没有发送任何内容，则会将 408（请求超时）错误返回给客户端。

9.3.7 client_header_buffer_size

语法:client_header_buffer_size size

默认值:1k

上下文:http, server

设置读取客户端请求头的缓冲区大小。对于大多数请求，1K 字节的缓冲区就足够了。但是，如果请求中包含长 cookie，或者来自 WAP 客户端，则可能 1K 是不适用的。如果请求行或请求头域不适合此缓冲区，则会分配由 large_client_header_buffers 指令配置的较大缓冲区。

9.3.8 client_header_timeout

语法:client_header_timeout time

默认值:60

上下文:http, server

定义读取客户端请求头的超时时间。如果客户端在这段时间内没有传输整个报头，则将 408（请求超时）错误返回给 客户端。

9.3.9 client_max_body_size

语法:client_max_body_size size

默认值:client_max_body_size 1m

上下文:http, server, location

在 Content-Length 请求头域中指定客户端请求体的最大允许大小。如果请求的大小超过配置值，则将 413 （请 求实体过大）错误返回给客户端。请注意，浏览器无法正确显示此错误。将 size 设置为 0 将禁用检查客户端请 求正文大小。

9.3.10 default_type

语法:default_type MIME-type

默认值:default_type text/plain

上下文:http, server, location

定义响应的默认 MIME 类型。可以使用 types 指令对 MIME 类型的文件扩展名进行映射

示例:

```
location = /proxy.pac {
    default_type application/x-ns-proxy-autoconfig;
}
location = /wpad.dat {
    rewrite . /proxy.pac;
    default_type application/x-ns-proxy-autoconfig;
}
```

9.3.11 directio

语法:directio [size|off]

默认值:directio off

上下文:http, server, location

开启当读取大于或等于指定大小的文件时，使用 O_DIRECT 标志（FreeBSD、Linux）、F_NOCACHE 标志（macOS）或 directio() 函数（Solaris）。该指令自动禁用给定请求使用的 sendfile。它可以 用于服务大文件:

```
 directio  4m;
```

9.3.12 error_page

语法:error_page code [code...] [= | =answer-code] uri | @named_location

默认值:no

上下文:http, server, location, if in location

定义针对指定错误显示的 URI。

示例:

```
error_page   404          /404.html;
error_page   502 503 504  /50x.html;
error_page   403          http://example.com/forbidden.html;
error_page   404          = @fetch;
```

此外，可以使用 =response 语法将修改响应代码，示例:

```
error_page 404 =200 /.empty.gif;
```

如果错误响应是由代理服务器或 FastCGI/uwsgi/SCGI 服务器处理，并且服务器可能返回不同的响应代码（例 如，200、302、401 或 404），则可以使用其返回的代码进行响应:

```
error_page   404  =  /404.php;
```

9.3.13 if_modified_since

语法:if_modified_since [off|exact|before]

默认值:if_modified_since exact

上下文:http, server, location

指定如何将响应的修改时间与 If-Modified-Since 请求头字段中的时间进行比较:

- off —忽略 If-Modified-Since 请求头字段;
- exact —完全匹配;
- before —响应的修改时间小于或等于 If-Modified-Since 请求头字段中的时间。

9.3.14 index

语法:index file [file...]

默认值:index index.html

上下文:http, server, location

定义将用作索引的文件。文件名可以包含变量。以指定的顺序检查文件。列表的最后一个元素可以是一个具有绝对路 径的文件。例入:

```
index  index.$geo.html index.0.html  /index.html;
```

应该注意的是，使用索引文件发起内部重定向，可以在不同的 location 处理请求。示例，使用以下配置:

```
location = / {
index index.html;
}

location / {
...
}
```

/ 请求实际上是将在第二个 location 处理为 /index.html 。

9.3.15 internal

语法:internal

默认值:no

上下文:location

指定给定的 location 只能用于内部请求。对于外部请求，返回客户端错误 404（未找到）。内部请求如下:

由 error_page、index、random_index 和 try_files 指令重定向的请求;

由 upstream server 的 “ X-Accel-Redirect”响应头字段重定向的请求;

由 http_ssi_module 模块的“include virtual ”命令、http_addition_module 模块指令、以及 auth_request 和 mirror 指令组成的子请求;

由 rewrite 指令更改的请求。

示例:

```
error_page 404 /404.html;
location  /404.html {
internal;
}
```

9.3.16 keepalive_timeout

语法:keepalive_timeout [time]

默认值:keepalive_timeout 75

上下文:http, server, location

第一个参数设置一个超时时间，keep-alive 客户端连接将在服务端保持打开状态。零值将禁用 keep-alive 客户端连接。可选的第二个参数在 Keep-Alive: timeout= time 响应头域中设

置一个值。两个参数可能不同。

Mozilla 和 Konqueror 会识别 Keep-Alive: timeout= time 头字段。MSIE 大约在 60 秒钟内自行 关闭 keep-alive 连接。

9.3.17 keepalive_requests

语法:keepalive_requests n

默认值:keepalive_requests 100

上下文:http, server, location

设置可通过一个 keepalive 连接提供的最大请求数。在达到最大请求数后，连接将关闭。

9.3.18 large_client_header_buffers

语法:large_client_header_buffers number size

默认值:large_client_header_buffers 4 4k/8k

上下文:http, server

设置用于读取大客户端请求头的缓冲区的最大 number（数量）和 size（大小）。请求行不能超过一个缓冲区 的大小，否则将返回 414（请求 URI 太长）错误给客户端。请求头字段也不能超过一个缓冲区的大小，否则将返回 400（错误请求）错误给客户端。缓冲区只能按需分配。默认情况下，缓冲区大小等于 8K 字节。如果在请求处理结 束之后，连接被转换为 keep-alive 状态，这些缓冲区将被释放。

9.3.19 limit_except

语法:limit_except methods {...}

默认值:no

上下文:location

限制给定的 location 内允许的 HTTP 方法。method 参数可以是以下之 一: GET 、HEAD 、POST 、PUT 、DELETE 、MKCOL 、COPY 、MOVE 、OPTIONS 、PROPFIND 、PROPPATCH 、LOCK 、UNLOCK 或 PATCH 。允许 GET 方法也将使得 HEAD 方法被允许。可以使用 http_access_module 和 http_auth_basic_module 模块指令来限制对其他方法的访问:

```
limit_except  GET {
allow  192.168.1.0/32;
deny   all;}
```

请注意，这将限制访问除 GET 和 HEAD 之外的所有方法。

9.3.20 limit_rate

语法:limit_rate speed

默认值:no

上下文:http, server, location, if in location

限制客户端的响应传输速率。rate 以字节/秒为单位。零值将禁用速率限制。限制设置是针对每个请求，因此如果 客户端同时打开两个连接，则整体速率将是指定限制的两倍。也可以在 $limit_rate 变量中设置速率限制。根据某些条件来限制速率可能会有用:

```
server {
   if ($slow) {
       set $limit_rate  4k;
      }
 }
```

9.3.21 limit_rate_after

语法:limit_rate_after time

默认值:limit_rate_after 1m

上下文:http, server, location, if in location

在已经传输一部分内容之后限制传输的速率。

```
limit_rate_after 1m;
limit_rate 100k;
```

9.3.22 listen

语法:listen address:port [default [backlog=num | rcvbuf=size | sndbuf=size | accept_filter=filter | deferred | bind | ssl]]

默认值:listen 80

上下文:server

指定监听的地址和端口。

```
listen 127.0.0.1:8000;
listen 127.0.0.1;
listen 8000;
listen *:8000;
listen localhost:8000;
```

IPv6 地址在中括号中设置:

```
listen [::]:8000;
listen [fe80::1];
```

9.3.23 location

语法:location [=|~*|^~] /uri/ { ... }*

默认值:no

上下文:server

这个参数根据URI的不同需求来进行配置，可以使用字符串与正则表达式匹配，如果要使用正则表达式，你必须指定下列前缀:

1. ~* 不区分大小写。

2. ~ 区分大小写。

要确定该指令匹配特定的查询，程序将首先对字符串进行匹配，字符串匹配将作为查询的开始，最确切的匹配将被使用。然后，正则表达式的匹配查询开始，匹配查询的第一个正则表达式找到后会停止搜索，如果没有找到正则表达式，将使用字符串的搜索结果。在一些操作系统，如Mac OS X和Cygwin，字符串将通过不区分大小写的方式完成匹配，但是，比较仅限于单字节的语言环境。正则表达式可以包含捕获，并用于其它指令中。可以使用“^~”标记禁止在字符串匹配后检查正则表达式，如果最确切的匹配location有这个标记，那么正则表达式不会被检查。使用“=”标记可以在URI和location之间定义精确的匹配，在精确匹配完成后并不进行额外的搜索，例如有请求“/”发生，则可以使用“location = /”来加速这个处理。即使没有“=”和“^~”标记，精确的匹配location在找到后同样会停止查询。下面是各种查询方式的总结：1、前缀“=”表示精确匹配查询，如果找到，立即停止查询。2、指令仍然使用标准字符串，如果匹配使用“^~”前缀，停止查询。3、正则表达式按照他们在配置文件中定义的顺序。4、如果第三条产生一个匹配，这个匹配将被使用，否则将使用第二条的匹配。例:

```
location  = / {
   // 只匹配 / 的查询.
  [ configuration A ]
}
location  / {
     //匹配任何以 / 开始的查询，但是正则表达式与一些较长的字符串将被首先匹配。
  [ configuration B ]
}
location ^~ /images/ {
    //匹配任何以 /images/ 开始的查询并且停止搜索，不检查正则表达式。
  [ configuration C ]
}
location ~* \.(gif|jpg|jpeg)$ {
    //匹配任何以gif, jpg, or jpeg结尾的文件,
    ↪  但是所有 /images/ 目录的请求将在Configuration C中处理。
  [ configuration D ]
}
```

各请求的处理如下例:

```
·/ -> configuration A
·/documents/document.html -> configuration B
·/images/1.gif -> configuration C
·/documents/1.jpg -> configuration D
```

注意你可以以任何顺序定义这4个配置并且匹配结果是相同的，但当使用嵌套的location结构时可能会将配置文件变的复杂并且产生一些比较意外的结果。标记“@”指定一个命名的location，这种location并不会在正常请求中执行，它们仅使用在内部重定向请求中。

9.3.24 log_not_found

语法:log_not_found [on|off]

默认值:log_not_found on

上下文:http, server, location

启用或禁用将文件未找到错误记录到 error_log 中。

9.3.25 log_subrequest

语法:log_subrequest [on|off]

默认值:log_subrequest off

上下文:http, server, location

启用或禁用将子请求记录到 access_log 中。

9.3.26 msie_padding

语法:msie_padding [on|off]

默认值:msie_padding on

上下文:http, server, location

启用或禁用向状态超过 400 的 MSIE 客户端响应添加注释以将响应大小增加到 512 字节。

9.3.27 msie_refresh

语法:msie_refresh [on|off]

默认值:msie_refresh off

上下文:http, server, location

启用或禁用发布刷新替代 MSIE 客户端重定向

9.3.28 open_file_cache

语法:open_file_cache max = N [inactive = time] | off

默认值:open_file_cache off

上下文:http, server, location

配置一个缓存，可以存储:

- 打开文件描述符、大小和修改时间。
- 有关目录是否存在信息。

- 文件查找错误，如找不到文件、没有读取权限等。

可选参数:

- `max` - 设置缓存中元素的最大数量。在缓存溢出时，最近最少使用的（LRU）元素将被移除;
- `inactive` - 定义一个时间，在这个时间之后，元素在缓存中将被删除，默认是 60 秒;
- `off` - 禁用缓存.

示例:

```
open_file_cache max=1000 inactive=20s;
open_file_cache_valid    30s;
open_file_cache_min_uses 2;
open_file_cache_errors   on;
```

9.3.29 open_file_cache_errors

语法:open_file_cache_errors on | off

默认值:open_file_cache_errors off

上下文:http, server, location

通过 open_file_cache 启用或禁用文件查找错误缓存。

9.3.30 open_file_cache_min_uses

语法:open_file_cache_min_uses number

默认值:open_file_cache_min_uses 1

上下文:http, server, location

设置由 open_file_cache 指令的 inactive 参数配置的时间段内文件访问的最小 number（次数），这是 文件描述符在缓存中保持打开状态所必需的。

9.3.31 open_file_cache_valid

语法:open_file_cache_valid time

默认值:open_file_cache_valid 60

上下文:http, server, location

设置 open_file_cache 元素应该验证的时间。

9.3.32 port_in_redirect

语法:port_in_redirect [on|off]

默认值:port_in_redirect on

上下文:http, server, location

启用或禁用指定由 BES WebServer 发出的绝对重定向中的端口。

9.3.33 recursive_error_pages

语法:recursive_error_pages [on|off]

默认值:recursive_error_pages off

上下文:http, server, location

启用或禁用使用 error_page 指令执行多个重定向。这种重定向的数量是有限的。

9.3.34 resolver

语法:resolver address

默认值:no

上下文:http, server, location

将用于解析 upstream 服务器名称的名称服务器配置进指定的地址，示例:

```
resolver 127.0.0.1 [::1]:5353;
```

地址可以指定为域名或 IP 地址，也可以指定一个可选的端口。如果没有指定端口，则使用端口 53。名称服务器以轮询方式查询。

9.3.35 resolver_timeout

语法:resolver_timeout time

默认值:30

上下文:http, server, location

设置 DNS 操作的超时时间。

9.3.36 root

语法:root path

默认值:root html

上下文:http, server, location, if in location

指定请求 URL 的本地文件根目录。

```
location  /i/ {
   root  /spool/w3;
 }
```

当请求”/i/top.gif”时，将会转向文件“/spool/w3/i/top.gif”。

9.3.37 satisfy

语法:satisfy any|all

默认值:satisfy all;

上下文:http,server,location

如果所有(all)或至少一个(any)允许访问，则允许访问。

```
location  / {
   satisfy any;
   allow  192.168.1.0/32;
   deny   all;
   auth_basic    "closed site";
   auth_basic_user_file  conf/htpasswd;
 }
```

9.3.38 send_timeout

语法:send_timeout the time

默认值:send_timeout 60

上下文:http, server, location

设置向客户端发送响应的超时时间。超时设置只限定在两次连续的写入操作之间，而不是用于整个响应的传输。如果 客户端在此时间内没有收到任何内容，则连接将被关闭。

9.3.39 sendfile

语法:sendfile [on|off]

默认值:sendfile off

上下文:http, server, location

启用或禁用 sendfile() 。

9.3.40 server

语法:server {...}

默认值:no

上下文:http

设置虚拟服务器的配置。基于 IP（基于 IP 地址）和基于名称（基于 Host 请求头字段）的虚拟服务器之间没有明确的界限。相反，listen 指令描述应接受服务器连接的所有地址和端口，server_name 指令列出所有服务器名 称。如何处理请求文档中提供了配置示例。

9.3.41 server_name

语法:server_name name [...]

默认值:server_name hostname

上下文:server

设置虚拟服务器名称，示例:

```
server {
   server_name example.com www.example.com;
}
```

第一个名字将成为主服务器名称。服务器名称可以包含一个星号（* ）以代替名称的第一部分或最后一部分

```
server {
   server_name   example.com  *.example.com  www.example.*;
}
```

这样的名称被称为通配符名称。上面提到的前两个名字可以合并成一个:

```
server {
    server_name  .example.com;
}
```

也可以在服务器名称中使用正则表达式，在名称前面加上波浪号（~）:

```
server {
    server_name   www.example.com   ~^www\d+\.example\.com$;
}
```

9.3.42 server_name_in_redirect

语法:server_name_in_redirect on|off

默认值:server_name_in_redirect on

上下文:http, server, location

启用或禁用在由 BES WebServer 发出的 absolute 指令中使用由 server_name 指令指定的主服务器名称。当禁用主服 务器名称时，将使用 Host 请求头字段中的名称。如果此字段不存在，则使用服务器的 IP 地址。重定向中使用端口由 port_in_redirect 指令控制。

9.3.43 server_names_hash_max_size

语法:server_names_hash_max_size number

默认值:server_names_hash_max_size 512

上下文:http

设置服务器名称哈希表的最大值大小。

9.3.44 server_names_hash_bucket_size

语法:server_names_hash_bucket_size number

默认值:server_names_hash_bucket_size 32/64/128

上下文:http

设置服务器名称哈希表的存储桶大小。默认值取决于处理器缓存行的大小。

9.3.45 server_tokens

语法:server_tokens on|off

默认值:server_tokens on

上下文:http, server, location

在错误页面和 Server 响应头字段中启用或禁用发送 BES WebServer 版本。

9.3.46 tcp_nodelay

语法:tcp_nodelay [on|off]

默认值:tcp_nodelay on

上下文:http, server, location

启用或禁用 TCP_NODELAY 选项的使用。该选项仅在连接转换到 keep-alive 状态时启用。

9.3.47 tcp_nopush

语法:tcp_nopush [on|off]

默认值:tcp_nopush off

上下文:http, server, location

启用或禁用在 FreeBSD 上使用 TCP_NOPUSH 套接字选项或在 Linux 上使用 TCP_CORK 套接字选项。这些选 项只有在使用 sendfile 时才能使用。启用该选项将允许:

在 Linux 和 FreeBSD 4 上，在同一包中可发送响应头和文件开头。

以完整包的方式发送一个文件

9.3.48 try_files

语法:try_files file1 [file2 ... filen] fallback

默认值:none

上下文:location

以指定顺序检查文件是否存在，并使用第一个找到的文件进行请求处理。处理将在当前上下文中执行。指向文件的路 径根据 root 和 alias 指令从 file 参数构造。可以通过在名称末尾指定斜线来检查目录是否存在，例 如，$URI/ 。如果找不到任何文件，则内部重定向将指向最后一个参数中指定的 uri 。示例:

```
location / {
    try_files index.html index.htm @fallback;
}
location @fallback {
    root  /var/www/error;
    index index.html;
}
```

9.3.49 types

语法:types {...}

上下文:http, server, location

文件拓展名和MIME-types类型映射表

```
 types  {
     text/html    html;
     image/gif    gif;
     image/jpeg   jpg;
 }
```

完整的映射列表在 `conf/mime.types`.

9.4 stream模块

此模块主要用来配置四层代理，包含如下核心指令。

9.4.1 listen

语法: listen address:port [ssl] [udp] [fastopen=number] [backlog=number] [rcvbuf=size] [sndbuf=size] [bind] [ipv6only=on|off] [reuseport] [so_keepalive=on|off|[keepidle]:[keepintvl]:[keepcnt]];

默认值: —

上下文: server

说明：指定监听地址

ssl: 指定该端口使用 ssl 协议

udp: 指定该端口使用 udp 协议，需要开启

fastopen：启用“TCP Fast Open”

backlog: listen 函数 backlog，linux 系统为 511

sndbuf：发送缓冲区大小

rcvbuf: 接收缓冲区大小

reuseport：指定为每个工作进程创建监听套接字，Linux 3.9+有效

9.4.2 preread_buffer_size

语法: preread_buffer_size size;

默认值: preread_buffer_size 16k;

上下文: stream, server

说明：preread 缓冲区大小，preread 为读取协议初始数据

9.4.3 preread_timeout

语法: preread_timeout timeout;

默认值: preread_timeout 30s;

上下文: stream, server

说明:preread 超时时间

9.4.4 resolver

语法: resolver address ... [valid=time] [ipv6=on|off] ;

默认值: 无

上下文: stream, server

说明：设置 DNS 服务器

9.4.5 resolver_timeout

语法: resolver_timeout time;

默认值: resolver_timeout 30s;

上下文: stream, server

说明：DNS 解析超时时间

9.4.6 server

语法: server { ... }

默认值: —无

上下文: stream

说明：server 块标识

9.4.7 stream

语法: stream { ... }

默认值: 无

上下文: main

说明：stream 块标识

9.4.8 tcp_nodelay

语法: tcp_nodelay on | off;

默认值: tcp_nodelay on;

上下文: stream, server

说明：启用或禁用 TCP_NODELAY 选项的使用，该选项对客户端和代理服务器连接都启用

9.4.9 variables_hash_bucket_size

语法: variables_hash_bucket_size size;

默认值: variables_hash_bucket_size 64;

上下文: stream

说明：设置用于存储变量的哈希桶大小

9.4.10 variables_hash_max_size

语法: variables_hash_max_size size;

默认值: variables_hash_max_size 1024;

上下文: stream

说明：存储变量哈希表最大值

9.5 upstream模块

这个模块提供一个简单方法来实现在轮询和客户端IP之间的后端服务器负荷平衡。

配置范例:

```
upstream backend  {
    server backend1.example.com weight=5;
    server backend2.example.com:8080;
    server unix:/tmp/backend3;
}
server {
    location / {
        proxy_pass  http://backend;
    }
}
```

配置指导

9.5.1 server

语法:server address [parameters] |server : port1-port2 | server : <port1, port2-port3>

默认值: none

上下文: upstream

指定工作节点及其他参数

weight=number：设置节点权重，默认为 1

max_conns=number：和后端工作节点的最大连接数，如果开启了 ZONE,最大连接数为所有工作进程的总数，sticky 不支持这个参数。

max_fails=number：在单位周期为 fail_timeout 的时间中失败次数达到 max_fails 次，在这个周期内，如果后端同一个节点不可用，将该节点标记为不可用，并等待下一个周期（同样时长为fail_timeout）再次尝试

fail_timeout=time：单位周期，默认为 10s

backup：标记节点为备机，当其他节点都不可用时，请求转发到备机

down：标记节点永久不可用

ip段配置举例:

```
upstream ups {
  server <192.168.29.0-192.168.29.1>:80-81;
}
等效于
upstream ups {
    server 192.168.29.0:80;
    server 192.168.29.0:81;
    server 192.168.29.1:80;
    server 192.168.29.1:81;
}
```

9.6 access模块

此模块提供了一个简易的基于主机的访问控制。

http_access_module 模块使有可能对特定IP客户端进行控。规则检查按照第一次匹配的顺序

配置样例

```
location / {
    deny    192.168.1.1;
    allow   192.168.1.0/24;
    allow   10.1.1.0/16;
    deny    all;
  }
```

在上面的例子中,仅允许网段 10.1.1.0/16 和 192.168.1.0/24中除 192.168.1.1之外的ip访问。

当执行很多规则时,最好使用 http_geo_module 模块。

9.6.1 允许

语法:allow [address | CIDR | all]

默认值:no

上下文:http, server, location, limit_except

以上描述的网络地址有权直接访问

9.6.2 禁止

语法:deny [address | CIDR | all]

默认值:no

上下文:http, server, location, limit_except

以上描述的网络地址拒绝访问

9.7 authbasic模块

该模块可以使你使用用户名和密码基于 HTTP 基本认证方法来保护你的站点或其部分内容。

实例配置

```
location  /  {
     auth_basic            "Restricted";
     auth_basic_user_file  conf/htpasswd;
 }
```

9.7.1 auth_basic

语法： auth_basic [text|off]

默认值:auth_basic off

上下文： http, server, location, limit_except

该指令包含用于 HTTP 基本认证的测试名和密码, 分配的参数用于认证领域。值“off”可以使其覆盖来自上层指令的继承性。

9.7.2 auth_basic_user_file

语法： auth_basic_user_file the_file

默认值:no

上下文： http, server, location, limit_except

该指令为某认证领域指定 htpasswd 文件名。

文件格式类似于下面的内容:

```
用户名:密码用户名2:密码2:注释用户名3:密码3
```

密码必须使用函数 crypt(3) 加密。你可以使用来自 Apache 的 htpasswd 工具来创建密码文件。

你也可以使用perl 创建密码文件, pw.pl 的内容:

```
#!/usr/bin/perluse strict;my $pw=$ARGV[0] ;print crypt($pw,$pw)."\n";
```

然后执行

```
chmod +x pw.pl./pw.pl passwordpapAq5PwY/QQM
```

papAq5PwY/QQM 就是password 的crypt()密码

9.8 autoindex模块

此模块用于自动生成目录列表.

http_autoindex_module只在 http_index_module模块未找到索引文件时发出请求。

配置实例

```
location  /  {
  autoindex  on;
}
```

9.8.1 autoindex

语法:autoindex [on|off]

默认值:autoindex off

上下文:http, server, location

激活/关闭自动索引

9.8.2 autoindex_exact_size

语法:autoindex_exact_size [on|off]

默认值:autoindex_exact_size on

上下文:http, server, location

设定索引时文件大小的单位(B,KB, MB 或 GB)

9.8.3 autoindex_localtime

语法:autoindex_localtime [on|off]

默认值:autoindex_localtime off

上下文:http, server, location

开启以本地时间来显示文件时间的功能。默认为关（GMT时间）

9.9 charset模块

模块将指定的字符集添加到 Content-Type 响应头域。此外，该模块可以将数据从一个 字符集转换为另一个字符集，但也存在一些限制：

（1）转换工作只能是从服务器到客户端。

（2）只能转换单字节字符集。

（3）或转为/来自 UTF-8 的单字节字符集。

示例:

```
charset         windows-1251;source_charset  koi8-r;
```

9.9.1 charset

语法:charset encoding|off

默认值:charset off

上下文:http, server, location, if in location

将指定的字符集添加到 Content-Type 响应头域。如果此字符集与 source_charset 指令指定的字符集不同，则执行转换。参数 off 取消将字符集添加到 Content-Type 响应头。可以使用一个变量来定义字符集：

```
charset $charset;
```

在这种情况下，变量的值至少要在 charset_map、charset 或 source_charset 其中一个指令配置一次。对于utf-8 、windows-1251 和 koi8-r 字符集，将 conf/koi-win 、conf/koi-utf 和 conf/win-utf 文件 包含到配置中就足够了。对于其他字符集，只需制作一个虚构的转换表即可，示例：

```
charset_map iso-8859-5 _ { }
```

另外,可以在 X-Accel-Charset 响应头域中设置一个字符集。可以使 用proxy_ignore_headers、fastcgi_ignore_headers、uwsgi_ignore_headers 和 scgi_ignore_headers 指令禁用此功能。

9.9.2 charset_map

语法:charset_map encoding1 encoding2 {...}

默认值:no

上下文:http, server, location

描述转换表，将一个字符集转换到另一个字符集。反向转换表也使用相同的数据构建。字符代码是十六进制格式。不 在 80-FF 范围内的字符将被替换为? 。当从 UTF-8 转换时，一个字节字符集中丢失的字符将被替换为 &#XXXX; 。

示例:

```
charset_map  koi8-r  windows-1251 {
   C0  FE ; # small yu
   C1  E0 ; # small a
   C2  E1 ; # small b
   C3  F6 ; # small ts
   # ...
 }
```

9.9.3 override_charset

语法:override_charset on|off

默认值:override_charset off

上下文:http, server, location, if in location

当应答已经在 Content-Type 响应头域中携带字符集时，确定是否应该对从代理或 FastCGI/uwsgi/SCGI 服务 器接收的应答执行转换。如果启用转换，则在接收到的响应中指定的字符集将用作源字符集。

9.9.4 source_charset

语法:source_charset encoding

默认值:no

上下文:http, server, location, if in location

定义响应的源字符集。如果此字符集与 charset 指令中指定的字符集不同，则执行转换。

9.10 fastcgi模块

这个模块允许BES WebServer 与FastCGI 进程交互，并通过传递参数来控制FastCGI 进程工作。

配置实例:

```
location / {
    fastcgi_pass   localhost:9000;
```

```
    fastcgi_index  index.php;
    fastcgi_param  SCRIPT_FILENAME  /home/www/scripts/php$fastcgi_script_name;
    fastcgi_param  QUERY_STRING     $query_string;
    fastcgi_param  REQUEST_METHOD   $request_method;
    fastcgi_param  CONTENT_TYPE     $content_type;
    fastcgi_param  CONTENT_LENGTH   $content_length;
}
```

9.10.1 fastcgi_buffers

语法:fastcgi_buffers the_number is_size;

默认值: fastcgi_buffers 8 4k/8k;

上下文: http, server, location

该指令集设置缓冲区的数量和大小，用于缓存从 FastCGI Server 接收到的数据。默认情况下，一个缓冲区的大小相当于一个页面的大小。根据平台的不同设置为4K/8K 。

9.10.2 fastcgi_buffer_size

语法:fastcgi_buffer_size the_size

默认值: fastcgi_buffer_size 4k/8k

上下文: http, server, location

设置读取 FastCGI 服务器收到的响应的第一部分的缓冲区的 size （大小）。该部分通常包含一个小的响应头。默认情况下，缓冲区大小等于一个内存页。为 4K 或 8K，因平台而异。但是，它可以设置得更小。

9.10.3 fastcgi_cache

语法:fastcgi_cache zone;

默认值: none

上下文: http, server, location

定义用于缓存的共享内存区域。同一个区域可以在几个地方使用。off 参数将禁用从上级配置级别继承的缓存配置。

9.10.4 fastcgi_cache_key

语法:fastcgi_cache_key line ;

默认值: none

上下文: http, server, location

为缓存定义一个 key，示例:

```
fastcgi_cache_key localhost:9000$request_uri;
```

9.10.5 fastcgi_cache_methods

语法:fastcgi_cache_methods [GET HEAD POST];

默认值: fastcgi_cache_methods GET HEAD;

上下文: http,server,location

如果此指令中存在当前客户端请求方法，那么响应将被缓存。虽然 GET 和 HEAD 方法总是在该列表中，但我们还是建议您明确指定它们。

```
fastcgi_cache_methods  POST;
```

9.10.6 fastcgi_cache_min_uses

语法:fastcgi_cache_min_uses n

默认值: fastcgi_cache_min_uses 1

上下文: http, server, location

设置指定数量（number ）请求后响应将被缓存。

9.10.7 fastcgi_cache_path

语法:fastcgi_cache_path /path/to/cache [levels=m:n keys_zone=name:time inactive=time clean_time=time]

默认值: none

上下文: http

设置缓存的路径和其他参数。缓存数据存储在文件中。缓存中的 key 和文件名是代理 URL 经过 MD5 函数处理后 得到的值。levels 参数定义缓存的层次结构级别：范围从 1 到 3 ，每个级别可接受值为 1 或 2 。示例，在以下配置中：

```
fastcgi_cache_path /data/BES WebServer/cache levels=1:2 keys_zone=one:10m;
```

缓存中的文件名如下所示：

```
/data/BES WebServer/cache/c/29/b7f54b2df7773722d382f4809d65029c
```

首先将缓存的响应写入临时文件，然后重命名该文件。从 0.8.9 版本开始，临时文件和缓存可以放在不同的文件系统上。但是，请注意，在这种情况下，文件复制将要跨两个文件系统，而不是简单的重命名操作。因此建议，对于任何给定的位置，缓存和保存临时文件的目录都应该放在同一个文件系统上。临时文件的目录根据 use_temp_path 参数设置。如果忽略此参数或将其设置为 on ，则将使用由 fastcgi_temp_path 指令设置的目录。如果该值设置为 off ，临时文件将直接放在缓存目录中。另外，所有活跃的 key 和有关数据的信息都存储在共享内存区中，其名称和大小由 keys_zone 参数配置。一个兆字节的区域可以存储大约 8 千个 key。

9.10.8 fastcgi_cache_use_stale

语法:fastcgi_cache_use_stale [updating|error|timeout|invalid_header|http_500]

默认值: fastcgi_cache_use_stale off;

上下文: http, server, location

当在与 FastCGI 服务器通信期间发生错误时可以使用陈旧的缓存响应。该指令的参数与 fastcgi_next_upstream 指令的参数相匹配。

9.10.9 fastcgi_cache_valid

语法:fastcgi_cache_valid [http_error_code|time]

默认值: none

上下文: http, server, location

为不同的响应码设置缓存时间。示例:

```
fastcgi_cache_valid 200 302 10m;
fastcgi_cache_valid 404 1m;
```

对响应码为 200 和 302 的响应设置 10 分钟缓存，对响应码为 404 的响应设置为 1 分钟。

如果只指定缓存时间（time）:

```
fastcgi_cache_valid 5m
```

那么只缓存 200 、301 和 302 响应。

另外，可以指定 any 参数来缓存任何响应:

```
fastcgi_cache_valid 200 302 10m;
fastcgi_cache_valid 301 1h;
fastcgi_cache_valid any 1m;
```

9.10.10 fastcgi_index

语法:fastcgi_index file

默认值: none

上下文: http, server, location

在 $fastcgi_script_name 变量的值中设置一个文件名，该文件名追加到 URL 后面并以一个斜杠结尾。示例以下设置:

```
fastcgi_index index.php;
fastcgi_param SCRIPT_FILENAME /home/www/scripts/php$fastcgi_script_name;
```

9.10.11 fastcgi_hide_header

语法:fastcgi_hide_header name

上下文: http, server, location

默认情况下BES WebServer 不会从FastCGI 进程里给客户端发送”Status”和”X-Accel-...”消息头。这个指令可以用来掩饰别的headers 。如果需要”Status”和”X-Accel-...”消息头,那就需要使用这个指令让FastCGI 强制发送消息头给客户端。

9.10.12 fastcgi_ignore_client_abort

语法:fastcgi_ignore_client_abort on|off

默认值: fastcgi_ignore_client_abort off

上下文: http, server, location

这个指令用来决定忽略用户取消的请求。

9.10.13 fastcgi_intercept_errors

语法:fastcgi_intercept_errors on|off

默认值: fastcgi_intercept_errors off

上下文: http, server, location

这个指令用来决定是否要把客户端转向4xx和5xx错误页，或允许BES WebServer自动指定错误页页。注意：你需要在此明确错误页,它才是有用的。Igor 曾说：“如果没有定制的处理机制，BES WebServer不会拦截一个没有缺省页的错误“。BES WebServer 只会拦截一些小的错误，放过其他一些。

9.10.14 fastcgi_param

语法:fastcgi_param parameter value

默认值: none

上下文: http, server, location

设置应传递给 FastCGI 服务器的 parameter (参数)。该值可以包含文本、变量及其组合。当且仅当在当前级别上没有定义 fastcgi_param 指令时，这些指令才从前一级继承。以下示例展示了 PHP 的最小要求配置：

```
fastcgi_param SCRIPT_FILENAME /home/www/scripts/php$fastcgi_script_name;
fastcgi_param QUERY_STRING $query_string;
```

9.11 gzip模块

这个模块支持在线实时压缩输出数据流。

使用范例

```
gzip             on;
gzip_min_length  1000;
gzip_proxied     expired no-cache no-store private auth;
gzip_types       text/plain application/xml;
```

内置变量 $gzip_ratio 可以获取到gzip的压缩比率。

9.11.1 gzip

语法:gzip on|off

默认值:gzip off

上下文:http, server, location, if (x) location

开启或者关闭gzip模块。

9.11.2 gzip_buffers

语法:gzip_buffers number size

默认值:gzip_buffers 4 4k/8k

上下文:http, server, location

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。示例 4 4k 代表以4k为单位，按照原始数据大小以4k为单位的4倍申请内存。4 8k 代表以8k为单位，按照原始数据大小以8k为单位的4倍申请内存。

如果没有设置，默认值是申请跟原始数据相同大小的内存空间去存储gzip压缩结果。

9.11.3 gzip_comp_level

语法:gzip_comp_level 1..9

默认值:gzip_comp_level 1

上下文:http, server, location

gzip压缩比，1 压缩比最小处理速度最快，9 压缩比最大但处理最慢（传输快但比较消耗cpu）。

9.11.4 gzip_min_length

语法:gzip_min_length length

默认值:gzip_min_length 0

上下文:http, server, location

设置允许压缩的页面最小字节数，页面字节数从header头中的Content-Length中进行获取。

默认值是0，不管页面多大都压缩。

建议设置成大于1k的字节数，小于1k可能会越压越大。即: gzip_min_length 1024

9.11.5 gzip_http_version

语法:gzip_http_version 1.0|1.1

默认值:gzip_http_version 1.1

上下文:http, server, location

识别http的协议版本。由于早期的一些浏览器或者http客户端，可能不支持gzip自解压，用户就会看到乱码，所以做一些判断还是有必要的。注：21世纪都来了，现在除了类似于百度的蜘蛛之类的东西不支持自解压，99.99%的浏览器基本上都支持gzip解压了，所以可以不用设这个值，保持系统默认即可。

9.11.6 gzip_proxied

语法:gzip_proxied [off|expired|no-cache|no-store|private|no_last_modified|no_etag|auth|any] ...

默认值:gzip_proxied off

上下文:http, server, location

BES WebServer作为反向代理的时候启用，开启或者关闭后端服务器返回的结果，匹配的前提是后端服务器必须要返回包含”Via”的 header头。

- off - 关闭所有的代理结果数据的压缩
- expired - 启用压缩，如果header头中包含“Expires”头信息
- no-cache - 启用压缩，如果header头中包含“Cache-Control:no-cache”头信息
- no-store - 启用压缩，如果header头中包含“Cache-Control:no-store”头信息
- private - 启用压缩，如果header头中包含“Cache-Control:private”头信息
- no_last_modified - 启用压缩，如果header头中不包含“Last-Modified”头信息
- no_etag - 启用压缩,如果header头中不包含“ETag”头信息
- auth - 启用压缩, 如果header头中包含“Authorization”头信息
- any - 无条件启用压缩

9.11.7 gzip_types

语法:gzip_types mime-type [mime-type ...]

默认值:gzip_types text/html

上下文:http, server, location

匹配MIME类型进行压缩，（无论是否指定）“text/html”类型总是会被压缩的。

注意：如果作为http server来使用，主配置文件中要包含文件类型配置文件：

```
http{
   include       conf/mime.types;
```

```
    ......
}
```

如果你希望压缩常规的文件类型，可以写成这个样子：

```
http {
    include       conf/mime.types;
    gzip on;
    gzip_min_length  1000;
    gzip_buffers     4 8k;
    gzip_http_version 1.1;
    gzip_types       text/plain application/x-javascript text/css text/html a
    ↪  pplication/xml;
    ......
 }
```

9.12 headers模块

本模板可以设置HTTP报文的头标。

示例

```
expires     24h;
expires     0;
expires     -1;
expires     epoch;
add_header  Cache-Control  private;
```

9.12.1 增加头标

语法：add_header name value

默认值:none

上下文：http, server, location

当HTTP应答状态码为 200、204、301、302 或 304 的时候，增加指定的HTTP头标。

其中头标的值可以使用变量。

9.12.2 expires

语法：expires [time|epoch|max|off]

默认值:expires off

上下文：http, server, location

使用本指令可以控制HTTP应答中的“Expires”和“Cache-Control”的头标，（起到控制页面缓存的作用）。

可以在time值中使用正数或负数。“Expires”头标的值将通过当前系统时间加上您设定的time 值来获得。

epoch 指定“Expires”的值为 1 January, 1970, 00:00:01 GMT。

max 指定“Expires”的值为 31 December 2037 23:59:59 GMT，“Cache-Control”的值为10年。

-1 指定“Expires”的值为 服务器当前时间 -1s,即永远过期。

“Cache-Control”头标的值由您指定的时间来决定：

- 负数：Cache-Control: no-cache
- 正数或零：Cache-Control: max-age = #, # 为您指定时间的秒数。

“off”表示不修改“Expires”和“Cache-Control”的值。

9.13 index模块

语法:index file [file...]

默认值:index index.html

上下文:http, server, location

该指令用来指定用来做默认文档的文件名，可以在文件名处使用变量。如果您指定了多个文件，那么将按照您指定的顺序逐个查找。可以在列表末尾加上一个绝对路径名的文件。

示例:

```
index  index.$geo.html  index.0.html  /index.html;
```

9.14 limit_conn模块

本模块可以针对条件，进行会话的并发连接数控制。（示例：限制每个IP的并发连接数。）

配置示例

```
http {
    limit_conn_zone $binary_remote_addr zone=one:10m;
    ...
    server {
       ...
       location /download/{
           limit_conn   one  1;
        }
```

9.14.1 limit_conn_zone

语法：limit_zone $variable zone=zone_name:the_size

默认值:no

上下文：http

本指令定义了一个数据区，里面记录会话状态信息。$variable 定义判断会话的变量；the_size 定义记录区的总容量。

例子：

```
limit_conn_zone $binary_remote_addr zone=one:10m;
```

定义一个叫“one”的记录区，总容量为 10M，以变量 $binary_remote_addr 作为会话的判断基准（即一个地址一个会话）。

注意，在这里使用的是 $binary_remote_addr 而不是 $remote_addr。

$remote_addr 的长度为 7 至 15 bytes，会话信息的长度为 32 或 64 bytes。而 $binary_remote_addr 的长度为 4 bytes，会话信息的长度为 32 bytes。

当区的大小为 1M 的时候，大约可以记录 32000 个会话信息（一个会话占用 32 bytes）。

9.14.2 limit_conn

语法：limit_conn zone_name the_size

默认值:no

上下文：http, server, location

指定一个会话最大的并发连接数。当超过指定的最发并发连接数时，服务器将返回“Service unavailable”(503)。

例子：

```
limit_zone   one  $binary_remote_addr  10m;
    server{
        location /download/ {
            limit_conn   one  1;
        }
```

定义一个叫“one”的记录区，总容量为 10M，以变量 $binary_remote_addr 作为会话的判断基准（即一个地址一个会话）。限制 /download/ 目录下，一个会话只能进行一个连接。简单点，就是限制 /download/ 目录下，一个IP只能发起一个连接，多过一个，一律503。

9.15 limit_request模块

用于限制每个已定义 key 的请求处理速率，特别是来自单个 IP 地 址请求的处理速率。限制机制采用了 leaky bucket （漏桶算法）方法完成。

示例：

```
http {
    limit_req_zone  $binary_remote_addr  zone=one:10m   rate=1r/s;
    ...
    server {
```

```
...
    location /search/ {
        limit_req   zone=one  burst=5;
    }
```

语法:limit_req_zone $session_variable zone=name_of_zone**:**size rate=rate

默认值:none

上下文:http

为共享内存区域设置参数，该区域将保留各种键的状态。特别是，该状态包含当前的连接数。key 可以包含文本、变量及其组合。不包括有空键值的请求。

```
limit_req_zone  $binary_remote_addr  zone=one:10m   rate=1r/s;
```

在这里，状态保持在 10 兆字节的区域 one，并且该区域的平均请求处理速率不能超过每秒 1 个请求。客户端 IP 地址作为 key。请注意,不是 $remote_addr ,而是使用 $binary_remote_addr 变 量。$binary_remote_addr 变量的大小始终为 4 个字节，对于 IPv6 地址则为 16 个字节。存储状态在 32 位平 台上始终占用 32 或 64 个字节，在 64 位平台上占用 64 个字节。一兆字节的区域可以保持大约 32000 个 32 字节的状态或大约 16000 个 64 字节的状态或大约 8000 个 128 字节的状态。如果区域存储耗尽，最近最少使用的状态将被删除。即使在此之后无法创建新状态，该请求也会因错误而终止。速率以每秒请求数（r/s）指定。如果需要每秒小于一个请求的速率，则按每分钟请求（r/m）指定。示例，每秒半请 求是 30r/m。

语法:limit_req zone=**zone** burst=**burst** [nodelay]

默认值:none

上下文:http, server, location

设置共享内存区域和请求的最大突发大小。如果请求速率超过为某个区域配置的速率，则它们的处理会延迟，从而使 请求以定义的速率处理。过多的请求被延迟，直到它们的数量超过最大突发大小，在这种情况下请求被终止并出现错 误。默认情况下，最大突发大小等于零。示例:

```
limit_req_zone  $binary_remote_addr  zone=one:10m   rate=1r/s;
server {
    location /search/ {
        limit_req   zone=one  burst=5;
     }
```

平均每秒不超过 1 个请求，并且突发不超过 5 个请求。如果在限制期间延迟请求过多，则不需要使用参数 nodelay :

```
 limit_req   zone=one  burst=5  nodelay;
```

9.16 log模块

http_log_module 模块可让请求日志以指定的格式写入。请求会在处理结束的 location 的上下文中记录。如果在请求处理期间发生内部重定向，可能会造成与原始 location 不同。

示例:

```
log_format   '$remote_addr - $remote_user [$time_local]' '"$request" $status 」
↪  $bytes_sent' '"$http_referer" "$http_user_agent" "$gzip_ratio"';
access_log  /spool/logs/BES WebServer-access.log  gzip  buffer=32k;
```

9.16.1 access_log

语法：access_log path [format [buffer=size | off]

默认值:access_log log/access.log combined

上下文：http, server, location

指令 access_log 指派路径、格式和缓存大小。参数“off”将清除当前级别的所有 access_log 指令。如果未指定格式，则使用预置的“combined”格式。缓存不能大于能写入磁盘的文件的最大大小。在 FreeBSD 3.0-6.0 ，缓存大小无此限制。

9.16.2 log_format

语法：log_format name format [format ...]

默认值:log_format combined “...”

上下文：httpserver

指定日志格式，日志格式可以包含公共变量和仅在日志写入时存在的变量:

- $body_bytes_sent, the number of bytes, transmitted to client minus the response headers
- $bytes_sent, 发送给客户端的字节数。
- $connection, 连接序列号。
- $msec, 以秒为单位的时间，日志写入时的毫秒精度。
- $pipe, 如果请求是 pipe，则为 p，否则为。
- $request_length, 请求长度（包括请求行、头部和请求体）。
- $request_time,以毫秒为精度的请求处理时间，以秒为单位。从客户端读取第一个字节到最后一个字节发送到客户端并写入日志 过程的时间。
- $status, 响应状态。
- $time_local,本地时间采用 的格式

发送到客户端的头字段前缀为 sent_http_ ，示例 $sent_http_content_range 。配置始终包含预定义的 combined 格式:

```
log_format  combined  '$remote_addr - $remote_user [$time_local]' '"$request"」
↪  $status $body_bytes_sent ' '"$http_referer" "$http_user_agent"';
```

9.17 map

此模块用于创建一个变量，该变量的值依赖于其他变量的值。

示例:

```
map  $http_host  $name {
    hostnames;
    default          0;
    example.com      1;
    *.example.com    1;
    test.com         2;
    *.test.com       2;
    .site.com        3;
  }
```

9.17.1 map

语法:map $var1 $var2 { ... }

默认值:none

上下文:http

创建一个新变量，其值取决于第一个参数中指定的一个或多个源变量的值。map 块内的参数指定源和结果值之间的映射。还支持以下特殊参数:

- default —如果源值不匹配指定变体，则设置结果值。如果未指定 default，则默认结果值为空字符串。
- hostnames —表示源值可以是具有前缀或后缀掩码的主机名:

```
*.example.com  1;
```

以下两条记录

```
example.com    1;
*.example.com  1;
```

可以合并:

```
 .example.com   1;
```

- include —包含一个包含值的文件。可以有多个包含。

9.17.2 map_hash_max_size

语法:map_hash_max_size number

默认值:map_hash_max_size 2048

上下文:http

设置 map 变量哈希表的最大大小（size ）。

9.17.3 map_hash_bucket_size

语法:map_hash_bucket_size n

默认值:map_hash_bucket_size 32/64/128

上下文:http

设置 map 变量哈希表的桶大小。默认值取决于处理器的缓存行大小。

9.18 proxy模块

本模块允许将请求传递给另一台服务器，这是种 HTTP/1.0 版本的无请求保持代理。(因为每个请求都是在后台连接中创建和销毁的) BES WebServer 和浏览器使用 HTTP/1.1 进行对话,而在后台服务中使用 HTTP/1.0;

示例

```
location / {
    proxy_pass        http://localhost:8000;
    proxy_set_header  X-Real-IP  $remote_addr;
  }
```

注意一点,当使用HTTP PROXY 模块时(或者甚至是使用FastCGI时),用户的整个请求会在BES WebServer中缓冲直至传送给后端被代理的服务器.因此,上传进度的测算就会运作得不正确,如果它们通过测算后端服务器收到的数据来工作的话

9.18.1 proxy_buffer_size

语法:proxy_buffer_size the_size

默认值:proxy_buffer_size 4k/8K

上下文:http, server, location

设置用于读取从代理服务器收到的第一部分响应的缓冲区大小（size ）。这部分通常包含一个小的响应头。默认情 况下，缓冲区大小等于一个内存页。4K 或 8K，因平台而异。但是，它可以设置得更小。

9.18.2 proxy_buffering

语法:proxy_buffering on|off

默认值:proxy_buffering on

上下文:http, server, location

启用或禁用来自代理服务器的响应缓冲。当启用缓冲时，BES WebServer 会尽可能快地收到接收来自代理服务器的响应，并将其保存到由 proxy_buffer_size 和 proxy_buffers 指令设置的缓冲区中。如果内存放不下整个响应，响应的一部分可以保存到磁盘上的临时文件中。写入临时文件由 proxy_max_temp_file_size 和 proxy_temp_file_write_size 指令控制。

当缓冲被禁用时，BES WebServer 在收到响应时立即同步传递给客户端，不会尝试从代理服务器读取整个响应。BES WebServer 一次可以从服务器接收的最大数据量由 proxy_buffer_size 指令设置。

通过在 X-Accel-Buffering 响应头字段中通过 yes 或 no 也可以启用或禁用缓冲。可以使用 proxy_ignore_headers 指令禁用此功能。

9.18.3 proxy_buffers

语法:proxy_buffers the_number is_size;

默认值:proxy_buffers 8 4k/8k;

上下文:http, server, location

设置单个连接从代理服务器读取响应的缓冲区的 number （数量）和 size （大小）。默认情况下，缓冲区大 小等于一个内存页。为 4K 或 8K，因平台而异。

9.18.4 proxy_busy_buffers_size

语法:proxy_busy_buffers_size size;

默认值:proxy_busy_buffers_size proxy_buffer_size 2;

上下文:http, server, location, if

当启用代理服务器响应缓冲时，限制缓冲区的总大小（size ）在当响应尚未被完全读取时可向客户端发送响应。同 时，其余的缓冲区可以用来读取响应，如果需要的话，缓冲部分响应到临时文件中。默认情况下，size 受 proxy_buffer_size 和 proxy_buffers 指令设置的两个缓冲区的大小限制。

9.18.5 proxy_connect_timeout

语法:proxy_connect_timeout timeout_in_seconds

上下文:http, server, location

定义与代理服务器建立连接的超时时间。应该注意，此超时时间通常不会超过 75 秒。

9.18.6 proxy_headers_hash_bucket_size

语法:proxy_headers_hash_bucket_size size;

默认值:proxy_headers_hash_bucket_size 64;

上下文:http, server, location, if

设置 proxy_hide_header 和 proxy_set_header 指令使用的哈希表的桶 size 。

9.18.7 proxy_headers_hash_max_size

语法:proxy_headers_hash_max_size size;

默认值:proxy_headers_hash_max_size 512;

上下文:http, server, location, if

设置 proxy_hide_header 和 proxy_set_header 指令使用的哈希表的最大 size 。

9.18.8 proxy_hide_header

语法:proxy_hide_header the_header

上下文:http, server, location

默认情况下，BES WebServer 不会从代理服务器向客户端的响应中传递头字段 Date、Server、X-Pad 和 X-Accel- ...。proxy_hide_header 指令设置不传递的其他字段。相反，如果需要允许传递字段，则可以使用 proxy_pass_header 指令设置。

9.18.9 proxy_ignore_client_abort

语法:proxy_ignore_client_abort [on|off]

默认值:proxy_ignore_client_abort off

上下文:http, server, location

确定在客户端关闭连接不等待响应时是否应关闭与代理服务器的连接。

9.18.10 proxy_intercept_errors

语法:proxy_intercept_errors [on|off]

默认值:proxy_intercept_errors off

上下文:http, server, location

确定状态码大于或等于 300 的代理响应是应该传递给客户端还是拦截并重定向到 BES WebServer 以便使用 error_page 指令进行处理。

9.18.11 proxy_max_temp_file_size

语法:proxy_max_temp_file_size size;

默认值:proxy_max_temp_file_size 1G;

上下文:http, server, location, if

当启用缓冲来自代理服务器的响应,并且整个响应不符合 proxy_buffer_size 和 proxy_buffers 指令设置的 缓冲时，部分响应可以保存到临时文件中。该指令设置临时文件的最大 size 。一次写入临时文件的数据大小由 proxy_temp_file_write_size 指令设置。零值则禁止将缓冲响应写入临时文件。

9.18.12 proxy_method

语法:proxy_method [method]

默认值: None

上下文:http, server, location

指定转发到代理服务器的请求使用的 HTTP 方法（method ），而不是使用来自客户端请求的方法。参数值可以包 含变量。

示例:

```
proxy_method PROPFIND;
```

9.18.13 proxy_next_upstream

语法:proxy_next_upstream [error|timeout|invalid_header|http_500|http_503|http_404|off]

默认值:proxy_next_upstream error timeout

上下文:http, server, location

指定应将请求传递到下一个服务器的条件:

- error —与服务器建立连接，向其传递请求或读取响应头时发生错误;
- timeout —在与服务器建立连接，向其传递请求或读取响应头时发生超时;
- invalid_header —服务器返回空或无效响应;
- http_500 —服务器返回代码为 500 的响应;
- http_503 —服务器返回代码为 503 的响应;
- http_404 —服务器返回代码 404 的响应;
- off —禁止将请求传递给下一个服务器。

注意：只有在尚未向客户端发送任何内容的情况下，才能将请求传递给下一个服务器。也就是说，如果在传输响 应的过程中发生错误或超时，则无法修复此问题。

9.18.14 proxy_pass

语法:proxy_pass URL

默认值:no

上下文:location, if in location

设置代理服务器的协议、地址以及应映射位置的可选 URI。协议可以指定 http 或 https 。可以将地址指定为 域名或 IP 地址，以及一个可选端口号:

```
proxy_pass http://localhost:8000/uri/;
```

或者在 unix 单词后面指定使用冒号括起来的 UNIX 域套接字路径:

```
proxy_pass http://unix:/tmp/backend.socket:/uri/;
```

如果域名解析为多个地址，则所有这些地址将以轮询的方式使用。此外，可以将地址指定为服务器组

参数值可以包含变量。在这种情况下，如果将地址指定为域名，则在所描述的服务器组中搜索名称，如果未找到，则 使用解析器确定。请求 URI 按如下方式传递给服务器：

- 如果指定了带有 URI 的 proxy_pass , 那么当请求传递给服务器时, 与该位置（location）匹配的规范化请求 URI 的部分将被指令中指定的 URI 替换：

```
location /name/ {
    proxy_pass http://127.0.0.1/remote/;
}
```

如果指定了没有 URI 的 proxy_pass ，则请求 URI 将以与处理原始请求时客户端发送的格式相同的形式传 递给服务器，或者在处理更改的 URI 时传递完整的规范化请求 URI：

```
location /some/path/ {
    proxy_pass http://127.0.0.1;
}
```

在某些情况下，无法确定请求 URI 要替换的部分：

- 使用正则表达式指定 location （位置）时，以及在命名位置内指定位置。在这些情况下，应指定 proxy_pass 而不使用 URI。“;
- 使用 rewrite 指令在代理位置内更改 URI 时，将使用相同的配置来处理请求（break ）：

```
location /name/ {
     rewrite /name/([^/]+) /users?name=$1 break;
     proxy_pass http://127.0.0.1;
 }
```

在这种情况下，将忽略指令中指定的 URI，并将完整更改的请求 URI 传递给服务器.

- 在 proxy_pass 中使用变量时：

```
location /name/ {
    proxy_pass http://127.0.0.1$request_uri;
}
```

在这种情况下，如果在指令中指定了 URI，则将其原样传递给服务器，替换原始请求 URI。

9.18.15 proxy_pass_header

语法:proxy_pass_header the_name

上下文:http, server, location

允许将已禁用的头字段从代理服务器传递到客户端

示例:

```
location / {
    proxy_pass_header Server;
    proxy_pass_header X-MyHeader;
 }
```

9.18.16 proxy_pass_request_body

语法:proxy_pass_request_body [on | off] ;

默认值:proxy_pass_request_body on;

上下文:http, server, location

指示是否将原始请求体传递给代理服务器。

9.18.17 proxy_pass_request_headers

语法:proxy_pass_request_headers [on | off] ;

默认值:proxy_pass_request_headers on;

上下文:http, server, location

指示是否将原始请求的 header 字段传递给代理服务器。

9.18.18 proxy_redirect

语法:proxy_redirect [default|off|redirect replacement]

默认值:proxy_redirect default

上下文:http, server, location

设置代理服务器响应 header 中的 Location 和 Refresh 字段应要更改的文本。假设代理服务器返回header 字段为 Location: http://localhost:8000/two/some/uri/ 。

```
proxy_redirect   http://localhost:8000/two/   http://frontend/one/;
```

将此字符串重写为 `Location: http://frontend/one/some/uri/`。

9.18.19 proxy_read_timeout

语法:proxy_read_timeout the_time

默认值:proxy_read_timeout 60

上下文:http, server, location

定义从代理服务器读取响应的超时时间。该超时时间仅针对两个连续的读操作之间设置，而不是整个响应的传输过 程。如果代理服务器在该时间内未传输任何内容，则关闭连接。

9.18.20 proxy_send_lowat

语法:proxy_send_lowat [on | off]

默认值:proxy_send_lowat off;

上下文:http, server, location, if

如果指令设置为非零值,则 BES WebServer 将尝试通过使用 kqueue 方法的 NOTE_LOWAT 标志或有指定大小的 SO_SNDLOWAT 套接字选项来最小化到代理服务器的传出连接上的发送操作数。

在 Linux、Solaris 和 Windows 上忽略此指令。

9.18.21 proxy_send_timeout

语法:proxy_send_timeout time_in_seconds

默认值:proxy_send_timeout 60

上下文:http, server, location

设置将请求传输到代理服务器的超时时间。超时时间仅作用于两个连续的写操作之间，而不是整个请求的传输过程。如果代理服务器在该时间内未收到任何内容，则关闭连接。

9.18.22 proxy_set_header

语法:proxy_set_header header value

默认值: Host and Connection

上下文:http, server, location

允许将字段重新定义或附加到传递给代理服务器的请求 header。该值可以包含文本、变量及其组合。当且仅当在当 前级别上没有定义 proxy_set_header 指令时，这些指令才从上层级别继承。默认情况下，只重新定义了两个字段:

```
proxy_set_header Host $proxy_host;
proxy_set_header Connection Close;
```

如果启用了缓存，则来自原始请求的 header 字段 If-Modified-Since、If-Unmodified-Since、If-None-Match、If-Match、Range 和 If-Range 不会传递给代理服务器。一个未经更改的请求头（header）字段 Host 可以像这样传递:

```
 proxy_set_header Host $http_host;
```

但是，如果客户端请求 header 中不存在此字段，则不会传递任何内容。在这种情况下，最好使用 $host 变量——它的值等于 Host 请求头字段中的服务器名称，或者如果此字段不存在则等于主服务器名称:

```
proxy_set_header Host $host;
```

此外，服务器名称可以与代理服务器的端口一起传递：

```
proxy_set_header Host $host:$proxy_port;
```

如果头字段的值为空字符串，则此字段将不会传递给代理服务器：

```
 proxy_set_header Accept-Encoding "";
```

9.18.23 proxy_store

语法:proxy_store [on | off | path]

默认值:proxy_store off

上下文:http, server, location

允许将文件保存到磁盘。on 参数使用与 alias 或 root 指令对应的路径保存文件。off 参数禁用文件保 存。此外，可以使用带变量的字符串显式设置文件名：

```
proxy_store   /data/www$original_uri;
```

根据接收的 Last-Modified 响应 header 字段设置文件的修改时间。首先将响应写入临时文件，然后重命名该文 件。从 0.8.9 版本开始，临时文件和持久化存储可以放在不同的文件系统上。但是，请注意，在这种情况下，文件 复制需要跨越两个文件系统，而不是简单的重命名操作。因此，建议由 proxy_temp_path 指令设置的保存文件和 保存临时文件的目录都放在同一文件系统上。该指令可用于创建静态不可更改文件的本地副本，示例：

```
location /images/ {
    root                /data/www;
    error_page          404 = /fetch$uri;
 }

location /fetch {
    internal;
    proxy_pass          http://backend;
    proxy_store         on;
    proxy_store_access  user:rw  group:rw  all:r;
    proxy_temp_path     /data/temp;
    alias               /data/www;
 }
```

9.18.24 proxy_store_access

语法:proxy_store_access users:permissions [users:permission ...]

默认值:proxy_store_access user:rw

上下文:http, server, location

为新创建的文件和目录设置访问权限，示例：

```
proxy_store_access  user:rw  group:rw  all:r;
```

如果指定了任何 group 或 all 访问权限，则可以省略用户权限:

```
proxy_store_access  group:rw  all:r;
```

9.18.25 proxy_temp_file_write_size

语法:proxy_temp_file_write_size size;

默认值:proxy_temp_file_write_size 8k;

上下文:http, server, location, if

当启用缓冲从代理服务器到临时文件的响应时，限制一次写入临时文件的数据大小（size)。默认情况 下，size 由 proxy_buffer_size 和 proxy_buffers 指令设置的两个缓冲区限制。临时文件的最大大小由 proxy_max_temp_file_size 指令设置。

9.18.26 proxy_temp_path

语法:proxy_temp_path dir-path [level1 [level2 [level3] ;

默认值:proxy_temp_path proxy_temp;

上下文:http, server, location

定义用于存储临时文件的目录，其中包含从代理服务器接收的数据。在指定目录下最多可有三级子目录。示例在以下 配置:

```
proxy_temp_path /spool/BES WebServer/proxy_temp 1 2;
```

9.18.27 内嵌变量

http_proxy_module 模块支持内嵌变量，可使用 proxy_set_header 指令来聚合 header:

$proxy_host:代理服务器的地址;

$proxy_port:代理服务器的端口;

$proxy_add_x_forwarded_for:X-Forwarded-For 客户端请求头字段，其中附加了 $remote_addr 变量，以逗号分割。如果客户端请求头 中不存在 X-Forwarded-For”字段，则 $proxy_add_x_forwarded_for 变量等于 $remote_addr 变量。

$remote_addr$: 如果客户端请求中没有”$X-Forwarded-For$”，变量‘proxy_add_x_forwarded_for 等于$remote_addr‘。

9.19 rewrite模块

该模块使用 PCRE 正则表达式更改请求 URI、返回重定向和有条件地选择配置。

在 server 级别下，该模块的指令按顺序执行

重复执行:

（1）基于请求 URI 搜索 location

（2）在 location 内找的该模块的指令按顺序执行

（3）如果请求 URI 被重写，则重复循环，但不超过 10 次。

9.19.1 break

语法:break

默认值:none

上下文:server, location, if

停止处理当前的 http_rewrite_module 指令集。如果在该 location 内指定了指令，则请求的下一步处理在该位置将继续。

示例:

```
if ($slow) {
    limit_rate  10k;
    break;
}
```

9.19.2 if

语法:if (condition) { ... }

默认:none

上下文:server, location

指定的 condition 求值之后，如果为 true ，则执行在大括号内指定的该模块的指令，并在 if 指令内为该 请求分配配置。if 指令内的配置继承自上一层的配置级别。

condition 可以是以下任何一种:

- 变量名，如果变量的值为空字符串或 0 ，则为 false;
- 使用 = 和!= 运算符比较变量和字符串;
- 使用 ~（区分大小写的匹配）和 ~*（不区分大小写的匹配）运算符，变量将与正则表达式进行匹配。正 则表达式可以包含可供以后在 $1..$9 变量中重用的捕获。反操作符!~ 和!~* 也可用。如果正则 表达式包含 } 或; 字符，则整个表达式应使用单引号或双引号包围起来。
- 使用 -f 和!-f 运算符检查文件是否存在;
- 使用 -d 和!-d 运算符检查目录是否存在;
- 使用 -e 和!-e 运算符检查文件、目录或符号链接是否存在;
- ~*大小写不敏感的符合（firefox符合FireFox）;
- 使用 -x 和!-x 运算符检查是否为可执行文件;

9.19.3 return

语法:return code

默认值:none

上下文:server, location, if

停止处理并将指定的 code 返回给客户端。非标准代码 444 在不发送响应头的情况下关闭连接。

9.19.4 rewrite

语法:rewrite regex replacement flag

默认:none

上下文:server, location, if

如果指定的正则表达式与请求 URI 匹配，则 URI 将根据 replacement 中的指定进行更改。rewrite 指令按 照它们在配置文件中的出现顺序依次执行。可以使用标志来终止指令的下一步处理。如果替换以 http:// 、https:// 或 $scheme 开头的字符串，则处理流程将停止并将重定向返回给客户端。

可选的 flag 参数可以是以下之一:

- last - 停止处理当前的 http_rewrite_module 指令集并开始搜索新的 location 来匹配变更的 URI;
- break - 与 break 指令一样，停止处理当前的 http_rewrite_module 指令集;
- redirect - 返回带有 302 代码的临时重定向，如果替换字符串不以 http:// 、https:// 或 $scheme 开头，则 生效。
- permanent -返回 301 代码的永久重定向。

完整重定向 URL 根据请求模式($scheme)以及 server_name_in_redirect 和 port_in_redirect 指令 形成。示例:

```
rewrite  ^(/download/.*)/media/(.*)\..*$  $1/mp3/$2.mp3  last;
rewrite  ^(/download/.*)/audio/(.*)\..*$  $1/mp3/$2.ra   last;
return   403;
```

但是如果这些指令放在 /download/ 位置，last 标志应该用 break 替换，否则 BES Web-Server 会产生 10 个循 环并返回 500 错误:

```
location /download/ {
    rewrite  ^(/download/.*)/media/(.*)\..*$  $1/mp3/$2.mp3  break;
    rewrite  ^(/download/.*)/audio/(.*)\..*$  $1/mp3/$2.ra   break;
    return   403;
}
```

如果 replacement 包含新请求参数，则先前的请求参数将最加在它们之后。如果不希望这样，可在 replacement 的末尾加上一个问号可以避免追加，示例:

```
rewrite  ^/users/(.*)$  /show?user=$1?  last;
```

9.19.5 set

语法:set variable value

默认值:none

上下文:server, location, if

为指定的 variable 设置一个 value ， value 可以包含文本、变量及其组合。

9.19.6 uninitialized_variable_warn

语法:uninitialized_variable_warn on|off

默认值:uninitialized_variable_warn on

上下文:http, server, location, if

控制是否记录有关未初始化变量的警告。

9.20 ssi模块

此模块模块是一个过滤器，在经过它的响应中处理 SSI（服务器端包含）命令。目前，支持的 SSI 命令列表并不完整。

配置示例

```
location / {
    ssi  on;
}
```

9.20.1 ssi

语法:ssi [on | off]

默认值:ssi off

上下文:http, server, location* 在location作用域中将启用SSI文件处理.

启用或禁用处理响应中的 SSI 命领。

9.20.2 ssi_silent_errors

语法:ssi_silent_errors [on|off]

默认值:ssi_silent_errors off

上下文:http, server, location

在处理SSI文件出错时不输出错误提示: “[an error occurred while processing the directive]”

9.20.3 ssi_types

语法:ssi_types mime-type [mime-type ...]

默认值:ssi_types text/html

上下文:http, server, location

除了 text/html 之外，还可以指定在其他 MIME 类型的响应中处理 SSI 命令。特殊值 * 匹配任何 MIME 类型。

9.20.4 ssi_value_length

语法:ssi_value_length length

默认值:ssi_value_length 256

上下文:http, server, location

设置 SSI 命令中参数值的最大长度。。

9.20.5 SSI 命令

SSI 命令的通用格式:

```
<!--# command parameter1=value1 parameter2=value2 ... -->
```

支持的SSI 命令如下:

- `block` —定义一个可在 include 命令中用作存根的块。该块可以包含其他 SSI 命令。该命令有以下参数:
- name —块名称，示例:

```
<!--# block name="one" -->
stub
<!--# endblock -->
```

- `config` —设置SSI处理期间使用的一些参数，即:
- errmsg —t在 SSI 处理期间发生错误时输出的字符串。默认输出以下字符串: “[an error occurred while processing the directive]”
- timefmt —传递给 strftime() 函数的格式化字符串，用于输出日期和时间。默认为以下格式:

```
"%A, %d-%b-%Y %H:%M:%S %Z"
```

%s 格式适合以秒为单位输出时间。

9.21 userid

本模块设置方便客户端识别的 cookie。可以使用内嵌变量 $uid_got 和 $uid_set 记 录已接收和设置的 cookie。该模块与 Apache 的 mod_uid 模块兼容。

示例配置:

```
userid          on;
userid_name     uid;
userid_domain   example.com;
userid_path     /;
userid_expires  365d;
userid_p3p      'policyref="/w3c/p3p.xml", CP="CUR ADM OUR NOR STA NID"';
```

9.21.1 userid

语法:userid [on|v1|log||off]

默认值:userid off

上下文:http, server, location

启用或禁用设置 cookie 和记录接受到的 cookie:

- on - 启用版本 2 cookie 设置并记录接收到的 cookie;
- v1 - 启用版本 1 cookie 设置并记录接收到的 cookie;
- log - 禁用 cookie 设置，但允许记录接收到的 cookie;
- off - 禁用 cookie 设置和记录接收到的 cookie;

9.21.2 userid_domain

语法:userid_domain [name | none]

默认值:userid_domain none

上下文:http, server, location

为设置的 cookie 定义域。none 参数禁用 cookie

9.21.3 userid_expires

语法:userid_expires [time | max| off]

默认值:userid_expires off;

上下文:http, server, location

设置浏览器保留 cookie 的时间（time ）。特殊值 max 将 cookie 设置在 31 Dec 2037 23:55:55 GMT 时 到期。如果未指定参数，cookie 将在浏览器会话结束时到期。

9.21.4 userid_name

语法:userid_name name

默认值:userid_name uid

上下文:http, server, location

如果参数不是 off ，则启用 cookie 标记机制并设置用作标记的字符。此机制用于在保留客户端标识符的同时添 加或更改 userid_p3p 和/或 cookie 的过期时间。标记可以是英文字母（区分大小写）、数字或 = 字符的 任何字符。如果设置了标记，则将其与 cookie 中传递的客户端标识符的 base64 形式中的第一个填充符号进行比较。如果它 们不匹配，则会重新发送带有指定标记、到期时间和 P3P 头的 cookie。

9.21.5 userid_p3p

语法:userid_p3p line

默认值:none

上下文:http, server, location

设置将与 cookie 一起发送的 P3P 头字段的值。如果指令设置为特殊值 none ，则不会在响应中发送 P3P 头。

9.21.6 userid_path

语法:userid_path path

默认值:userid_path /

上下文:http, server, location

为设置的 cookie 定义路径。

9.21.7 userid_service

语法:userid_service number

默认值:userid_service address

上下文:http, server, location

如果标识符由多个服务器（服务）发出，则应为每个服务分配其自己的编号（number ），以确保客户端标识符是唯 一的。对于版本 1 cookie，默认值为零。对于版本 2 cookie，默认值是从服务器 IP 地址的最后四个八位字节 组成的数字。

9.21.8 内嵌变量

http_userid_module 模块支持以下内嵌变量:

$uid_got–cookie 名称和收到的客户端标识符;

$uid_reset–如果变量设置为非空字符串且非“0”，则重置客户端标识符。特殊值 log 会将关于重置标识符的消息输 出到 error_log;

$uid_set cookie –名称和已发送的客户端标识符。

9.22 内置变量

核心模块支持内置的变量，示例：`$http_user_agent`, $http_cookie等等。

此外，还支持其他变量:

9.22.1 $args

请求中的参数值。

9.22.2 $binary_remote_addr

客户端地址的二进制形式，固定长度为4个字节。

9.22.3 $body_bytes_sent

传输给客户端的字节数，响应头不计算在内；这个变量和Apache的mod_log_config模块中的”%B”参数保持兼容。

9.22.4 $content_length

“Content-Length”请求头字段。

9.22.5 $content_type

“Content-Type”请求头字段。

9.22.6 $cookie_name

cookie名称。

9.22.7 $document_root

当前请求的文档根目录或别名。

9.22.8 $document_uri

同 $uri.

9.22.9 $host

优先级：HTTP请求行的主机名>“HOST”请求头字段>符合请求的服务器名.请求中的主机头字段，如果请求中的主机头不可用，则为服务器处理请求的服务器名称。

9.22.10 $is_args

如果请求中有参数，值为”？“，否则为空字符串。

9.22.11 $limit_rate

用于设置响应的速度限制。

9.22.12 $proxy_host

后端工作节点的 IP 地址和端口

9.22.13 $proxy_port

后端工作节点端口，如果未配置端口，则为协议默认端口

9.22.14 $proxy_add_x_forwarded_for

客户端X-Forwarded-For包头值加$remote_addr$,若客户端$X-Forwarded-For$包头值为空，该值和remote_a
相同

9.22.15 $query_string

同 $args.

9.22.16 $remote_addr

客户端地址。

9.22.17 $remote_port

客户端端口。

9.22.18 $remote_user

用于HTTP基础认证服务的用户名。

9.22.19 $request_filename

当前连接请求的文件路径，由root或alias指令与URI请求生成。

9.22.20 $request_body

客户端的请求主体:此变量可在location中使用,将请求主体通过proxy_pass,fastcgi_pass,uwsgi_pass和scgi_pass传递给下一级的代理服务器。

9.22.21 $request_body_file

客户端请求主体保存的临时文件。

9.22.22 $request_completion

如果请求成功，值为”OK”，如果请求未完成或者请求不是一个范围请求的最后一部分，则为空。

9.22.23 $request_method

HTTP请求方法，通常为”GET”或”POST”。

9.22.24 $request_uri

这个变量等于包含一些客户端请求参数的原始URI，它无法修改，请查看$uri更改或重写URI，不包含主机名，示例：“/cnphp/test.php?arg=freemouse”。

9.22.25 $scheme

请求使用的Web协议，“http”或“https”。示例:

```
rewrite  ^(.+)$  $scheme://example.com$1  redirect;
```

9.22.26 $server_addr

服务器端地址，需要注意的是：为了避免访问linux系统内核，应将ip地址提前设置在配置文件中。

9.22.27 $server_name

服务器名。

9.22.28 $server_port

服务器端口。

9.22.29 $server_protocol

服务器的HTTP版本，通常是 `HTTP/1.0` or `HTTP/1.1`。

9.22.30 $uri

请求中的当前URI(不带请求参数，参数位于 `$args`)，可以不同于浏览器传递的$request_uri$的值，它可以通过内部重定向，或者使用$index$指令进行修改，uri不包含主机名, 如”/foo/bar.html”。