Nginx

Nginx反向代理配置指南

Nginx作为反向代理服务器是现代化 Web 架构中的关键组件，它能够高效地处理客户端请求并将其转发到后端服务器。以下是 Nginx 反向代理的详细配置说明。

基础反向代理配置

1. 最简单的反向代理配置

```
server {
    listen 80;
    server_name example.com;

    location / {
        proxy_pass http://localhost:3000;  # 转发到本地的 3000端口
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }
}
```

2. 常用 proxy指令说明

指令	说明
proxy_pass	定义后端服务器地址
proxy_set_header	修改转发给后端服务器的请求头
proxy_redirect	修改后端返回的 Location和 Refresh头
proxy_buffering	控制是否缓冲后端响应
proxy_connect_timeout	连接后端服务器的超时时间
proxy_read_timeout	从后端读取响应的超时时间

高级配置选项

1. 负载均衡配置

```
http {
    upstream backend {
        server backend1.example.com weight=5;
        server backend2.example.com;
        server backend3.example.com backup;

        # 负载均衡方法
        # least_conn;   # 最少连接
        # ip_hash;      # IP哈希
        # random;       # 随机
    }

    server {
        listen 80;
        server_name example.com;

        location / {
            proxy_pass http://backend;
            include proxy_params;  # 包含通用       proxy配置
        }
    }
}
```

2. WebSocket反向代理

```
server {
    listen 80;
    server_name example.com;

    location /ws/ {
        proxy_pass http://websocket_backend;
        proxy_http_version 1.1;
```

```
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_read_timeout 86400;  # WebSocket长连接超时
    }
}
```

3. HTTPS/SSL配置

```
server {
    listen 443 ssl;
    server_name example.com;

    ssl_certificate /path/to/cert.pem;
    ssl_certificate_key /path/to/key.pem;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers HIGH:!aNULL:!MD5;

    location / {
        proxy_pass http://backend;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}

# HTTP重定向到       HTTPS
server {
    listen 80;
    server_name example.com;
    return 301 https://$host$request_uri;
}
```

性能优化配置

1. 缓冲和缓存设置

```
location / {
    proxy_pass http://backend;

    # 缓冲设置
    proxy_buffers 16 32k;
    proxy_buffer_size 64k;
    proxy_busy_buffers_size 128k;

    # 缓存设置
    proxy_cache my_cache;
    proxy_cache_valid 200 302 10m;
    proxy_cache_valid 404 1m;

    # 临时文件
    proxy_temp_file_write_size 64k;
}
```

2. 连接优化

```
location / {
    proxy_pass http://backend;

    # 连接池设置
    keepalive 32;
    keepalive_timeout 30s;
    keepalive_requests 100;

    # 超时设置
    proxy_connect_timeout 5s;
    proxy_send_timeout 10s;
    proxy_read_timeout 30s;
}
```

安全配置

1. 基础安全设置

```
server {
    # ...

    # 隐藏  Nginx版本号
    server_tokens off;

    # 防止点击劫持
    add_header X-Frame-Options "SAMEORIGIN";

    # XSS保护
    add_header X-XSS-Protection "1; mode=block";

    # 内容安全策略
    add_header Content-Security-Policy "default-src 'self'";

    # 禁止不安全的          HTTP方法
    if ($request_method !~ ^(GET|HEAD|POST)$ ) {
        return 405;
    }
}
```

2. 限制访问

```
location /admin/ {
    proxy_pass http://backend;

    # IP限制
    allow 192.168.1.0/24;
    allow 10.0.0.1;
    deny all;

    # 基础认证
    auth_basic "Admin Area";
    auth_basic_user_file /etc/nginx/.htpasswd;

    # 速率限制
    limit_req zone=one burst=10 nodelay;
}
```

常用变量

Nginx反向代理中常用的变量:

变量	说明
$host	原始请求的 Host头
$remote_addr	客户端 IP地址
$proxy_add_x_forwarded_for	追加客户端 IP到 X-Forwarded-For头
$scheme	请求协议 (http或 https)
$request_uri	完整的原始请求 URI
$server_port	服务器端口

调试与日志

1. 访问日志配置

```
http {
    log_format proxy_log '$remote_addr - $remote_user [$time_local] '
                         '"$request" $status $body_bytes_sent '
                         '"$http_referer" "$http_user_agent" '
                         '$upstream_addr $upstream_response_time';

    access_log /var/log/nginx/proxy_access.log proxy_log;
}
```

2. 错误调试

```
location / {
    proxy_pass http://backend;

    # 调试头
    add_header X-Upstream-Addr $upstream_addr;
    add_header X-Cache-Status $upstream_cache_status;
```

```
    # 错误页面
    proxy_intercept_errors on;
    error_page 500 502 503 504 /50x.html;
}
```

最佳实践建议

1. 保持配置模块化：将通用配置放在单独文件（如 `proxy_params`）中，通过 `include` 引入
2. 连接复用：使用 `keepalive` 指令保持与后端的长连接
3. 适当缓冲：根据应用特性调整缓冲设置，大文件上传需要特别配置
4. 监控性能：关注 `$upstream_response_time` 等指标
5. 安全头：始终设置安全相关的 HTTP 头
6. 限制上传大小：对于文件上传应用，设置 `client_max_body_size`

通过以上配置，您可以构建一个高性能、安全可靠的 Nginx 反向代理服务器，满足各种 Web 应用场景的需求。

Nginx proxy_pass URL

规则详解

Nginx 的 `proxy_pass` 指令是反向代理配置中最核心的部分，其 URL 处理规则直接影响请求如何转发到后端服务器。以下是详细的规则解析和配置示例。

基本语法规则

```
location /path/ {
    proxy_pass http://backend;
}
```

1. 以斜杠 (/) 结尾的 proxy_pass

规则：当 `proxy_pass` 的 URL 以斜杠结尾时，Nginx 会将 `location` 匹配的部分从原始 URI 中移除后转发。

```
location /api/ {
    # 请求 /api/user → 转发到 http://backend/user
    proxy_pass http://backend/;
}
```

2. 不以斜杠结尾的 proxy_pass

规则：当 `proxy_pass` 的 URL不以斜杠结尾时，Nginx会将完整的 URI路径传递给后端。

```
location /api/ {
    # 请求 /api/user → 转发到 http://backend/api/user
    proxy_pass http://backend;
}
```

3. 带 URI路径的 proxy_pass

规则：当 `proxy_pass` 包含 URI路径时，Nginx会用指定的路径替换匹配的 `location` 部分。

```
location /old/ {
    # 请求 /old/page → 转发到 http://backend/new/page
    proxy_pass http://backend/new/;
}
```

特殊转发场景

1. 正则表达式 location中的 proxy_pass

```
location ~ ^/user/(\d+) {
    # 请求 /user/123 → 转发到 http://backend/profile/123
    proxy_pass http://backend/profile/$1;
}
```

2. 重写 URI后再转发

```
location /shop/ {
    rewrite ^/shop/(.*)$ /store/$1 break;
    # □□ /shop/item → □□□ http://backend/store/item
    proxy_pass http://backend;
}
```

3. □□□□□□□□□ URL

```
location / {
    # □□□□□□□□
    proxy_pass http://backend$request_uri;
}
```

□□□ proxy_pass□□□□

1. □□ $scheme□□

```
location / {
    # □□□□ http□ https□□
    proxy_pass $scheme://backend;
}
```

2. □□ $host□□

```
location / {
    # □□□□ Host□
    proxy_pass http://backend/$host$uri;
}
```

□□□□□□

1. WebSocket□□

```
location /ws/ {
    proxy_pass http://backend;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
}
```

2. HTTPS后端

```
location / {
    proxy_pass https://backend;
    proxy_ssl_verify off;  # 不验证后端证书
}
```

常见问题解决方案

1. 尾部斜杠问题

问题：访问 `/api` 和 `/api/` 结果不同

解决：

```
location /api {
    # 统一处理带 /和不带 /的请求
    proxy_pass http://backend/api/;
}
```

2. 路径重复问题

问题：转发后路径中出现重复部分如 `/api/api`

解决：

```
location /api/ {
    # 确保 proxy_pass以 /结尾
    proxy_pass http://backend/;
```

```
}
```

3. 特殊字符编码问题

问题：URL中的特殊字符被编码

解决：

```
location / {
    proxy_pass http://backend;
    proxy_redirect off;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
}
```

最佳实践建议

1. 一致性：统一在 `proxy_pass` 中使用或不使用结尾斜杠
2. 测试：使用 `curl -v` 测试转发后的实际URL
3. 日志：检查Nginx访问日志中的 `$upstream_addr` 和 `$request_uri`
4. 变量使用：谨慎使用变量，确保不会构造出无效URL
5. 编码处理：注意URL编码问题，特别是包含中文等非ASCII字符时

通过理解这些规则，您可以精确控制Nginx如何将请求转发到后端服务器，满足各种复杂的代理需求。

rocky 安装 Nginx和基本操作

在 Rocky Linux 上安装 Nginx 的步骤如下:

1. 更新系统

确保系统是最新的:

```
sudo dnf update -y
```

2. 安装 Nginx

Rocky Linux 默认仓库包含 Nginx, 可以直接安装:

```
sudo dnf install nginx -y
```

3. 启动并设置开机自启

```
sudo systemctl start nginx
sudo systemctl enable nginx
```

4. 检查 Nginx 状态

```
sudo systemctl status nginx
```

如果看到 `active (running)`, 说明 Nginx 已成功运行。

5. 配置防火墙（如果启用）

允许 HTTP (80) 和 HTTPS (443) 流量:

```
sudo firewall-cmd --permanent --add-service=http
sudo firewall-cmd --permanent --add-service=https
```

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

6. 测试访问

在浏览器输入服务器的 IP 地址，应该能看到 Nginx 默认欢迎页面：

```
http://你的服务器 IP
```

（可选）调整 Nginx 配置

主配置文件位于 `/etc/nginx/nginx.conf`，网站配置文件通常在 `/etc/nginx/conf.d/` 或 `/etc/nginx/sites-available/`（需手动创建）。

修改配置后，测试并重载 Nginx：

```
sudo nginx -t  # 检查配置语法
sudo systemctl reload nginx  # 重新加载配置
```

这样，Nginx 就已经在 Rocky Linux 上安装并运行了！

Nginx 基本操作

1. 启动、停止、重启 Nginx

```
sudo systemctl start nginx     # 启动
sudo systemctl stop nginx      # 停止
sudo systemctl restart nginx   # 重启（强制重新加载）
sudo systemctl reload nginx    # 平滑重载（不中断服务，仅重新加载配置）
```

2. 查看 Nginx 状态

```
sudo systemctl status nginx    # 查看运行状态
sudo nginx -t                  # 检查配置文件语法（测试配置是否正确）
```

3. 设置开机自启

```
sudo systemctl enable nginx    # 开机自动启动
sudo systemctl disable nginx   # 取消开机自启
```

4. 查看 Nginx 版本

```
nginx -v      # 查看版本
nginx -V      # 查看详细版本及编译参数
```

5. 检查 Nginx 进程

```
ps aux | grep nginx  # 查看 Nginx 进程
```

6. 日志查看

Nginx 日志默认位置:

- 访问日志: `/var/log/nginx/access.log`
- 错误日志: `/var/log/nginx/error.log`

实时查看日志:

```
sudo tail -f /var/log/nginx/access.log  # 实时监控访问日志
sudo tail -f /var/log/nginx/error.log  # 实时监控错误日志
```

7. 配置文件管理

- 主配置文件: `/etc/nginx/nginx.conf`
- 站点配置: 通常在 `/etc/nginx/conf.d/` 或 `/etc/nginx/sites-available/` (需手动创建)

修改配置后，测试并重载:

```
sudo nginx -t            # 测试配置是否正确
sudo systemctl reload nginx  # 重新加载配置（不中断服务）
```

8. 虚拟主机（Server Block）配置示例

在 `/etc/nginx/conf.d/example.conf` 添加:

```
server {
    listen 80;
    server_name example.com www.example.com;
    root /var/www/example;
    index index.html;

    location / {
```

```
        try_files $uri $uri/ =404;
    }
}
```

然后测试并重载：

```
sudo nginx -t && sudo systemctl reload nginx
```

9. 卸载 Nginx

```
sudo dnf remove nginx -y    # 卸载 Nginx
sudo rm -rf /etc/nginx/     # 删除配置文件（谨慎操作！）
```

掌握这些基本操作，就可以管理 Nginx 服务了！ □□

Nginx [illegible]

Nginx
[illegible]
Nginx [illegible]

1. [illegible] logrotate ([illegible])

logrotate [illegible] Linux [illegible] Nginx [illegible]

[illegible]

1. [illegible] logrotate [illegible] ([illegible] `/etc/logrotate.d/nginx`)

```
sudo nano /etc/logrotate.d/nginx
```

2. [illegible]

```
/var/log/nginx/*.log {
    daily
    missingok
    rotate 14
    compress
    delaycompress
    notifempty
    create 0640 www-data adm
    sharedscripts
    postrotate
        if [ -f /var/run/nginx.pid ]; then
            kill -USR1 `cat /var/run/nginx.pid`
        fi
    endscript
}
```

[illegible]

- `daily`：每天轮转
- `missingok`：如果日志文件不存在也不报错
- `rotate 14`：保留14个旧日志文件
- `compress`：使用gzip压缩旧日志
- `delaycompress`：延迟压缩前一个日志文件
- `notifempty`：如果日志为空则不轮转
- `create 0640 www-data adm`：创建新日志文件的权限和所有者
- `sharedscripts`：所有日志处理完后才运行postrotate脚本
- `postrotate`：通知Nginx重新打开日志文件

3. 测试配置是否正确

```
sudo logrotate -d /etc/logrotate.d/nginx
```

4. 手动运行logrotate

```
sudo logrotate -f /etc/logrotate.d/nginx
```

2. 使用cron脚本手动分割

1. 创建日志分割脚本`/usr/local/bin/nginx_logrotate.sh`：

```
#!/bin/bash
# Nginx 日志目录
LOGS_PATH=/var/log/nginx
# 获取昨天日期
YESTERDAY=$(date -d "yesterday" +%Y-%m-%d)
# 重命名日志文件
mv ${LOGS_PATH}/access.log ${LOGS_PATH}/access_${YESTERDAY}.log
mv ${LOGS_PATH}/error.log ${LOGS_PATH}/error_${YESTERDAY}.log
# 向 Nginx 主进程发送 USR1 信号，重新打开日志文件
kill -USR1 $(cat /var/run/nginx.pid)
```

2. 给脚本执行权限

```
chmod +x /usr/local/bin/nginx_logrotate.sh
```

3. 设置cron任务每天执行

```
crontab -e
```

添加以下内容（每天午夜执行）：

```
0 0 * * * /usr/local/bin/nginx_logrotate.sh
```

3. 使用系统自带的时间戳日志 (Nginx 1.19.6+)

Nginx 1.19.6 及以上版本支持在日志文件名中使用变量：

```
http {
    log_format custom '$remote_addr - $remote_user [$time_local] '
                      '"$request" $status $body_bytes_sent '
                      '"$http_referer" "$http_user_agent"';

    access_log /var/log/nginx/access-${year}${month}${day}.log custom;
}
```

然后使用 `map` 指令定义时间变量：

```
map $time_iso8601 $year {
    default '0000';
    '~^(?<yyyy>\d{4})-' $yyyy;
}

map $time_iso8601 $month {
    default '00';
    '~^\d{4}-(?<mm>\d{2})-' $mm;
}

map $time_iso8601 $day {
    default '00';
    '~^\d{4}-\d{2}-(?<dd>\d{2})' $dd;
}
```

日志分割后的管理

1. 删除旧日志：定期清理过期的日志文件
2. 日志分析：使用工具如 GoAccess、AWStats 分析分割后的日志
3. 日志备份：重要的日志文件可以备份到远程存储

注意事项

1. 确保 Nginx 进程有权限写入新的日志文件
2. 分割日志时确保磁盘空间充足
3. 生产环境建议使用 logrotate 方案，它更可靠且功能完善
4. 对于高流量网站，可以考虑按小时分割日志

以上方法可以根据实际需求选择或组合使用，logrotate 通常是大多数 Linux 发行版的首选方案。