502 Bad Gateway 排查命令清单

> 一份来自15年运维老兵的实战手册 | 适用于 Nginx/Apache + PHP/Python/Java 后端架构

一、排查总览（思维导图）

```
502 Bad Gateway
├── 1. 网络层（连通性/DNS/防火墙）
├── 2. 系统层（CPU/内存/磁盘/连接数/文件句柄）
├── 3. 负载均衡层（LB/反向代理健康检查）
├── 4. Web 服务层（Nginx/Apache 配置与状态）
├── 5. 后端应用层（PHP-FPM/Tomcat/Node/上游服务）
├── 6. 数据库层（慢查询/连接池/死锁）
├── 7. 配置层（超时/缓冲区/代理配置）
└── 8. 日志分析（error.log/access.log/slowlog）
```

> **核心理念**：从外到内、从下到上、先软后硬、先查后改。

二、第一层：网络层排查（5分钟快速定位）

2.1 验证基础连通性

```
# 1) ping 测试基础网络（注意：生产环境很多禁ping，仅作参考）
sshmcp -h prod-web-01 -u root -c "ping -c 4 backend-app-01"

# 2) traceroute 路由追踪
sshmcp -h prod-web-01 -u root -c "traceroute -n -T -p 9000 backend-app-01"

# 3) telnet 测试后端端口（最常用）
sshmcp -h prod-web-01 -u root -c "telnet backend-app-01 9000"
# 或使用更现代的 nc
sshmcp -h prod-web-01 -u root -c "nc -zv backend-app-01 9000 -w 5"

# 4) 批量测试后端端口（脚本化）
sshmcp -h prod-web-01 -u root -c "for host in backend-app-01 backend-app-02 backend-app-03;
do echo -n \"\$host:9000 -> \"; timeout 3 bash -c \"</dev/tcp/\$host/9000\" 2>&1 && echo OK
|| echo FAIL; done"
```

2.2 DNS 解析检查

```
# 1) 检查 upstream 域名解析是否正确
sshmcp -h prod-web-01 -u root -c "nslookup backend.internal.com"
sshmcp -h prod-web-01 -u root -c "dig backend.internal.com +short"

# 2) 查看 /etc/hosts 是否被改动
sshmcp -h prod-web-01 -u root -c "cat /etc/hosts | grep -v '^#' | grep -v '^$'"

# 3) 检查 systemd-resolved（Ubuntu/Debian）
sshmcp -h prod-web-01 -u root -c "systemd-resolve --status | head -30"
```

2.3 防火墙与安全组

```
# 1) iptables 规则检查（看是否被拦截）
sshmcp -h prod-web-01 -u root -c "iptables -L -n -v | head -50"

# 2) 检查是否有 DROP/REJECT 规则命中
sshmcp -h prod-web-01 -u root -c "iptables -L -n -v | awk '\$1 ~ /^[0-9]+$/ && \$2 > 1000
{print}'"

# 3) firewalld 区域规则（CentOS/RHEL 7+）
sshmcp -h prod-web-01 -u root -c "firewall-cmd --list-all"

# 4) 查看连接跟踪表是否爆满
sshmcp -h prod-web-01 -u root -c "cat /proc/sys/net/netfilter/nf_conntrack_max"
sshmcp -h prod-web-01 -u root -c "cat /proc/sys/net/netfilter/nf_conntrack_count"
```

三、第二层：系统层排查（资源瓶颈）

3.1 CPU 使用率

```
# 1) 实时查看 CPU 负载
sshmcp -h prod-web-01 -u root -c "top -bn1 | head -20"

# 2) 找出 CPU 最高的进程
sshmcp -h prod-web-01 -u root -c "ps -eo pid,ppid,%cpu,%mem,cmd --sort=-%cpu | head -20"

# 3) 定位 CPU 高的线程（Java/PHP 必备）
sshmcp -h prod-web-01 -u root -c "top -H -p <PID>"

# 4) 查看系统平均负载（Load Average）
sshmcp -h prod-web-01 -u root -c "uptime"
sshmcp -h prod-web-01 -u root -c "cat /proc/loadavg"

# 5) CPU 核心数参考
sshmcp -h prod-web-01 -u root -c "nproc"
```

3.2 内存与 Swap

```
# 1) 内存使用概览
sshmcp -h prod-web-01 -u root -c "free -h"

# 2) 内存详情（看 cached/available）
sshmcp -h prod-web-01 -u root -c "cat /proc/meminfo"

# 3) 找出吃内存最多的进程
sshmcp -h prod-web-01 -u root -c "ps -eo pid,ppid,%mem,rss,cmd --sort=-%mem | head -20"

# 4) Swap 使用情况
sshmcp -h prod-web-01 -u root -c "swapon --show"
sshmcp -h prod-web-01 -u root -c "vmstat 1 5"
```

3.3 磁盘 I/O 与空间

```
# 1) 磁盘空间（502的常见元凶：日志/临时文件把磁盘写满）
sshmcp -h prod-web-01 -u root -c "df -h"
sshmcp -h prod-web-01 -u root -c "df -i"   # inode 用尽也会502

# 2) 找出大文件
sshmcp -h prod-web-01 -u root -c "du -sh /* 2>/dev/null | sort -hr | head -20"

# 3) 磁盘 I/O 性能
sshmcp -h prod-web-01 -u root -c "iostat -xm 1 5"
sshmcp -h prod-web-01 -u root -c "iotop -b -n 3 -d 2"

# 4) 找出大量 I/O 的进程
sshmcp -h prod-web-01 -u root -c "ps aux | awk '\$4 > 50 {print}'"
```

3.4 文件句柄与连接数

```
# 1) 系统级句柄使用（502的隐形杀手）
sshmcp -h prod-web-01 -u root -c "cat /proc/sys/fs/file-nr"
# 输出说明：已分配/已使用/上限，例如：1024  5000  100000

# 2) 进程级句柄数（按进程）
sshmcp -h prod-web-01 -u root -c "ls /proc/<PID>/fd | wc -l"

# 3) Nginx 进程句柄数
sshmcp -h prod-web-01 -u root -c "for pid in \$(pidof nginx); do echo \"nginx pid=\$pid
fds=\$(ls /proc/\$pid/fd 2>/dev/null | wc -l)\"; done"

# 4) TCP 连接状态统计
sshmcp -h prod-web-01 -u root -c "ss -s"
sshmcp -h prod-web-01 -u root -c "ss -tan | awk 'NR>1 {print \$1}' | sort | uniq -c | sort -
rn"

# 5) 重点关注 TIME_WAIT 数量（连接打满的标志）
```

```
sshmcp -h prod-web-01 -u root -c "ss -tan | grep -c TIME_WAIT"

# 6) 半连接/全连接队列
sshmcp -h prod-web-01 -u root -c "ss -lnt | head -20"
sshmcp -h prod-web-01 -u root -c "netstat -s | grep -i 'listen\|overflow\|drop'"
```

四、第三层：Web 服务层排查（Nginx 重点）

4.1 Nginx 服务状态

```
# 1) Nginx 进程与版本
sshmcp -h prod-web-01 -u root -c "ps aux | grep nginx | grep -v grep"
sshmcp -h prod-web-01 -u root -c "nginx -v"

# 2) 测试配置语法
sshmcp -h prod-web-01 -u root -c "nginx -t"

# 3) 当前连接状态
sshmcp -h prod-web-01 -u root -c "ss -tan | grep ':80\\|:443' | wc -l"

# 4) Nginx 自带的 stub_status（需先开启配置）
sshmcp -h prod-web-01 -u root -c "curl -s http://127.0.0.1/nginx_status"
# 关注 Active connections、Reading/Writing 数量
```

4.2 Nginx 错误日志（定位 502 的金矿）

```
# 1) 实时跟踪错误日志
sshmcp -h prod-web-01 -u root -c "tail -f /var/log/nginx/error.log"

# 2) 过滤 502 相关错误
sshmcp -h prod-web-01 -u root -c "grep -i '502\\|connect() failed\\|upstream timed out\\|no
live upstreams' /var/log/nginx/error.log | tail -50"

# 3) 统计各 upstream 失败次数
sshmcp -h prod-web-01 -u root -c "grep 'connect() failed' /var/log/nginx/error.log | awk -
F'peer:' '{print \$2}' | awk '{print \$1}' | sort | uniq -c | sort -rn"

# 4) 统计各类错误码出现次数
sshmcp -h prod-web-01 -u root -c "awk '\$9 ~ /50[0-9]/ {print \$9}'
/var/log/nginx/access.log | sort | uniq -c | sort -rn"

# 5) 查找超时错误
sshmcp -h prod-web-01 -u root -c "grep -i 'upstream timed out' /var/log/nginx/error.log |
tail -20"
```

4.3 关键配置项检查（最容易踩坑的）

```
# 查看 nginx.conf 及虚拟主机配置
sshmcp -h prod-web-01 -u root -c "cat /etc/nginx/nginx.conf"
sshmcp -h prod-web-01 -u root -c "cat /etc/nginx/conf.d/*.conf"
sshmcp -h prod-web-01 -u root -c "cat /etc/nginx/vhost/*.conf"

# 重点检查以下指令：
#   - proxy_pass http://upstream_name;
#   - upstream backend { server 127.0.0.1:9000; keepalive 1024; }
#   - proxy_connect_timeout 60s;
#   - proxy_send_timeout 60s;
#   - proxy_read_timeout 60s;
#   - proxy_buffer_size / proxy_buffers
#   - proxy_next_upstream error timeout http_502 http_503 http_504;
#   - proxy_next_upstream_tries 3;
```

4.4 upstream 健康状态

```
# 1) 通过 Nginx Plus 或 Tengine 查看 upstream 状态
sshmcp -h prod-web-01 -u root -c "curl -s http://127.0.0.1/upstream_check"

# 2) 自定义健康检查配置示例
sshmcp -h prod-web-01 -u root -c "grep -A 10 'check' /etc/nginx/conf.d/upstream.conf"

# 3) 临时手动验证 upstream 节点
sshmcp -h prod-web-01 -u root -c "for ip in 10.0.1.11 10.0.1.12 10.0.1.13; do echo -n
\"\$ip:9000 -> \"; curl -s -o /dev/null -w '%{http_code} %{time_total}s\\n' --max-time 5
http://\$ip:9000/health; done"
```

五、第四层：后端应用层排查

5.1 PHP-FPM（最常见的 502 源头）

```
# 1) PHP-FPM 进程状态
sshmcp -h prod-web-01 -u root -c "ps aux | grep php-fpm | grep -v grep | wc -l"
sshmcp -h prod-web-01 -u root -c "ps -eo pid,ppid,%cpu,%mem,cmd --sort=-%cpu | grep php-fpm
| head -10"

# 2) PHP-FPM 错误日志（最直接的信息源）
sshmcp -h prod-web-01 -u root -c "tail -100 /var/log/php-fpm/error.log"
sshmcp -h prod-web-01 -u root -c "tail -100 /var/log/php7.4-fpm.log"
sshmcp -h prod-web-01 -u root -c "grep -i 'error\\|warning\\|fatal\\|segfault' /var/log/php-
fpm/error.log | tail -30"

# 3) PHP-FPM 慢日志（看哪些请求卡住了）
sshmcp -h prod-web-01 -u root -c "cat /var/log/php-fpm/www-slow.log 2>/dev/null | tail -50"

# 4) PHP-FPM 状态页（关键指标）
```

```
sshmcp -h prod-web-01 -u root -c "curl -s http://127.0.0.1:9000/fpm-status?full"
sshmcp -h prod-web-01 -u root -c "curl -s http://127.0.0.1:9000/fpm-status?json"
# 重点关注：
#   - active processes（活跃进程数）
#   - idle processes（空闲进程数）
#   - listen queue（监听队列，>0 表示已排队）
#   - max active processes（峰值活跃数）
#   - max children reached（达到上限次数 > 0 说明 pm.max_children 偏小）

# 5) PHP-FPM 关键配置检查
sshmcp -h prod-web-01 -u root -c "cat /etc/php-fpm.d/www.conf | grep -E
'pm\\s*=|pm.max_children|pm.start_servers|pm.min_spare_servers|pm.max_spare_servers|pm.max_r
equests|request_terminate_timeout|listen'"

# 6) 监听 socket 检查
sshmcp -h prod-web-01 -u root -c "ss -lntp | grep php-fpm"
sshmcp -h prod-web-01 -u root -c "ls -la /run/php-fpm/ 2>/dev/null"
sshmcp -h prod-web-01 -u root -c "ls -la /tmp/php-fpm.sock 2>/dev/null"

# 7) 紧急查看 PHP-FPM 是否在运行
sshmcp -h prod-web-01 -u root -c "systemctl status php-fpm"
sshmcp -h prod-web-01 -u root -c "systemctl status php7.4-fpm"
```

5.2 Tomcat / Java 应用

```
# 1) JVM 堆内存使用
sshmcp -h prod-app-01 -u root -c "jstat -gc <PID> 1000 5"

# 2) 线程状态
sshmcp -h prod-app-01 -u root -c "jstack <PID> | grep -A 2 'BLOCKED\\|WAITING' | head -50"

# 3) 找出 CPU 高的 Java 线程
sshmcp -h prod-app-01 -u root -c "top -H -p <PID>"
# 然后将线程 PID 转为 16 进制
sshmcp -h prod-app-01 -u root -c "printf '%x\\n' <线程PID>"
# 在 jstack 输出中搜索该十六进制
sshmcp -h prod-app-01 -u root -c "jstack <PID> | grep -A 30 'nid=0x<十六进制>'"

# 4) Tomcat 日志
sshmcp -h prod-app-01 -u root -c "tail -200 /opt/tomcat/logs/catalina.out"
sshmcp -h prod-app-01 -u root -c "grep -i 'exception\\|error' /opt/tomcat/logs/catalina.out
| tail -30"

# 5) GC 情况
sshmcp -h prod-app-01 -u root -c "jstat -gcutil <PID> 1000 5"
```

5.3 Python (Gunicorn / uWSGI)

```
# 1) Gunicorn 进程
sshmcp -h prod-app-01 -u root -c "ps aux | grep gunicorn | grep -v grep"

# 2) Gunicorn 配置
sshmcp -h prod-app-01 -u root -c "grep -E 'workers|timeout|keepalive'
/etc/gunicorn/gunicorn.conf.py"

# 3) uWSGI 检查
sshmcp -h prod-app-01 -u root -c "ps aux | grep uwsgi | grep -v grep"
sshmcp -h prod-app-01 -u root -c "tail -100 /var/log/uwsgi/app.log"
```

5.4 Node.js 应用

```
# 1) Node 进程与内存
sshmcp -h prod-app-01 -u root -c "ps aux | grep node | grep -v grep"

# 2) 内存泄漏排查
sshmcp -h prod-app-01 -u root -c "node --inspect=0.0.0.0:9229 server.js"
# 然后用 Chrome DevTools 连接

# 3) PM2 进程列表
sshmcp -h prod-app-01 -u root -c "pm2 list"
sshmcp -h prod-app-01 -u root -c "pm2 logs --lines 200"
```

六、第五层：数据库层排查

> 数据库慢查询/连接池耗尽是 502 的隐形推手。

```
# 1) MySQL 连接数
sshmcp -h prod-db-01 -u root -c "mysql -e 'show status like \"Threads_connected\";'"
sshmcp -h prod-db-01 -u root -c "mysql -e 'show variables like \"max_connections\";'"
sshmcp -h prod-db-01 -u root -c "mysql -e 'show processlist;' | wc -l"

# 2) 慢查询
sshmcp -h prod-db-01 -u root -c "mysql -e 'show global status like \"Slow_queries\";'"
sshmcp -h prod-db-01 -u root -c "tail -100 /var/log/mysql/slow.log"

# 3) 当前锁等待
sshmcp -h prod-db-01 -u root -c "mysql -e 'SELECT * FROM information_schema.INNODB_TRX;'"
sshmcp -h prod-db-01 -u root -c "mysql -e 'SELECT * FROM
information_schema.INNODB_LOCK_WAITS;'"

# 4) Redis 连接与慢命令
sshmcp -h prod-db-01 -u root -c "redis-cli info clients"
sshmcp -h prod-db-01 -u root -c "redis-cli slowlog get 20"
sshmcp -h prod-db-01 -u root -c "redis-cli config get maxclients"

# 5) 网络连接到 DB
```

```
sshmcp -h prod-web-01 -u root -c "nc -zv prod-db-01 3306 -w 3"
```

七、第六层：连接数与并发瓶颈（高级排查）

```
# 1) 系统最大连接数
sshmcp -h prod-web-01 -u root -c "cat /proc/sys/net/core/somaxconn"
# 默认 128，Nginx 高并发必须调大

# 2) 端口范围
sshmcp -h prod-web-01 -u root -c "cat /proc/sys/net/ipv4/ip_local_port_range"

# 3) TCP 各项参数
sshmcp -h prod-web-01 -u root -c "sysctl -a | grep -E
'tcp_tw_reuse|tcp_tw_recycle|tcp_fin_timeout|tcp_max_syn_backlog|tcp_keepalive_time'"

# 4) 实时查看哪些 IP 在连接（被攻击的征兆）
sshmcp -h prod-web-01 -u root -c "ss -tan | awk 'NR>1 {print \$5}' | cut -d: -f1 | sort |
uniq -c | sort -rn | head -20"

# 5) 查看连接状态详细分布
sshmcp -h prod-web-01 -u root -c "ss -tan | awk 'NR>1 {print \$1}' | sort | uniq -c"
# ESTABLISHED 过大、CLOSE_WAIT 过多、TIME_WAIT 爆炸，都要警惕

# 6) 内核丢包统计
sshmcp -h prod-web-01 -u root -c "netstat -s | grep -E 'listen queue|overflow|drop|pruned'"
```

八、第七层：日志聚合分析（黄金信息源）

8.1 实时抓取关键错误

```
# 1) 实时跟踪所有相关日志（开多个窗口）
sshmcp -h prod-web-01 -u root -c "tail -F /var/log/nginx/error.log /var/log/php-
fpm/error.log"

# 2) 实时过滤 502 错误
sshmcp -h prod-web-01 -u root -c "tail -F /var/log/nginx/error.log | grep --line-buffered -E
'502|upstream|connect'"

# 3) 找出触发 502 的具体 URL
sshmcp -h prod-web-01 -u root -c "awk '\$9==502 {print \$7}' /var/log/nginx/access.log |
sort | uniq -c | sort -rn | head -20"

# 4) 找出触发 502 的客户端 IP
sshmcp -h prod-web-01 -u root -c "awk '\$9==502 {print \$1}' /var/log/nginx/access.log |
sort | uniq -c | sort -rn | head -20"
```

8.2 日志分析常用套路

```
# 1) 错误码趋势（看是从哪个时间点开始 502 的）
sshmcp -h prod-web-01 -u root -c "awk '\$9==502 {print \$4}' /var/log/nginx/access.log | cut -d: -f1,2 | uniq -c"

# 2) 找出最耗时的请求（往往这些会拖垮后端）
sshmcp -h prod-web-01 -u root -c "awk '\$NF > 5 {print \$NF, \$7}' /var/log/nginx/access.log | sort -rn | head -20"
# 注：需要在 log_format 中加 $request_time

# 3) 配合日志分析工具
sshmcp -h prod-web-01 -u root -c "goaccess -f /var/log/nginx/access.log -o /tmp/report.html --log-format=COMBINED"
```

九、应急处置流程（高风险操作，需确认）

> **以下操作可能影响生产，请确认无误后执行！**

9.1 重启服务（按风险从低到高）

```
# 1) 优雅重载 Nginx（不中断连接）
sshmcp -h prod-web-01 -u root -c "nginx -s reload"

# 2) 重启 PHP-FPM（会断开现有请求）
sshmcp -h prod-web-01 -u root -c "systemctl restart php-fpm"

# 3) 重启 Nginx
sshmcp -h prod-web-01 -u root -c "systemctl restart nginx"
```

9.2 临时缓解措施

```
# 1) 紧急清理磁盘（先确认能删）
sshmcp -h prod-web-01 -u root -c "find /var/log/nginx -name '*.gz' -mtime +7 -delete"
sshmcp -h prod-web-01 -u root -c "journalctl --vacuum-time=3d"

# 2) 清理 PHP-FPM 慢日志
sshmcp -h prod-web-01 -u root -c "> /var/log/php-fpm/www-slow.log"

# 3) 临时扩大 PHP-FPM 子进程数（应急）
sshmcp -h prod-web-01 -u root -c "sed -i 's/pm.max_children = .*/pm.max_children = 200/' /etc/php-fpm.d/www.conf && systemctl reload php-fpm"
```

9.3 临时摘除故障节点

```
# 1) 在 upstream 中注释故障节点
sshmcp -h prod-web-01 -u root -c "sed -i 's/^    server 10.0.1.13/# server 10.0.1.13  # 临时下线/' /etc/nginx/conf.d/backend.conf && nginx -s reload"

# 2) 通过 iptables 临时阻断（慎用）
sshmcp -h prod-web-01 -u root -c "iptables -I INPUT -s 10.0.1.13 -j DROP"
```

十、常见 502 错误对照表

错误关键字（error.log）	根本原因	解决方案
`connect() failed (111: Connection refused)`	后端服务没启动 / 端口错	启动后端 / 检查端口
`connect() failed (113: No route to host)`	网络不通	检查防火墙 / 安全组
`upstream timed out (110: Connection timed out)`	后端响应慢 / 网络问题	调大超时 / 优化后端
`no live upstreams`	upstream 全部 down	检查所有后端
`while connecting to upstream`	Socket 路径错误	检查 fastcgi_pass / proxy_pass
`recv() failed (104: Connection reset by peer)`	后端主动断开	检查后端进程 / OOM
`upstream prematurely closed connection`	后端没发完响应就关闭	检查 PHP 致命错误 / 段错误
`504 Gateway Time-out`	后端处理超时	优化 SQL / 增加 worker

十一、预防性监控（建议长期落地）

```
# 1) Nginx 状态页 + 监控（zabbix/prometheus）
sshmcp -h prod-web-01 -u root -c "cat /etc/nginx/conf.d/status.conf"
# location /nginx_status { stub_status on; access_log off; allow 127.0.0.1; deny all; }

# 2) PHP-FPM 状态页
sshmcp -h prod-web-01 -u root -c "cat /etc/php-fpm.d/www.conf | grep pm.status_path"

# 3) 推荐告警阈值
# - Nginx active connections > 80% worker_connections
# - listen queue > 10 持续 1 分钟
# - listen queue len overflow > 0
# - PHP-FPM active processes = max_children 持续 30s
# - 502 错误率 > 0.5% 持续 1 分钟
```

```
# - TCP TIME_WAIT > 30000
# - 磁盘使用率 > 85%
# - inode 使用率 > 85%
```

十二、排查流程图（速记版）

```
┌──────────────────────────────────────────┐
│  用户反馈 502 或监控告警                  │
└──────────────┬───────────────────────────┘
               ↓
┌──────────────────────────────────────────┐
│ ① 5分钟内：自测访问？影响全部还是部分？    │
│   - curl -I http://domain                │
│   - 看监控 502 错误率走势                 │
└──────────────┬───────────────────────────┘
               ↓
┌──────────────────────────────────────────┐
│ ② nginx error.log 必看（10秒定位）        │
│   - connect refused → 后端挂了            │
│   - timed out → 后端慢                    │
│   - no live upstreams → 全挂              │
└──────────────┬───────────────────────────┘
               ↓
┌──────────────────────────────────────────┐
│ ③ 验证后端存活                            │
│   - telnet/nc 后端 IP:端口                │
│   - 进程在不在？日志有无报错？             │
└──────────────┬───────────────────────────┘
               ↓
┌──────────────────────────────────────────┐
│ ④ 系统资源？连接数？句柄？                │
│   - top / free / df / ss -s / file-nr    │
└──────────────┬───────────────────────────┘
               ↓
┌──────────────────────────────────────────┐
│ ⑤ 定位后，紧急止血 → 根因分析 → 永久修复   │
└──────────────────────────────────────────┘
```

附录：推荐排查工具清单

工具	用途	安装
`ss / netstat`	网络连接	自带
`iproute2`	高级网络	`yum install iproute`
`htop`	增强 top	`yum install htop`
`glances`	全能监控	`pip install glances`

工具	用途	安装
`iotop`	磁盘 I/O	`yum install iotop`
`tcpdump`	抓包分析	`yum install tcpdump`
`strace`	跟踪进程系统调用	`yum install strace`
`lsof`	文件/网络句柄	`yum install lsof`
`goaccess`	日志可视化	`yum install goaccess`
`perf`	性能剖析	`yum install perf`

> **最后忠告**：502 的本质是「上游不可用」，**先恢复后调查**。任何重启之前，**先抓现场**（错误日志、状态、配置），不然修好了也不知道根因。