　　一、日志的概念

　　什么是日志？日志就是历史事件。历史事件包括时间、地点、人物、时间。这个是生活中所说的日志很好理解。在Linux中也有类似的服务，它主要作用就是记录Linux系统的历史事件，包括什么时间什么服务或者那个进程或者pid发生的一些事件，通过记录发生的事件，我们可以查看日志来了解在过去的一段时间Linux系统发生了什么事，从而可以帮助我们解决一些问题。

　　在Linux系统里日志是有级别的，也就是说事件的关键程度，比如说有些事件只是警告，需要我们注意，起个提醒我们的目的，我们可以后面去处理，也可以不处理，但是有些事件级别比较紧急，它不仅仅只是提示我们的作用，很有可能这一秒发生了这样的事件，下一秒Linux系统就挂了，所以在Linux系统里事件的关键性程度非常重要。在centos5之前日志系统的名称叫syslog，它主要有两个服务组成，一个是syslogd（system application ）它主要记录着应用程序的一些日志，一个是klogd（Linux kernel）它主要记录着Linux内核的日志。通常记录事件的格式是，日期时间  主机   进程[pid]  事件内容。Linux日志系统不仅仅可以用做本地记录本机的日志，它还可以通过tcp或者udp协议的服务完成日志的传送，从而实现帮助其他主机记录日志功能，我们把这样的服务器称为日志服务器。

　　二、rsyslog介绍

　　在centos6和centos7上rsyslog有如下特性

　　1）多进程

　　2）支持UDP、TCP、SSL、TLS、RELP等协议

　　3）可以通过网络将日志储存到Mysql、PGSQL、Oracle等数据库中管理

　　4）支持强大的过滤器，可实现过滤记录日志信息中的任意部分

　　5）支持自定义日志输出格式

　　rsyslog日志收集器重要术语

　　facility:中文翻译过来是设施的意思，从功能或程序上对日志分类，在Linux中常见的facility有auth（认证相关的日志），authpriv（授权相关的日志），cron（计划任务相关日志），daemon（系统服务相关日志），ftp（ftp服务相关的日志），kern（内核相关日志），lpr（打印相关的日志），mail（邮件相关日志），news（新闻相关的日志），security（安全相关的日志），user（用户相关的日志），uucp（文件copy相关的日志），local0-local7（自定义相关的日志）

　　priority：优先级别，从低到高排序debug（调试），info（消息），notice（注意），warn（warning警告），err（error错误），crit（critical严重警告），alert（需要立即修改的信息）emerg（panic内核崩溃，内核恐慌等严重的信息）

　　程序环境：

　　　　程序包：rsyslog

　　　　主程序：/usr/sbin/rsyslogd

　　　　主配置文件：/etc/rsyslog.conf,/etc/rsyslog.d/\*.conf

　　　　库文件：/lib64/rsyslog/\*.so

　　　　服务脚本：

　　　　　　centos6：service rsyslog {start|stop|restart|status}

　　　　　　centos7:/usr/lib/systemd/system/rsyslog.service

　　　　配置文件格式：由三部分组成

　　　　　　MODULES：相关模块配置

　　　　　　GLOBAL DIRECTIVES：全局配置

　　　　　　RULES：日志记录相关的规则设置

　　　　RULES配置格式：facility.priority;facility.priority;……  target

　　　　　　facility:

　　　　　　　　\*：所有的facility

　　　　　　　　facility1,facility2,facility3,…：指定的facility列表

　　　　　　priority：

　　　　　　　　\*：表示所有级别

　　　　　　　　none：没有级别

　　　　　　　　priority：此级别以及高于此级别的所有级别

　　　　　　　　=priority：仅此级别

　　　　　　target:

　　　　　　　　文件路径：通常在/var/log/，文件路前的“-”表示异步写入

　　　　　　　　用户：将日志事件通知给指定用户，是通过将信息发送给登录到系统上的用户的终端进行显示；\*表示登录的所有用户

　　　　　　　　日志服务器：@host，把日志送往指定的远程服务器记录；host：表示日志服务器的地址，默认监听在tcp或者udp协议的514端口以提供服务

　　　　　　　　管道：|command，转发给其他命令处理

　　　　其他日志：

　　　　　　/var/log/wtmp:当前系统成功登录系统的日志 需要使用last命令查看

　　　　　　/var/log/btmp:当前系统尝试登录系统失败的日志 需要使用lastb命令查看

　　　　　　/var/log/dmesg:系统引导过程中的日志信息； 也可使用dmesg命令进行查看

　　　　　　lastlog:显示当前系统上的所有用户最近一次登录系统的时间

　　三、实验将sshd的日志分离到/var/log/sshd.log

　　sshd是远程登录Linux系统的一个服务，默认工作在22端口，通常情况下它的日志是记录在/var/log/secure 文件中，在之前我们不知道它为什么要记录在这个文件中，我们学习了rsyslog后，就明白了。

　　首先我们来看看sshd的配置文件

|  |  |
| --- | --- |
| 1  2  3  4 | [root@test ~]#grep "log" /etc/ssh/sshd\_config  #SyslogFacility AUTH  SyslogFacility AUTHPRIV  [root@test ~]# |

　　说明：可以看到sshd的配置文件中明确定义了syslogfacility authpriv。通过上面的介绍我们大概知道rsyslog 的facility 中就包括authpriv 这个设施。接下来我们在来看看rsyslog的配置文件

|  |  |
| --- | --- |
| 1  2  3  4  5 | [root@test ~]#grep "authpriv" /etc/rsyslog.conf  \*.info;mail.none;authpriv.none;cron.none                /var/log/messages  # The authpriv file has restricted access.  authpriv.\*                                              /var/log/secure  [root@test ~]# |

　　说明：看到以上的结果，结合我们之前介绍的rsyslog，是不是很清楚知道sshd的日志为什么记录在/etc/log/secure中了嘛。rsyslog的配置文件中明确定义了authpriv设施中的任何级别的日志都记录在/var/log/secure中。

　　更改sshd 配置文件 将日志的设施更改为自定义设施local3

|  |  |
| --- | --- |
| 1  2  3  4  5 | [root@test ~]#grep "log" /etc/ssh/sshd\_config  #SyslogFacility AUTH  #SyslogFacility AUTHPRIV  SyslogFacility local3  [root@test ~]# |

　　在rsyslog配置文件中指定 local3设施中的任何级别的目标文件为/var/log/sshd.log

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | [root@test ~]#grep "local" /etc/rsyslog.conf  $ModLoad imuxsock # provides support for local system logging (e.g. via logger command)  # Turn off message reception via local log socket;  # local messages are retrieved through imjournal now.  local7.\*                                                /var/log/boot.log  local3.\*                                                /var/log/sshd.log  [root@test ~]# |

　　重启rsyslogd 和sshd 服务

|  |  |
| --- | --- |
| 1 | [root@test ~]#systemctl restart rsyslog sshd |

　　查看/var/log/sshd.log

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | [root@test ~]#ll /var/log/sshd.log  -rw-------. 1 root root 207 12月 24 19:23 /var/log/sshd.log  [root@test ~]#cat /var/log/sshd.log  Dec 24 19:23:33 test sshd[4532]: Received signal 15; terminating.  Dec 24 19:23:33 test sshd[4575]: Server listening on 0.0.0.0 port 41319.  Dec 24 19:23:33 test sshd[4575]: Server listening on :: port 41319.  [root@test ~]# |

　　说明：要想用rsyslog来管理应用程序的日志，前提是应用程序内部实现rsyslog的日志接口，否则是不可以通过rsyslog来管理日志

　　四、日志管理小工具

　　logger：这个小工具可以生成日志，主要用于我们配置的日志系统是否可以正常的记录日志

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | [root@test ~]#logger  --help    用法：   logger [选项] [消息]    选项：   -T, --tcp             只使用 TCP   -d, --udp             只使用 UDP   -i, --id              同时记录进程 ID   -f, --file <文件>     记录此文件的内容   -h, --help            显示此帮助并退出   -S, --size <num>      maximum size for a single message (default 1024)   -n, --server <name>   write to this remote syslog server   -P, --port <port>     use this port for UDP or TCP connection   -p, --priority <prio> mark given message with this priority   -s, --stderr          output message to standard error as well   -t, --tag <标志>      用此标志标记每一行   -u, --socket <套接字> 写入此 Unix 套接字   -V, --version         输出版本信息并退出    [root@test ~]# |

　　给local3发送一条info级别或以上级别的日志

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | [root@test ~]#logger -p "local3.info" "this is test log"  [root@test ~]#tail /var/log/sshd.log  Dec 24 19:23:33 test sshd[4532]: Received signal 15; terminating.  Dec 24 19:23:33 test sshd[4575]: Server listening on 0.0.0.0 port 41319.  Dec 24 19:23:33 test sshd[4575]: Server listening on :: port 41319.  Dec 24 19:42:49 test qiuhom: this is test log  [root@test ~]# |

　　说明：有了这个工具我们可以很好的测试日志系统是否在正常记录日志

　　配置local4 的所有级别消息都发送给所有登录到系统的用户终端进行显示

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | [root@test ~]#grep "local" /etc/rsyslog.conf  $ModLoad imuxsock # provides support for local system logging (e.g. via logger command)  # Turn off message reception via local log socket;  # local messages are retrieved through imjournal now.  local7.\*                                                /var/log/boot.log  local3.\*                                                /var/log/sshd.log  local4.\*                                                \*  [root@test ~]#systemctl restart rsyslog  [root@test ~]#syst^C  [root@test ~]#who  root     tty1         2019-12-24 19:50  qiuhom   pts/0        2019-12-24 19:03 (192.168.0.232)  qiuhom   pts/1        2019-12-24 19:50 (192.168.0.232)  [root@test ~]#logger -p "local4.info" "this is test log"    Message from syslogd@test at Dec 24 19:53:02 ...   qiuhom:this is test log  [root@test ~]# |

　　journalctl：此工具是centos7上的一个日志管理工具。systemd统一管理所有unit的启动日志，带来的好处就是，可以用journalctl一个命令查看所有日志（内核日志和应用日志），日志的配置文件/etc/systemd/journald.conf

　　1）查看所有日志（默认情况下，只保存本次启动的日志）

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41 | [root@test ~]#journalctl  -- Logs begin at 一 2019-12-23 12:42:48 CST, end at 二 2019-12-24 20:01:01 CST. --  12月 23 12:42:48 docker systemd-journal[105]: Runtime journal is using 8.0M (max allowed 91.3M, trying to leave 136.9  12月 23 12:42:48 docker kernel: Initializing cgroup subsys cpuset  12月 23 12:42:48 docker kernel: Initializing cgroup subsys cpu  12月 23 12:42:48 docker kernel: Initializing cgroup subsys cpuacct  12月 23 12:42:48 docker kernel: Linux version 3.10.0-957.27.2.el7.x86\_64 (mockbuild@kbuilder.bsys.centos.org) (gcc ve  12月 23 12:42:48 docker kernel: Command line: BOOT\_IMAGE=/vmlinuz-3.10.0-957.27.2.el7.x86\_64 root=/dev/mapper/centos-  12月 23 12:42:48 docker kernel: Disabled fast string operations  12月 23 12:42:48 docker kernel: e820: BIOS-provided physical RAM map:  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x0000000000000000-0x000000000009ffff] usable  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x00000000000a0000-0x00000000000fffff] reserved  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x0000000000100000-0x000000007f045fff] usable  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x000000007f046000-0x000000007f0ccfff] ACPI NVS  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x000000007f0cd000-0x000000007f0cefff] reserved  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x000000007f0cf000-0x000000007f0d6fff] ACPI data  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x000000007f0d7000-0x000000007f103fff] reserved  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x000000007f104000-0x000000007f104fff] usable  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x000000007f105000-0x000000007f105fff] ACPI NVS  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x000000007f106000-0x000000007f125fff] reserved  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x000000007f126000-0x000000007f130fff] ACPI NVS  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x000000007f131000-0x000000007f158fff] reserved  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x000000007f159000-0x000000007f19bfff] ACPI NVS  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x000000007f19c000-0x000000007f586fff] usable  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x000000007f587000-0x000000007f6e3fff] reserved  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x000000007f6e4000-0x000000007f6effff] usable  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x000000007f6f0000-0x000000007fffffff] reserved  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x00000000e0000000-0x00000000efffffff] reserved  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x00000000fec00000-0x00000000fec00fff] reserved  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x00000000fed00000-0x00000000fed00fff] reserved  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x00000000fed1c000-0x00000000fed8ffff] reserved  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x00000000fee00000-0x00000000fee00fff] reserved  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x00000000ffe00000-0x00000000ffffffff] reserved  12月 23 12:42:48 docker kernel: NX (Execute Disable) protection: active  12月 23 12:42:48 docker kernel: e820: update [mem 0x0b91c018-0x0b92c057] usable ==> usable  12月 23 12:42:48 docker kernel: e820: update [mem 0x0b92d018-0x0b93d057] usable ==> usable  12月 23 12:42:48 docker kernel: extended physical RAM map:  12月 23 12:42:48 docker kernel: reserve setup\_data: [mem 0x0000000000000000-0x000000000009ffff] usable  12月 23 12:42:48 docker kernel: reserve setup\_data: [mem 0x00000000000a0000-0x00000000000fffff] reserved  12月 23 12:42:48 docker kernel: reserve setup\_data: [mem 0x0000000000100000-0x000000000b91c017] usable  lines 1-39 |

　　2）查看内核日志（不显示应用日志）

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41 | [root@test ~]#journalctl -k  -- Logs begin at 一 2019-12-23 12:42:48 CST, end at 二 2019-12-24 20:01:01 CST. --  12月 23 12:42:48 docker kernel: Initializing cgroup subsys cpuset  12月 23 12:42:48 docker kernel: Initializing cgroup subsys cpu  12月 23 12:42:48 docker kernel: Initializing cgroup subsys cpuacct  12月 23 12:42:48 docker kernel: Linux version 3.10.0-957.27.2.el7.x86\_64 (mockbuild@kbuilder.bsys.centos.org) (gcc ve  12月 23 12:42:48 docker kernel: Command line: BOOT\_IMAGE=/vmlinuz-3.10.0-957.27.2.el7.x86\_64 root=/dev/mapper/centos-  12月 23 12:42:48 docker kernel: Disabled fast string operations  12月 23 12:42:48 docker kernel: e820: BIOS-provided physical RAM map:  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x0000000000000000-0x000000000009ffff] usable  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x00000000000a0000-0x00000000000fffff] reserved  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x0000000000100000-0x000000007f045fff] usable  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x000000007f046000-0x000000007f0ccfff] ACPI NVS  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x000000007f0cd000-0x000000007f0cefff] reserved  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x000000007f0cf000-0x000000007f0d6fff] ACPI data  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x000000007f0d7000-0x000000007f103fff] reserved  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x000000007f104000-0x000000007f104fff] usable  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x000000007f105000-0x000000007f105fff] ACPI NVS  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x000000007f106000-0x000000007f125fff] reserved  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x000000007f126000-0x000000007f130fff] ACPI NVS  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x000000007f131000-0x000000007f158fff] reserved  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x000000007f159000-0x000000007f19bfff] ACPI NVS  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x000000007f19c000-0x000000007f586fff] usable  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x000000007f587000-0x000000007f6e3fff] reserved  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x000000007f6e4000-0x000000007f6effff] usable  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x000000007f6f0000-0x000000007fffffff] reserved  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x00000000e0000000-0x00000000efffffff] reserved  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x00000000fec00000-0x00000000fec00fff] reserved  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x00000000fed00000-0x00000000fed00fff] reserved  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x00000000fed1c000-0x00000000fed8ffff] reserved  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x00000000fee00000-0x00000000fee00fff] reserved  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x00000000ffe00000-0x00000000ffffffff] reserved  12月 23 12:42:48 docker kernel: NX (Execute Disable) protection: active  12月 23 12:42:48 docker kernel: e820: update [mem 0x0b91c018-0x0b92c057] usable ==> usable  12月 23 12:42:48 docker kernel: e820: update [mem 0x0b92d018-0x0b93d057] usable ==> usable  12月 23 12:42:48 docker kernel: extended physical RAM map:  12月 23 12:42:48 docker kernel: reserve setup\_data: [mem 0x0000000000000000-0x000000000009ffff] usable  12月 23 12:42:48 docker kernel: reserve setup\_data: [mem 0x00000000000a0000-0x00000000000fffff] reserved  12月 23 12:42:48 docker kernel: reserve setup\_data: [mem 0x0000000000100000-0x000000000b91c017] usable  12月 23 12:42:48 docker kernel: reserve setup\_data: [mem 0x000000000b91c018-0x000000000b92c057] usable  lines 1-39 |

　　3）查看系统本次启动的日志

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41 | [root@test ~]#journalctl -b 0  -- Logs begin at 一 2019-12-23 12:42:48 CST, end at 二 2019-12-24 20:01:01 CST. --  12月 23 12:42:48 docker systemd-journal[105]: Runtime journal is using 8.0M (max allowed 91.3M, trying to leave 136.9  12月 23 12:42:48 docker kernel: Initializing cgroup subsys cpuset  12月 23 12:42:48 docker kernel: Initializing cgroup subsys cpu  12月 23 12:42:48 docker kernel: Initializing cgroup subsys cpuacct  12月 23 12:42:48 docker kernel: Linux version 3.10.0-957.27.2.el7.x86\_64 (mockbuild@kbuilder.bsys.centos.org) (gcc ve  12月 23 12:42:48 docker kernel: Command line: BOOT\_IMAGE=/vmlinuz-3.10.0-957.27.2.el7.x86\_64 root=/dev/mapper/centos-  12月 23 12:42:48 docker kernel: Disabled fast string operations  12月 23 12:42:48 docker kernel: e820: BIOS-provided physical RAM map:  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x0000000000000000-0x000000000009ffff] usable  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x00000000000a0000-0x00000000000fffff] reserved  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x0000000000100000-0x000000007f045fff] usable  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x000000007f046000-0x000000007f0ccfff] ACPI NVS  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x000000007f0cd000-0x000000007f0cefff] reserved  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x000000007f0cf000-0x000000007f0d6fff] ACPI data  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x000000007f0d7000-0x000000007f103fff] reserved  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x000000007f104000-0x000000007f104fff] usable  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x000000007f105000-0x000000007f105fff] ACPI NVS  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x000000007f106000-0x000000007f125fff] reserved  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x000000007f126000-0x000000007f130fff] ACPI NVS  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x000000007f131000-0x000000007f158fff] reserved  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x000000007f159000-0x000000007f19bfff] ACPI NVS  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x000000007f19c000-0x000000007f586fff] usable  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x000000007f587000-0x000000007f6e3fff] reserved  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x000000007f6e4000-0x000000007f6effff] usable  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x000000007f6f0000-0x000000007fffffff] reserved  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x00000000e0000000-0x00000000efffffff] reserved  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x00000000fec00000-0x00000000fec00fff] reserved  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x00000000fed00000-0x00000000fed00fff] reserved  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x00000000fed1c000-0x00000000fed8ffff] reserved  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x00000000fee00000-0x00000000fee00fff] reserved  12月 23 12:42:48 docker kernel: BIOS-e820: [mem 0x00000000ffe00000-0x00000000ffffffff] reserved  12月 23 12:42:48 docker kernel: NX (Execute Disable) protection: active  12月 23 12:42:48 docker kernel: e820: update [mem 0x0b91c018-0x0b92c057] usable ==> usable  12月 23 12:42:48 docker kernel: e820: update [mem 0x0b92d018-0x0b93d057] usable ==> usable  12月 23 12:42:48 docker kernel: extended physical RAM map:  12月 23 12:42:48 docker kernel: reserve setup\_data: [mem 0x0000000000000000-0x000000000009ffff] usable  12月 23 12:42:48 docker kernel: reserve setup\_data: [mem 0x00000000000a0000-0x00000000000fffff] reserved  12月 23 12:42:48 docker kernel: reserve setup\_data: [mem 0x0000000000100000-0x000000000b91c017] usable  lines 1-39 |

　　4）查看指定时间的日志

|  |  |
| --- | --- |
| 1  2  3  4  5 | journalctl --since="2017-10-30 18:10:30"  journalctl --since "20 min ago"  journalctl --since yesterday  journalctl --since "2017-01-10" --until "2017-01-11 03:00"  journalctl --since 09:00 --until "1 hour ago" |

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41 | [root@test ~]#journalctl --since 09:00 --until "1 hour ago"  -- Logs begin at 一 2019-12-23 12:42:48 CST, end at 二 2019-12-24 20:01:01 CST. --  12月 24 09:01:01 test systemd[1]: Created slice User Slice of root.  12月 24 09:01:01 test systemd[1]: Started Session 22 of user root.  12月 24 09:01:01 test CROND[2543]: (root) CMD (run-parts /etc/cron.hourly)  12月 24 09:01:01 test run-parts(/etc/cron.hourly)[2546]: starting 0anacron  12月 24 09:01:01 test run-parts(/etc/cron.hourly)[2552]: finished 0anacron  12月 24 09:01:02 test systemd[1]: Removed slice User Slice of root.  12月 24 10:01:01 test systemd[1]: Created slice User Slice of root.  12月 24 10:01:01 test systemd[1]: Started Session 23 of user root.  12月 24 10:01:01 test CROND[2561]: (root) CMD (run-parts /etc/cron.hourly)  12月 24 10:01:01 test run-parts(/etc/cron.hourly)[2564]: starting 0anacron  12月 24 10:01:01 test run-parts(/etc/cron.hourly)[2570]: finished 0anacron  12月 24 10:01:01 test systemd[1]: Removed slice User Slice of root.  12月 24 11:01:01 test systemd[1]: Created slice User Slice of root.  12月 24 11:01:01 test systemd[1]: Started Session 24 of user root.  12月 24 11:01:01 test CROND[2579]: (root) CMD (run-parts /etc/cron.hourly)  12月 24 11:01:01 test run-parts(/etc/cron.hourly)[2582]: starting 0anacron  12月 24 11:01:01 test run-parts(/etc/cron.hourly)[2588]: finished 0anacron  12月 24 11:01:01 test systemd[1]: Removed slice User Slice of root.  12月 24 12:01:01 test systemd[1]: Created slice User Slice of root.  12月 24 12:01:01 test systemd[1]: Started Session 25 of user root.  12月 24 12:01:01 test CROND[2597]: (root) CMD (run-parts /etc/cron.hourly)  12月 24 12:01:01 test run-parts(/etc/cron.hourly)[2600]: starting 0anacron  12月 24 12:01:01 test run-parts(/etc/cron.hourly)[2606]: finished 0anacron  12月 24 12:01:01 test systemd[1]: Removed slice User Slice of root.  12月 24 12:58:31 test systemd[1]: Starting Cleanup of Temporary Directories...  12月 24 12:58:32 test systemd[1]: Started Cleanup of Temporary Directories.  12月 24 13:01:01 test systemd[1]: Created slice User Slice of root.  12月 24 13:01:01 test systemd[1]: Started Session 26 of user root.  12月 24 13:01:01 test CROND[2619]: (root) CMD (run-parts /etc/cron.hourly)  12月 24 13:01:01 test run-parts(/etc/cron.hourly)[2622]: starting 0anacron  12月 24 13:01:01 test run-parts(/etc/cron.hourly)[2628]: finished 0anacron  12月 24 13:01:01 test systemd[1]: Removed slice User Slice of root.  12月 24 13:16:24 test sshd[2635]: Accepted password for qiuhom from 192.168.0.232 port 2097 ssh2  12月 24 13:16:25 test systemd[1]: Created slice User Slice of qiuhom.  12月 24 13:16:25 test systemd-logind[773]: New session 27 of user qiuhom.  12月 24 13:16:25 test systemd[1]: Started Session 27 of user qiuhom.  12月 24 13:16:25 test sshd[2635]: pam\_unix(sshd:session): session opened for user qiuhom by (uid=0)  12月 24 13:16:28 test su[2673]: (to root) qiuhom on pts/0  lines 1-39 |

　　说明：指定时间不能超过记录时间的最早时间

　　5）显示尾部的最新日志默认是现实10行

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | [root@test ~]#journalctl -n  -- Logs begin at 一 2019-12-23 12:42:48 CST, end at 二 2019-12-24 20:01:01 CST. --  12月 24 19:52:16 test rsyslogd[6118]: error during parsing file /etc/rsyslog.conf, on or before line 75: warnings occ  12月 24 19:52:16 test polkitd[752]: Unregistered Authentication Agent for unix-process:6111:11217058 (system bus name  12月 24 19:53:02 test qiuhom[6222]: this is test log  12月 24 19:53:47 test su[6256]: (to root) qiuhom on pts/1  12月 24 19:53:47 test su[6256]: pam\_unix(su-l:session): session opened for user root by qiuhom(uid=1000)  12月 24 19:53:54 test qiuhom[6466]: this is test log  12月 24 20:01:01 test systemd[1]: Started Session 37 of user root.  12月 24 20:01:01 test CROND[6791]: (root) CMD (run-parts /etc/cron.hourly)  12月 24 20:01:01 test run-parts(/etc/cron.hourly)[6794]: starting 0anacron  12月 24 20:01:01 test run-parts(/etc/cron.hourly)[6800]: finished 0anacron  [root@test ~]#journalctl -n 15  -- Logs begin at 一 2019-12-23 12:42:48 CST, end at 二 2019-12-24 20:01:01 CST. --  12月 24 19:52:16 test systemd[1]: Stopped System Logging Service.  12月 24 19:52:16 test systemd[1]: Starting System Logging Service...  12月 24 19:52:16 test rsyslogd[6118]:  [origin software="rsyslogd" swVersion="8.24.0-34.el7" x-pid="6118" x-info="htt  12月 24 19:52:16 test rsyslogd[6118]: action '\*' treated as ':omusrmsg:\*' - please use ':omusrmsg:\*' syntax instead,  12月 24 19:52:16 test systemd[1]: Started System Logging Service.  12月 24 19:52:16 test rsyslogd[6118]: error during parsing file /etc/rsyslog.conf, on or before line 75: warnings occ  12月 24 19:52:16 test polkitd[752]: Unregistered Authentication Agent for unix-process:6111:11217058 (system bus name  12月 24 19:53:02 test qiuhom[6222]: this is test log  12月 24 19:53:47 test su[6256]: (to root) qiuhom on pts/1  12月 24 19:53:47 test su[6256]: pam\_unix(su-l:session): session opened for user root by qiuhom(uid=1000)  12月 24 19:53:54 test qiuhom[6466]: this is test log  12月 24 20:01:01 test systemd[1]: Started Session 37 of user root.  12月 24 20:01:01 test CROND[6791]: (root) CMD (run-parts /etc/cron.hourly)  12月 24 20:01:01 test run-parts(/etc/cron.hourly)[6794]: starting 0anacron  12月 24 20:01:01 test run-parts(/etc/cron.hourly)[6800]: finished 0anacron  [root@test ~]# |

　　6）实时滚动显示最新日志

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | [root@test ~]#journalctl -f  -- Logs begin at 一 2019-12-23 12:42:48 CST. --  12月 24 19:52:16 test rsyslogd[6118]: error during parsing file /etc/rsyslog.conf, on or before line 75: warnings occured in file '/etc/rsyslog.conf' around line 75 [v8.24.0-34.el7 try http://www.rsyslog.com/e/2207 ]  12月 24 19:52:16 test polkitd[752]: Unregistered Authentication Agent for unix-process:6111:11217058 (system bus name :1.95, object path /org/freedesktop/PolicyKit1/AuthenticationAgent, locale zh\_CN.UTF-8) (disconnected from bus)  12月 24 19:53:02 test qiuhom[6222]: this is test log  12月 24 19:53:47 test su[6256]: (to root) qiuhom on pts/1  12月 24 19:53:47 test su[6256]: pam\_unix(su-l:session): session opened for user root by qiuhom(uid=1000)  12月 24 19:53:54 test qiuhom[6466]: this is test log  12月 24 20:01:01 test systemd[1]: Started Session 37 of user root.  12月 24 20:01:01 test CROND[6791]: (root) CMD (run-parts /etc/cron.hourly)  12月 24 20:01:01 test run-parts(/etc/cron.hourly)[6794]: starting 0anacron  12月 24 20:01:01 test run-parts(/etc/cron.hourly)[6800]: finished 0anacron  12月 24 20:51:28 test qiuhom[8356]: this is a test log |

　　说明：此选项同tail -f 类似

　　7）查看指定服务的日志

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45 | [root@test ~]#journalctl /sbin/nginx  -- Logs begin at 一 2019-12-23 12:42:48 CST, end at 二 2019-12-24 20:51:28 CST. --  12月 23 12:43:07 test nginx[1050]: nginx: the configuration file /etc/nginx/nginx.conf syntax is ok  12月 23 12:43:07 test nginx[1050]: nginx: configuration file /etc/nginx/nginx.conf test is successful  [root@test ~]#journalctl /usr/lib/systemd/systemd  -- Logs begin at 一 2019-12-23 12:42:48 CST, end at 二 2019-12-24 20:51:28 CST. --  12月 23 12:42:49 docker systemd[1]: Started Setup Virtual Console.  12月 23 12:42:49 docker systemd[1]: Started dracut cmdline hook.  12月 23 12:42:49 docker systemd[1]: Starting dracut pre-udev hook...  12月 23 12:42:49 docker systemd[1]: Started dracut pre-udev hook.  12月 23 12:42:49 docker systemd[1]: Starting udev Kernel Device Manager...  12月 23 12:42:49 docker systemd[1]: Started udev Kernel Device Manager.  12月 23 12:42:49 docker systemd[1]: Starting udev Coldplug all Devices...  12月 23 12:42:49 docker systemd[1]: Mounting Configuration File System...  12月 23 12:42:49 docker systemd[1]: Mounted Configuration File System.  12月 23 12:42:49 docker systemd[1]: Started udev Coldplug all Devices.  12月 23 12:42:49 docker systemd[1]: Reached target System Initialization.  12月 23 12:42:49 docker systemd[1]: Starting Show Plymouth Boot Screen...  12月 23 12:42:49 docker systemd[1]: Starting dracut initqueue hook...  12月 23 12:42:49 docker systemd[1]: Started Show Plymouth Boot Screen.  12月 23 12:42:49 docker systemd[1]: Started Forward Password Requests to Plymouth Directory Watch.  12月 23 12:42:49 docker systemd[1]: Reached target Paths.  12月 23 12:42:49 docker systemd[1]: Reached target Basic System.  12月 23 12:42:51 docker systemd[1]: Found device /dev/mapper/centos-root.  12月 23 12:42:51 docker systemd[1]: Starting File System Check on /dev/mapper/centos-root...  12月 23 12:42:51 docker systemd[1]: Started File System Check on /dev/mapper/centos-root.  12月 23 12:42:51 docker systemd[1]: Started dracut initqueue hook.  12月 23 12:42:51 docker systemd[1]: Reached target Remote File Systems (Pre).  12月 23 12:42:51 docker systemd[1]: Reached target Remote File Systems.  12月 23 12:42:51 docker systemd[1]: Mounting /sysroot...  12月 23 12:42:52 docker systemd[1]: Mounted /sysroot.  12月 23 12:42:52 docker systemd[1]: Reached target Initrd Root File System.  12月 23 12:42:52 docker systemd[1]: Starting Reload Configuration from the Real Root...  12月 23 12:42:52 docker systemd[1]: Reloading.  12月 23 12:42:52 docker systemd[1]: Started Reload Configuration from the Real Root.  12月 23 12:42:52 docker systemd[1]: Reached target Initrd File Systems.  12月 23 12:42:52 docker systemd[1]: Reached target Initrd Default Target.  12月 23 12:42:52 docker systemd[1]: Starting dracut pre-pivot and cleanup hook...  12月 23 12:42:52 docker systemd[1]: Started dracut pre-pivot and cleanup hook.  12月 23 12:42:52 docker systemd[1]: Starting Cleaning Up and Shutting Down Daemons...  12月 23 12:42:52 docker systemd[1]: Stopped target Timers.  12月 23 12:42:52 docker systemd[1]: Starting Plymouth switch root service...  12月 23 12:42:52 docker systemd[1]: Stopped Cleaning Up and Shutting Down Daemons.  12月 23 12:42:52 docker systemd[1]: Stopped dracut pre-pivot and cleanup hook.  lines 1-39 |

　　8）查看指定进程的日志

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45 | [root@test ~]#journalctl \_PID=757  -- Logs begin at 一 2019-12-23 12:42:48 CST, end at 二 2019-12-24 21:08:23 CST. --  12月 23 12:42:56 test chronyd[757]: chronyd version 3.4 starting (+CMDMON +NTP +REFCLOCK +RTC +PRIVDROP +SCFILTER +SI  12月 23 12:42:56 test chronyd[757]: Frequency -5.019 +/- 0.085 ppm read from /var/lib/chrony/drift  12月 23 12:43:07 test chronyd[757]: Selected source 84.16.67.12  [root@test ~]#journalctl \_PID=10781  -- Logs begin at 一 2019-12-23 12:42:48 CST, end at 二 2019-12-24 21:08:23 CST. --  12月 24 21:08:08 test setroubleshoot[10781]: SELinux is preventing /usr/sbin/nginx from name\_connect access on the tc  12月 24 21:08:08 test python[10781]: SELinux is preventing /usr/sbin/nginx from name\_connect access on the tcp\_socket                                          \*\*\*\*\*  Plugin connect\_ports (85.9 confidence) suggests   \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*                                          If you want to allow /usr/sbin/nginx to connect to network port 8888                                        Then you need to modify the port type.                                        Do                                        # semanage port -a -t PORT\_TYPE -p tcp 8888                                            where PORT\_TYPE is one of the following: dns\_port\_t, dnssec\_port\_t, http\_po                                          \*\*\*\*\*  Plugin catchall\_boolean (7.33 confidence) suggests   \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*                                          If you want to allow httpd to can network connect                                        Then you must tell SELinux about this by enabling the 'httpd\_can\_network\_connec                                          Do                                        setsebool -P httpd\_can\_network\_connect 1                                          \*\*\*\*\*  Plugin catchall\_boolean (7.33 confidence) suggests   \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*                                          If you want to allow nis to enabled                                        Then you must tell SELinux about this by enabling the 'nis\_enabled' boolean.                                          Do                                        setsebool -P nis\_enabled 1                                          \*\*\*\*\*  Plugin catchall (1.35 confidence) suggests   \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*                                          If you believe that nginx should be allowed name\_connect access on the port 888                                        Then you should report this as a bug.                                        You can generate a local policy module to allow this access.                                        Do                                        allow this access for now by executing:                                        # ausearch -c 'nginx' --raw | audit2allow -M my-nginx                                        # semodule -i my-nginx.pp    lines 1-38/38 (END) |

　　9）查看某个路径下脚本的日志

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36 | [root@test ~]#journalctl /usr/bin/bash  -- Logs begin at 一 2019-12-23 12:42:48 CST, end at 二 2019-12-24 21:08:23 CST. --  12月 23 12:42:56 test augenrules[730]: /sbin/augenrules: No change  12月 23 12:42:56 test augenrules[730]: No rules  12月 23 12:43:06 test network[883]: 正在打开环回接口： [  确定  ]  12月 23 12:43:06 test network[883]: 正在打开接口 enp2s0： [  确定  ]  12月 23 13:01:01 test CROND[1515]: (root) CMD (run-parts /etc/cron.hourly)  12月 23 14:01:01 test CROND[2160]: (root) CMD (run-parts /etc/cron.hourly)  12月 23 15:01:01 test CROND[2185]: (root) CMD (run-parts /etc/cron.hourly)  12月 23 16:01:01 test CROND[2203]: (root) CMD (run-parts /etc/cron.hourly)  12月 23 17:01:01 test CROND[2221]: (root) CMD (run-parts /etc/cron.hourly)  12月 23 18:01:01 test CROND[2239]: (root) CMD (run-parts /etc/cron.hourly)  12月 23 19:01:02 test CROND[2256]: (root) CMD (run-parts /etc/cron.hourly)  12月 23 20:01:01 test CROND[2275]: (root) CMD (run-parts /etc/cron.hourly)  12月 23 21:01:01 test CROND[2291]: (root) CMD (run-parts /etc/cron.hourly)  12月 23 22:01:01 test CROND[2309]: (root) CMD (run-parts /etc/cron.hourly)  12月 23 23:01:01 test CROND[2328]: (root) CMD (run-parts /etc/cron.hourly)  12月 24 01:01:01 test CROND[2368]: (root) CMD (run-parts /etc/cron.hourly)  12月 24 02:01:01 test CROND[2388]: (root) CMD (run-parts /etc/cron.hourly)  12月 24 03:01:01 test CROND[2408]: (root) CMD (run-parts /etc/cron.hourly)  12月 24 04:01:01 test CROND[2455]: (root) CMD (run-parts /etc/cron.hourly)  12月 24 07:01:01 test CROND[2507]: (root) CMD (run-parts /etc/cron.hourly)  12月 24 08:01:01 test CROND[2525]: (root) CMD (run-parts /etc/cron.hourly)  12月 24 09:01:01 test CROND[2543]: (root) CMD (run-parts /etc/cron.hourly)  12月 24 10:01:01 test CROND[2561]: (root) CMD (run-parts /etc/cron.hourly)  12月 24 11:01:01 test CROND[2579]: (root) CMD (run-parts /etc/cron.hourly)  12月 24 12:01:01 test CROND[2597]: (root) CMD (run-parts /etc/cron.hourly)  12月 24 13:01:01 test CROND[2619]: (root) CMD (run-parts /etc/cron.hourly)  12月 24 14:01:01 test CROND[3415]: (root) CMD (run-parts /etc/cron.hourly)  12月 24 16:01:01 test CROND[3454]: (root) CMD (run-parts /etc/cron.hourly)  12月 24 17:01:01 test CROND[3472]: (root) CMD (run-parts /etc/cron.hourly)  12月 24 18:01:01 test CROND[3490]: (root) CMD (run-parts /etc/cron.hourly)  12月 24 19:01:01 test CROND[3509]: (root) CMD (run-parts /etc/cron.hourly)  12月 24 20:01:01 test CROND[6791]: (root) CMD (run-parts /etc/cron.hourly)  12月 24 21:01:01 test CROND[9711]: (root) CMD (run-parts /etc/cron.hourly)  [root@test ~]# |

　　10）查看指定用户的日志

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | [root@test ~]#id qiuhom  uid=1000(qiuhom) gid=1000(qiuhom) 组=1000(qiuhom)  [root@test ~]#journalctl \_UID=1000  -- Logs begin at 一 2019-12-23 12:42:48 CST, end at 二 2019-12-24 21:08:23 CST. --  12月 23 13:23:58 test su[1912]: (to root) qiuhom on pts/0  12月 23 13:23:58 test su[1912]: pam\_unix(su-l:session): session opened for user root by qiuhom(uid=1000)  12月 23 14:07:46 test su[1912]: pam\_unix(su-l:session): session closed for user root  12月 24 13:16:28 test su[2673]: (to root) qiuhom on pts/0  12月 24 13:16:28 test su[2673]: pam\_unix(su-l:session): session opened for user root by qiuhom(uid=1000)  12月 24 14:02:19 test su[2673]: pam\_unix(su-l:session): session closed for user root  12月 24 19:03:55 test su[3562]: (to root) qiuhom on pts/0  12月 24 19:03:55 test su[3562]: pam\_unix(su-l:session): session opened for user root by qiuhom(uid=1000)  12月 24 19:53:47 test su[6256]: (to root) qiuhom on pts/1  12月 24 19:53:47 test su[6256]: pam\_unix(su-l:session): session opened for user root by qiuhom(uid=1000)  [root@test ~]# |

　　11）查看某个unit的日志

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | [root@test ~]#journalctl -u nginx.service  -- Logs begin at 一 2019-12-23 12:42:48 CST, end at 二 2019-12-24 21:08:23 CST. --  12月 23 12:43:07 test systemd[1]: Starting The nginx HTTP and reverse proxy server...  12月 23 12:43:07 test nginx[1050]: nginx: the configuration file /etc/nginx/nginx.conf syntax is ok  12月 23 12:43:07 test nginx[1050]: nginx: configuration file /etc/nginx/nginx.conf test is successful  12月 23 12:43:08 test systemd[1]: Started The nginx HTTP and reverse proxy server.  [root@test ~]#journalctl -u nginx.service --since today  -- No entries --  [root@test ~]#systemctl restart nginx  [root@test ~]#journalctl -u nginx.service --since today  -- Logs begin at 一 2019-12-23 12:42:48 CST, end at 二 2019-12-24 21:14:31 CST. --  12月 24 21:14:31 test systemd[1]: Stopping The nginx HTTP and reverse proxy server...  12月 24 21:14:31 test systemd[1]: Stopped The nginx HTTP and reverse proxy server.  12月 24 21:14:31 test systemd[1]: Starting The nginx HTTP and reverse proxy server...  12月 24 21:14:31 test nginx[11296]: nginx: the configuration file /etc/nginx/nginx.conf syntax is ok  12月 24 21:14:31 test nginx[11296]: nginx: configuration file /etc/nginx/nginx.conf test is successful  12月 24 21:14:31 test systemd[1]: Started The nginx HTTP and reverse proxy server.  [root@test ~]# |

　　说明：可以同时指定多个unit，分别用-u指定其名即可，也可以用--since 指定时间，也可以用-f来跟踪某个nuit的最新日志

　　12）查看指定优先级（及其以上级别）的日志，共有8级

　　　　0: emerg  
　　　　1: alert  
　　　　2: crit  
　　　　3: err  
　　　　4: warning  
　　　　5: notice  
　　　　6: info  
　　　　7: debug

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | [root@test ~]#journalctl -p err  -- Logs begin at 一 2019-12-23 12:42:48 CST, end at 二 2019-12-24 21:14:31 CST. --  12月 23 12:42:50 docker kernel: gma500 0000:00:02.0: GPU: power management timed out.  12月 24 19:47:41 test rsyslogd[5521]: error during parsing file /etc/rsyslog.conf, on or before line 75: warnings occ  12月 24 19:52:16 test rsyslogd[6118]: error during parsing file /etc/rsyslog.conf, on or before line 75: warnings occ  12月 24 21:07:45 test setroubleshoot[10603]: SELinux is preventing /usr/sbin/nginx from name\_connect access on the tc  12月 24 21:07:48 test setroubleshoot[10603]: SELinux is preventing /usr/sbin/nginx from name\_connect access on the tc  12月 24 21:07:49 test setroubleshoot[10603]: SELinux is preventing /usr/sbin/nginx from name\_connect access on the tc  12月 24 21:07:50 test setroubleshoot[10603]: SELinux is preventing /usr/sbin/nginx from name\_connect access on the tc  12月 24 21:07:50 test setroubleshoot[10603]: SELinux is preventing /usr/sbin/nginx from name\_connect access on the tc  12月 24 21:07:51 test setroubleshoot[10603]: SELinux is preventing /usr/sbin/nginx from name\_connect access on the tc  12月 24 21:07:52 test setroubleshoot[10603]: SELinux is preventing /usr/sbin/nginx from name\_connect access on the tc  12月 24 21:07:53 test setroubleshoot[10603]: SELinux is preventing /usr/sbin/nginx from name\_connect access on the tc  12月 24 21:07:53 test setroubleshoot[10603]: SELinux is preventing /usr/sbin/nginx from name\_connect access on the tc  12月 24 21:07:54 test setroubleshoot[10603]: SELinux is preventing /usr/sbin/nginx from name\_connect access on the tc  12月 24 21:07:55 test setroubleshoot[10603]: SELinux is preventing /usr/sbin/nginx from name\_connect access on the tc  12月 24 21:07:56 test setroubleshoot[10603]: SELinux is preventing /usr/sbin/nginx from name\_connect access on the tc  12月 24 21:07:56 test setroubleshoot[10603]: SELinux is preventing /usr/sbin/nginx from name\_connect access on the tc  12月 24 21:07:57 test setroubleshoot[10603]: SELinux is preventing /usr/sbin/nginx from name\_connect access on the tc  12月 24 21:07:58 test setroubleshoot[10603]: SELinux is preventing /usr/sbin/nginx from name\_connect access on the tc  12月 24 21:07:58 test setroubleshoot[10603]: SELinux is preventing /usr/sbin/nginx from name\_connect access on the tc  12月 24 21:07:59 test setroubleshoot[10603]: SELinux is preventing /usr/sbin/nginx from name\_connect access on the tc  12月 24 21:08:00 test setroubleshoot[10603]: SELinux is preventing /usr/sbin/nginx from name\_connect access on the tc  12月 24 21:08:01 test setroubleshoot[10603]: SELinux is preventing /usr/sbin/nginx from name\_connect access on the tc  12月 24 21:08:01 test setroubleshoot[10603]: SELinux is preventing /usr/sbin/nginx from name\_connect access on the tc  12月 24 21:08:02 test setroubleshoot[10603]: SELinux is preventing /usr/sbin/nginx from name\_connect access on the tc  12月 24 21:08:03 test setroubleshoot[10603]: SELinux is preventing /usr/sbin/nginx from name\_connect access on the tc  12月 24 21:08:08 test setroubleshoot[10781]: SELinux is preventing /usr/sbin/nginx from name\_connect access on the tc  12月 24 21:08:23 test setroubleshoot[10826]: SELinux is preventing /usr/sbin/nginx from name\_connect access on the tc  [root@test ~]# |

　　13）日志默认分页输出，--no-pager 改为正常的标准输出

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | ……省略部分信息  12月 24 21:14:31 test polkitd[752]: Registered Authentication Agent for unix-process:11283:11710498 (system bus name :1.105 [/usr/bin/pkttyagent --notify-fd 5 --fallback], object path /org/freedesktop/PolicyKit1/AuthenticationAgent, locale zh\_CN.UTF-8)  12月 24 21:14:31 test systemd[1]: Stopping The nginx HTTP and reverse proxy server...  12月 24 21:14:31 test systemd[1]: Stopped The nginx HTTP and reverse proxy server.  12月 24 21:14:31 test systemd[1]: Starting The nginx HTTP and reverse proxy server...  12月 24 21:14:31 test nginx[11296]: nginx: the configuration file /etc/nginx/nginx.conf syntax is ok  12月 24 21:14:31 test nginx[11296]: nginx: configuration file /etc/nginx/nginx.conf test is successful  12月 24 21:14:31 test systemd[1]: Started The nginx HTTP and reverse proxy server.  12月 24 21:14:31 test polkitd[752]: Unregistered Authentication Agent for unix-process:11283:11710498 (system bus name :1.105, object path /org/freedesktop/PolicyKit1/AuthenticationAgent, locale zh\_CN.UTF-8) (disconnected from bus)  [root@test ~]# |

　　14）以json格式（单行）输出

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | [root@test ~]#journalctl -b -u nginx.service -o json  { "\_\_CURSOR" : "s=1757eca9c8674c60bc078967261ae3d2;i=4fe;b=e3110b5a73e44bebb9ac87b21fad016d;m=1401ea7;t=59a57a9eb3d4c  { "\_\_CURSOR" : "s=1757eca9c8674c60bc078967261ae3d2;i=50a;b=e3110b5a73e44bebb9ac87b21fad016d;m=1488bea;t=59a57a9f3aa8f  { "\_\_CURSOR" : "s=1757eca9c8674c60bc078967261ae3d2;i=50b;b=e3110b5a73e44bebb9ac87b21fad016d;m=1489f61;t=59a57a9f3be06  { "\_\_CURSOR" : "s=1757eca9c8674c60bc078967261ae3d2;i=50d;b=e3110b5a73e44bebb9ac87b21fad016d;m=14d1bc8;t=59a57a9f83a6e  { "\_\_CURSOR" : "s=1757eca9c8674c60bc078967261ae3d2;i=6b9;b=e3110b5a73e44bebb9ac87b21fad016d;m=1b44014f22;t=59a72ecac6  { "\_\_CURSOR" : "s=1757eca9c8674c60bc078967261ae3d2;i=6ba;b=e3110b5a73e44bebb9ac87b21fad016d;m=1b44020532;t=59a72ecad2  { "\_\_CURSOR" : "s=1757eca9c8674c60bc078967261ae3d2;i=6bb;b=e3110b5a73e44bebb9ac87b21fad016d;m=1b44024a99;t=59a72ecad6  { "\_\_CURSOR" : "s=1757eca9c8674c60bc078967261ae3d2;i=6bc;b=e3110b5a73e44bebb9ac87b21fad016d;m=1b44046779;t=59a72ecaf8  { "\_\_CURSOR" : "s=1757eca9c8674c60bc078967261ae3d2;i=6bd;b=e3110b5a73e44bebb9ac87b21fad016d;m=1b44046be4;t=59a72ecaf8  { "\_\_CURSOR" : "s=1757eca9c8674c60bc078967261ae3d2;i=6be;b=e3110b5a73e44bebb9ac87b21fad016d;m=1b440637c3;t=59a72ecb15  [root@test ~]# |

　　多行输出，可读性更好

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41 | [root@test ~]#journalctl -b -u nginx.service -o json-pretty  {          "\_\_CURSOR" : "s=1757eca9c8674c60bc078967261ae3d2;i=4fe;b=e3110b5a73e44bebb9ac87b21fad016d;m=1401ea7;t=59a57a9          "\_\_REALTIME\_TIMESTAMP" : "1577076187151692",          "\_\_MONOTONIC\_TIMESTAMP" : "20979367",          "\_BOOT\_ID" : "e3110b5a73e44bebb9ac87b21fad016d",          "PRIORITY" : "6",          "\_UID" : "0",          "\_GID" : "0",          "\_MACHINE\_ID" : "931bcb70deb1435eaea1d542d13878cc",          "SYSLOG\_FACILITY" : "3",          "SYSLOG\_IDENTIFIER" : "systemd",          "\_TRANSPORT" : "journal",          "\_PID" : "1",          "\_COMM" : "systemd",          "\_EXE" : "/usr/lib/systemd/systemd",          "\_CAP\_EFFECTIVE" : "1fffffffff",          "\_SYSTEMD\_CGROUP" : "/",          "CODE\_FILE" : "src/core/unit.c",          "CODE\_FUNCTION" : "unit\_status\_log\_starting\_stopping\_reloading",          "MESSAGE\_ID" : "7d4958e842da4a758f6c1cdc7b36dcc5",          "\_HOSTNAME" : "test",          "\_CMDLINE" : "/usr/lib/systemd/systemd --switched-root --system --deserialize 22",          "\_SELINUX\_CONTEXT" : "system\_u:system\_r:init\_t:s0",          "CODE\_LINE" : "1395",          "UNIT" : "nginx.service",          "MESSAGE" : "Starting The nginx HTTP and reverse proxy server...",          "\_SOURCE\_REALTIME\_TIMESTAMP" : "1577076187143557"  }  {          "\_\_CURSOR" : "s=1757eca9c8674c60bc078967261ae3d2;i=50a;b=e3110b5a73e44bebb9ac87b21fad016d;m=1488bea;t=59a57a9          "\_\_REALTIME\_TIMESTAMP" : "1577076187703951",          "\_\_MONOTONIC\_TIMESTAMP" : "21531626",          "\_BOOT\_ID" : "e3110b5a73e44bebb9ac87b21fad016d",          "PRIORITY" : "6",          "\_UID" : "0",          "\_GID" : "0",          "\_SYSTEMD\_SLICE" : "system.slice",          "\_MACHINE\_ID" : "931bcb70deb1435eaea1d542d13878cc",          "SYSLOG\_FACILITY" : "3",  [root@test ~]# |

　　15）显示日志占据的磁盘空间

|  |  |
| --- | --- |
| 1  2  3 | [root@test ~]#journalctl --disk-usage  Archived and active journals take up 8.0M on disk.  [root@test ~]# |

　　指定日志文件占据的最大空间

|  |  |
| --- | --- |
| 1  2  3 | [root@test ~]#journalctl --vacuum-size=1G  Vacuuming done, freed 0B of archived journals on disk.  [root@test ~]# |

　　指定日志文件保存多久

|  |  |
| --- | --- |
| 1  2  3 | [root@test ~]#journalctl --vacuum-time=1years  Vacuuming done, freed 0B of archived journals on disk.  [root@test ~]# |

　　logrotate：这个程序是一个日志文件管理工具。用来把旧的日志文件删除，并创建新的日志文件，我们把这一过程称为日志转储或滚动。它可以根据日志文件的大小，也可以根据其天数来转储，通常我们是设定定时计划任务去完成。

　　配置文件是/etc/logrotate.conf

　　主要的参数有：

　　　　compress： 通过gzip压缩转储以后的日志

　　　　nocompress：不需要压缩时，用这个参数

　　　　copytruncate：用于还在打开中的日志文件，把当前日志备份并截断

　　　　nocopytruncate：备份日志文件，但不截断

　　　　create mode owner group 转储文件，使用指定的文件模式创建新的日志文件

　　　　nocreate ：不建立新的日志文件

　　　　delaycompress和compress 一起使用时，转储的日志文件到下一次转储时才压缩

　　　　nodelaycompress：覆盖delaycompress选项，转储并压缩

　　　　errors address ：转储时代错误信息发送指定的email地址

　　　　ifempty：即使是空文件也转储，是缺省选项

　　　　notifempty：如果是空文件的话，不转储

　　　　mail address把转储的日志文件发送到指定的email地址

　　　　nomail：转储时不发送日志文件

　　　　olddir directory：转储后的日志文件放入指定的目录，必须和当前日志文件在同一个文件系统

　　　　noolddir：转储后的日志文件和当前日志文件放在同一个目录下

　　　　sharedscripts ： 运行postrotate脚本，作用是在所有日志都轮转后统一执行一次脚本。如果没有配置这个，那么每个日志轮转后都会执行一次脚本

　　　　prerotate/endscript ：在转储以前需要执行的命令可以放入这两个关键字中间，这两个关键字必须单独成行

　　　　postrotate/endscript：在转储以后需要执行的命令可以放入这两个关键字中间，这两个关键字必须单独成行

　　　　dateext ：使用当期日期作为命名格式

　　　　dateformat ：配合dateext使用，紧跟在下一行出现，定义文件切割后的文件名，必须配合dateext使用，只支持 %Y %m %d %s 这四个参数

　　　　daily：指定转储周期为每天

　　　　weekly：指定转储周期为每周

　　　　monthly：指定转储周期为每月

　　　　size：大小，指定日志超过多大时，就执行日志转储

　　　　rotate count：指定日志文件删除之前转储的次数，0指没有备份，5指保留5个备份

　　　　missingok：如果日志不存在，提示错误

　　　　nomissingok：如果日志不存在，继续下一次日志，不提示错误

　　默认配置文件

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37 | [root@test ~]#cat /etc/logrotate.conf  # see "man logrotate" for details  # rotate log files weekly  weekly    # keep 4 weeks worth of backlogs  rotate 4    # create new (empty) log files after rotating old ones  create    # use date as a suffix of the rotated file  dateext    # uncomment this if you want your log files compressed  #compress    # RPM packages drop log rotation information into this directory  include /etc/logrotate.d    # no packages own wtmp and btmp -- we'll rotate them here  /var/log/wtmp {      monthly      create 0664 root utmp          minsize 1M      rotate 1  }    /var/log/btmp {      missingok      monthly      create 0600 root utmp      rotate 1  }    # system-specific logs may be also be configured here.  [root@test ~]# |

　　说明：从上面的配置文件看，我们可以了解到/var/log/wtmp 这个日志文件是每一个月转储一次，并创建新的日志文件 其权限是0644 属主是root 属组是utmp ，日志文件小大超过1M，日志文件就会转储， 保留一个备份文件。这个文件定义了系统的软件日志文件的转储规则，其中include /etc/logrotate.d/表示将/etc/logrotate.d目录下的所有转储规则都导入配置文件中。这样一来就可以实现单独的应用可以用单独的配置文件存储。这样一来就很方便的实现了管理转储规则的目的。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57 | [root@test ~]#ll /etc/logrotate.d/  总用量 40  -rw-r--r--. 1 root root  91 4月  11 2018 bootlog  -rw-r--r--. 1 root root 160 9月  19 2018 chrony  -rw-r--r--. 1 root root 194 8月   6 21:44 httpd  -rw-r--r--. 1 root root 893 8月   8 19:49 mariadb  -rw-r--r--. 1 root root 243 5月  10 2019 nginx  -rw-r--r--. 1 root root 136 6月  10 2014 ppp  -rw-r--r--. 1 root root 115 8月   9 22:39 samba  -rw-r--r--. 1 root root 224 10月 18 23:48 syslog  -rw-r--r--. 1 root root 100 10月 31 2018 wpa\_supplicant  -rw-r--r--. 1 root root 103 11月  5 2018 yum  [root@test ~]#cat /etc/logrotate.d/chrony  /var/log/chrony/\*.log {      missingok      nocreate      sharedscripts      postrotate          /usr/bin/chronyc cyclelogs > /dev/null 2>&1 || true      endscript  }  [root@test ~]#cat /etc/logrotate.d/nginx  /var/log/nginx/\*log {      create 0644 nginx nginx      daily      rotate 10      missingok      notifempty      compress      sharedscripts      postrotate          /bin/kill -USR1 `cat /run/nginx.pid 2>/dev/null` 2>/dev/null || true      endscript  }    [root@test ~]#cat /etc/logrotate.d/syslog  /var/log/cron  /var/log/maillog  /var/log/messages  /var/log/secure  /var/log/spooler  {      missingok      sharedscripts      postrotate          /bin/kill -HUP `cat /var/run/syslogd.pid 2> /dev/null` 2> /dev/null || true      endscript  }  [root@test ~]#cat /etc/logrotate.d/yum  /var/log/yum.log {      missingok      notifempty      maxsize 30k      yearly      create 0600 root root  }  [root@test ~]# |

　　说明：/etc/logrotate.d/这个目录就是存放各种应用程序的日志文件转储规则

　　logrotate是基于cron来运行的，其脚本是/etc/cron.daily/logrotate，日志的转储是系统自动完成对，事实上我们运行logrotate会调用配置文件/etc/logrotate.conf，我们可以在/etc/logrotate.d目录下定义各种应用程序日志转储的规则，用来覆盖其logrotate的默认值。通常我们在测试自己写的转储规则时，我们可以用 logrotate -f /etc/logrotate.d/xxx，这条命令的的作用就是强制读取/etc/logrotate.d/xxx 来转储日志文件；以下是logrotate命令的用法

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | [root@test ~]#logrotate --help  用法: logrotate [OPTION...] <configfile>    -d, --debug               Don't do anything, just test (implies -v)    -f, --force               Force file rotation    -m, --mail=command        Command to send mail (instead of `/bin/mail')    -s, --state=statefile     Path of state file    -v, --verbose             Display messages during rotation    -l, --log=STRING          Log file    --version                 Display version information    Help options:    -?, --help                Show this help message    --usage                   Display brief usage message  [root@test ~]# |

　　说明：-d表示--debug,debug模式，测试配置文件是否有误，-f表示--force强制转储日志文件，-m指定压缩后的日志文件发送到邮箱地址，-s表示使用指定的状态文件，-v表示显示其转储过程。有了这个工具管理日志文件就很轻松，我们只需要定义其日志文件的转储规则即可。

　　五、启动网络日志服务，让rsyslog工作在tcp或者udp协议上，配置rsyslog成为日志服务器

　　　　1）rsyslog工作在tcp或者udp协议的514端口配置

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | [root@test ~]#grep -i "tcp" /etc/rsyslog.conf  # Provides TCP syslog reception  $ModLoad imtcp  $InputTCPServerRun 514  # Remote Logging (we use TCP for reliable delivery)  [root@test ~]#grep -i "udp" /etc/rsyslog.conf  # Provides UDP syslog reception  $ModLoad imudp  $UDPServerRun 514  [root@test ~]# |

　　说明：以上配置是将rsyslog配置成工作在udp 514端口上，此时配置好配置文件后重启服务，此服务器就成为了rsyslog日志服务器了，它可以帮助其他服务器记录日志。

　　　　2）重启rsyslog服务，在其客户机上配置rsyslog，让其日志发送给rsyslog服务器记录

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | [root@test ~]#systemctl restart rsyslog  [root@test ~]#ss -ntul  Netid State      Recv-Q Send-Q           Local Address:Port                          Peer Address:Port  udp   UNCONN     0      0                            \*:123                                      \*:\*  udp   UNCONN     0      0                    127.0.0.1:323                                      \*:\*  udp   UNCONN     0      0                            \*:514                                      \*:\*  udp   UNCONN     0      0                          ::1:323                                     :::\*  udp   UNCONN     0      0                           :::514                                     :::\*  tcp   LISTEN     0      100                  127.0.0.1:25                                       \*:\*  tcp   LISTEN     0      25                           \*:514                                      \*:\*  tcp   LISTEN     0      128                          \*:41319                                    \*:\*  tcp   LISTEN     0      50                           \*:3306                                     \*:\*  tcp   LISTEN     0      100                        ::1:25                                      :::\*  tcp   LISTEN     0      25                          :::514                                     :::\*  tcp   LISTEN     0      128                         :::41319                                   :::\*  tcp   LISTEN     0      128                         :::80                                      :::\*  [root@test ~]# |

　　说明：可以看到重启了服务后，514端口已经起来，接下来配置客户机的rsyslog,让其通过网络发送日志到日志服务器上

|  |  |
| --- | --- |
| 1  2  3 | [root@test-node1 ~]#grep "192.168.0.99" /etc/rsyslog.conf  \*.info;mail.none;authpriv.none;cron.none                @192.168.0.99  [root@test-node1 ~]# |

　　说明：以上配置的意思是除了mail ,authpriv,cron这三个以外的所有设施的info及info以上级别的日志都发往192.168.0.99记录，这里需要注意一点，一个“@”表示连接服务器是通过udp协议连接，日志通过udp协议传送，两个“@”表示连接服务器通过tcp去连接，日志通过tcp协议传送

　　　　3）重启客户机上的rsyslog服务，在服务器上查看客户机的日志

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | [root@test-node1 ~]#/etc/init.d/rsyslog restart  Shutting down system logger:                               [  OK  ]  Starting system logger:                                    [  OK  ]  [root@test-node1 ~]#logger "i am test-node1"  [root@test-node1 ~]#tail /var/log/messages  Dec 24 23:06:17 test kernel: cfg80211:   (57240000 KHz - 63720000 KHz @ 2160000 KHz), (N/A, 0 mBm), (N/A)  Dec 24 23:06:17 test kernel: EXT4-fs (sda1): mounted filesystem with ordered data mode. Opts:  Dec 24 23:06:17 test kernel: EXT4-fs (dm-2): mounted filesystem with ordered data mode. Opts:  Dec 24 23:06:17 test kernel: Adding 4128764k swap on /dev/mapper/VolGroup-lv\_swap.  Priority:-1 extents:1 across:4128764k  Dec 24 23:06:17 test kernel: sky2 eth0: enabling interface  Dec 24 23:06:17 test kernel: ADDRCONF(NETDEV\_UP): eth0: link is not ready  Dec 24 23:06:17 test kernel: sky2 eth0: Link is up at 1000 Mbps, full duplex, flow control both  Dec 24 23:06:17 test kernel: ADDRCONF(NETDEV\_CHANGE): eth0: link becomes ready  Dec 24 23:23:06 test kernel: Kernel logging (proc) stopped.  Dec 24 23:23:06 test rsyslogd: [origin software="rsyslogd" swVersion="5.8.10" x-pid="1471" x-info="http://www.rsyslog.com"] exiting on signal 15.  [root@test-node1 ~]# |

　　说明：可以看到客户机上没有记录日志了

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | [root@test ~]#tail /var/log/messages  Dec 24 21:43:07 test systemd: Started System Logging Service.  Dec 24 23:26:04 test systemd: Stopping System Logging Service...  Dec 24 23:26:04 test rsyslogd: [origin software="rsyslogd" swVersion="8.24.0-41.el7\_7.2" x-pid="16136" x-info="http://www.rsyslog.com"] exiting on signal 15.  Dec 24 23:26:04 test systemd: Stopped System Logging Service.  Dec 24 23:26:04 test systemd: Starting System Logging Service...  Dec 24 23:26:04 test rsyslogd: [origin software="rsyslogd" swVersion="8.24.0-41.el7\_7.2" x-pid="16359" x-info="http://www.rsyslog.com"] start  Dec 24 23:26:04 test rsyslogd: action '\*' treated as ':omusrmsg:\*' - please use ':omusrmsg:\*' syntax instead, '\*' will not be supported in the future [v8.24.0-41.el7\_7.2 try http://www.rsyslog.com/e/2184 ]  Dec 24 23:26:04 test systemd: Started System Logging Service.  Dec 24 23:26:04 test rsyslogd: error during parsing file /etc/rsyslog.conf, on or before line 76: warnings occured in file '/etc/rsyslog.conf' around line 76 [v8.24.0-41.el7\_7.2 try http://www.rsyslog.com/e/2207 ]  Dec 24 23:26:13 test-node1 qiuhom: i am test-node1  [root@test ~]# |

　　说明：在日志服务器上可以看到我们刚才的测试日志信息，这里需要说一下，我们客户端通过网络把日志发送给服务端，服务端里怎么储存要看服务端配置，服务端可以把它储存到数据库，储存到文件都可以。

　　六、rsyslog将日志记录于mysql中

　　1）准备数据库服务器

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28 | [root@test ~]#yum install mariadb  已加载插件：fastestmirror  Loading mirror speeds from cached hostfile   \* base: mirrors.aliyun.com   \* extras: mirrors.aliyun.com   \* updates: mirrors.aliyun.com  base                                                                                          | 3.6 kB  00:00:00  dockerrepo                                                                                    | 2.9 kB  00:00:00  epel                                                                                          | 5.3 kB  00:00:00  extras                                                                                        | 2.9 kB  00:00:00  updates                                                                                       | 2.9 kB  00:00:00  软件包 1:mariadb-5.5.64-1.el7.x86\_64 已安装并且是最新版本  无须任何处理  [root@test ~]#systemctl status mariadb  ● mariadb.service - MariaDB database server     Loaded: loaded (/usr/lib/systemd/system/mariadb.service; disabled; vendor preset: disabled)     Active: inactive (dead)  [root@test ~]#systemctl start mariadb  [root@test ~]#mysql  Welcome to the MariaDB monitor.  Commands end with ; or \g.  Your MariaDB connection id is 2  Server version: 5.5.64-MariaDB MariaDB Server    Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.    Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.    MariaDB [(none)]> |

　　说明：mariadb同mysql类似，yum安装mariadb 并启动服务即可

　　2）在mariadb server上授权rsyslog能连接至当前数据库服务器

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | MariaDB [(none)]> select user,host,password from mysql.user      -> ;  +------+-----------+----------+  | user | host      | password |  +------+-----------+----------+  | root | localhost |          |  +------+-----------+----------+  1 row in set (0.00 sec)    MariaDB [(none)]> grant all on Syslog.\* to 'rsyslog'@'%' identified by 'rsyslogpass';  Query OK, 0 rows affected (0.00 sec)    MariaDB [(none)]> select user,host,password from mysql.user;  +---------+-----------+-------------------------------------------+  | user    | host      | password                                  |  +---------+-----------+-------------------------------------------+  | root    | localhost |                                           |  | rsyslog | %         | \*3AABCFD2E87DD4D86B283A77A7B21E449FBA9AFA |  +---------+-----------+-------------------------------------------+  2 rows in set (0.00 sec)    MariaDB [(none)]> |

　　3）在rsyslog服务器上安装mysql模块相关的程序包rsyslog-mysql

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57 | [root@test ~]#yum install rsyslog-mysql  已加载插件：fastestmirror  Loading mirror speeds from cached hostfile   \* base: mirrors.aliyun.com   \* extras: mirrors.aliyun.com   \* updates: mirrors.aliyun.com  正在解决依赖关系  --> 正在检查事务  ---> 软件包 rsyslog-mysql.x86\_64.0.8.24.0-41.el7\_7.2 将被 安装  --> 正在处理依赖关系 rsyslog = 8.24.0-41.el7\_7.2，它被软件包 rsyslog-mysql-8.24.0-41.el7\_7.2.x86\_64 需要  --> 正在检查事务  ---> 软件包 rsyslog.x86\_64.0.8.24.0-34.el7 将被 升级  ---> 软件包 rsyslog.x86\_64.0.8.24.0-41.el7\_7.2 将被 更新  --> 解决依赖关系完成    依赖关系解决    =====================================================================================================================   Package                      架构                  版本                                源                      大小  =====================================================================================================================  正在安装:   rsyslog-mysql                x86\_64                8.24.0-41.el7\_7.2                   updates                 42 k  为依赖而更新:   rsyslog                      x86\_64                8.24.0-41.el7\_7.2                   updates                616 k    事务概要  =====================================================================================================================  安装  1 软件包  升级           ( 1 依赖软件包)    总下载量：659 k  Is this ok [y/d/N]: y  Downloading packages:  Delta RPMs disabled because /usr/bin/applydeltarpm not installed.  (1/2): rsyslog-mysql-8.24.0-41.el7\_7.2.x86\_64.rpm                                             |  42 kB  00:00:00  (2/2): rsyslog-8.24.0-41.el7\_7.2.x86\_64.rpm                                                   | 616 kB  00:00:00  ---------------------------------------------------------------------------------------------------------------------  总计                                                                                 858 kB/s | 659 kB  00:00:00  Running transaction check  Running transaction test  Transaction test succeeded  Running transaction    正在更新    : rsyslog-8.24.0-41.el7\_7.2.x86\_64                                                                 1/3    正在安装    : rsyslog-mysql-8.24.0-41.el7\_7.2.x86\_64                                                           2/3    清理        : rsyslog-8.24.0-34.el7.x86\_64                                                                     3/3    验证中      : rsyslog-8.24.0-41.el7\_7.2.x86\_64                                                                 1/3    验证中      : rsyslog-mysql-8.24.0-41.el7\_7.2.x86\_64                                                           2/3    验证中      : rsyslog-8.24.0-34.el7.x86\_64                                                                     3/3    已安装:    rsyslog-mysql.x86\_64 0:8.24.0-41.el7\_7.2    作为依赖被升级:    rsyslog.x86\_64 0:8.24.0-41.el7\_7.2    完毕！  [root@test ~]# |

　　说明：此插件必须在rsyslog服务器上安装，也就说你准备把那台服务器的日志记录到数据库中你就在那台日志服务器上安装此插件即可。

　　4）为rsyslog创建数据库及表

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40 | [root@test ~]#rpm -ql rsyslog-mysql  /usr/lib64/rsyslog/ommysql.so  /usr/share/doc/rsyslog-8.24.0/mysql-createDB.sql  [root@test ~]#mysql < /usr/share/doc/rsyslog-8.24.0/mysql-createDB.sql  [root@test ~]#mysql  Welcome to the MariaDB monitor.  Commands end with ; or \g.  Your MariaDB connection id is 4  Server version: 5.5.64-MariaDB MariaDB Server    Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.    Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.    MariaDB [(none)]> show databases;  +--------------------+  | Database           |  +--------------------+  | information\_schema |  | Syslog             |  | mysql              |  | performance\_schema |  | test               |  +--------------------+  5 rows in set (0.00 sec)    MariaDB [(none)]> use Syslog  Reading table information for completion of table and column names  You can turn off this feature to get a quicker startup with -A    Database changed  MariaDB [Syslog]> show tables;  +------------------------+  | Tables\_in\_Syslog       |  +------------------------+  | SystemEvents           |  | SystemEventsProperties |  +------------------------+  2 rows in set (0.00 sec)    MariaDB [Syslog]> |

　　说明：本人使用的是一台主机，若需要把远程主机上的日志记录到数据库中，导入sql脚本时需要指定用户名，主机以及密码

　　5）配置rsyslog将日志保存到mysql中

|  |  |
| --- | --- |
| 1  2  3  4 | [root@test ~]#grep "ommysql" /etc/rsyslog.conf  $ModLoad ommysql  \*.info;mail.none;authpriv.none;cron.none                :ommysql:192.168.0.99,Syslog,rsyslog,rsyslogpass  [root@test ~]# |

　　说明：在需要将日志记录到mysql的主机上编辑/etc/rsyslog.conf  将其$ModLoad ommysql 写在#### MODULES #### 下，将需要记录的日志设施和日志级别以及数据库的地址，数据库名，连接数据库用户名和密码写在#### RULES ####下，如上所示

　　6）重启rsyslog服务

|  |  |
| --- | --- |
| 1 | [root@test ~]#systemctl restart rsyslog |

　　到此rsyslog就会将日志记录到配置文件里指定的数据库里