Part 77
-
Syslog

Version: 2023-02-22

Introduction

In computing, syslog is a standard for message logging. It allows separation of the software that generates messages, the system that stores them, and the software that reports and analyzes them. Each message is labeled with a facility code, indicating the type of system generating the message, and is assigned a severity level.
Computer system designers may use syslog for system management and security auditing as well as general informational, analysis, and debugging messages. A wide variety of devices, such as printers, routers, and message receivers across many platforms use the syslog standard. This permits the consolidation of logging data from different types of systems in a central repository.
When operating over a network, syslog uses a client-server architecture where a syslog server listens for and logs messages coming from clients.

Syslog was developed in the 1980s by Eric Allman as part of the Sendmail project. It was readily adopted by other applications and has since become the standard logging solution on Unix-like systems. A variety of implementations also exist on other operating systems and it is commonly found in network devices, such as routers.
Syslog originally functioned as a de facto standard, without any authoritative published specification, and many implementations existed, some of which were incompatible. The Internet Engineering Task Force documented the status quo in RFC 3164 in August of 2001. It was standardized by RFC 5424 in March of 2009.

Configure Syslog Server

On Debian systems, Rsyslog is the default syslogd. Rsyslog package is already installed by default as well.

```
apt list rsyslog -a
```

Sample output;

```
Listing... Done
rsyslog/stable,now 8.2102.0-2 amd64 [installed]

rsyslog/stable 8.2102.0-2 i386
```

It is also started and and set to run on system boot. You can check status using the command below.

```
sudo systemctl status rsyslog
```

Sample output

```
rsyslog.service - System Logging Service
     Loaded: loaded (/lib/systemd/system/rsyslog.service; ena>
     Active: active (running) since Tue 2022-03-29 13:17:39 E>
TriggeredBy: syslog.socket
       Docs: man:rsyslogd(8)
             man:rsyslog.conf(5)
             https://www.rsyslog.com/doc/
   Main PID: 456 (rsyslogd)
      Tasks: 4 (limit: 7038)
     Memory: 8.2M
        CPU: 221ms
     CGroup: /system.slice/rsyslog.service
             └─456 /usr/sbin/rsyslogd -n -iNONE

Mar 29 13:17:38 debian systemd[1]: Starting System Logging Se>
Mar 29 13:17:39 debian systemd[1]: Started System Logging Ser>
Mar 29 13:17:39 debian rsyslogd[456]: imuxsock: Acquired UNIX>
Mar 29 13:17:39 debian rsyslogd[456]: [origin software="rsysl>
Mar 29 13:17:40 debian systemd[1]: rsyslog.service: Sent sign>
Mar 29 13:27:39 debian rsyslogd[456]: [origin software="rsysl>
```

Run the following command to open the syslog configuration file.

```
sudo nano /etc/rsyslog.conf
```

Next uncomment the lines below to configure the UDP and TCP protocols for log reception.

```
# provides UDP syslog reception
module(load="imudp")
input(type="imudp" port="514")

# provides TCP syslog reception
module(load="imtcp")
input(type="imtcp" port="514")
```

We'll create a new template that instructs the rsyslog server where to save incoming messages. Add the following after below TCP config.

```
$template Incoming-logs,"/var/log/%HOSTNAME%/%PROGRAMNAME%.log"
```

```
*.* ?Incoming-logs
```

It should look like the file below.

```
# provides UDP syslog reception
module(load="imudp")
input(type="imudp" port="514")

# provides TCP syslog reception
module(load="imtcp")
input(type="imtcp" port="514")

$template Incoming-logs,"/var/log/%HOSTNAME%/%PROGRAMNAME%.log"
*.* ?Incoming-logs

###########################
#### GLOBAL DIRECTIVES ####
###########################
```

After you've saved and closed the file, go ahead and inspect the configuration file using the following command.

```
sudo rsyslogd -N1 -f /etc/rsyslog.conf
```

For changes to take effect, use the command below to restart the rsyslog service.

```
sudo systemctl restart rsyslog
```

Check that the rsyslog service is listening on the ports specified.

```
sudo ss -tunlp | grep 514
```

Sample output

```
udp   UNCONN 0      0            0.0.0.0:514        0.0.0.0:*    users:(("rsyslogd",pid=3205,fd=6))
udp   UNCONN 0      0               [::]:514           [::]:*    users:(("rsyslogd",pid=3205,fd=7))
tcp   LISTEN 0      25           0.0.0.0:514        0.0.0.0:*    users:(("rsyslogd",pid=3205,fd=8))
tcp   LISTEN 0      25              [::]:514           [::]:*    users:(("rsyslogd",pid=3205,fd=9))
```

If you're using a firewall, enable `rsyslog` firewall port rules.

```
sudo ufw allow 514/tcp
sudo ufw allow 514/udp
```

After that, use the following command to restart your firewall.

```
sudo ufw reload
```

Configure Rsyslog Client

Set up your rsyslog client to send logs to a remote rsyslog server. Using the command below, open the configuration file.

```
sudo nano /etc/rsyslog.conf
```

Allow FQDN preservation by including the following in the config file.

```
$PreserveFQDN on
```

Configure a remote rsyslog server to send logs over UDP by adding the following line.

```
*.* @Rsysog-server-IP:514
```

Use double @ to send over TCP, as shown below.

```
*.* @@Rsysog-server-IP:514
```

Next set up a disk queue to save logs in case the rsyslog server goes down by adding the following lines.

```
$ActionQueueFileName queue
$ActionQueueMaxDiskSpace 1g
$ActionQueueSaveOnShutdown on
$ActionQueueType LinkedList
$ActionResumeRetryCount -1
```

Save and close the file. For changes to take effect, the rsyslog service must be restarted.

```
sudo systemctl restart rsyslog
```

View Log Files In Rsyslog Server

Our log is stored in the /var/log/remote-hostname/ directory according to the template we set earlier. Suppose the remote-hostname is “debian” and it is the app “systemd” on “debian” that emits syslog message, e.g.

```
cd /var/log/debian/
```

Type the command below to check the log, e.g. we’ll check logs for boot.

```
sudo tail -f /var/log/debian/systemd.log
```

Sample output

```
[  OK  ] Finished Permit User Sessions.
2022-03-29T13:40:04.966469+03:00 debian systemd[1]: rsyslog.service: Succeeded.
2022-03-29T13:40:04.967462+03:00 debian systemd[1]: Stopped System Logging Service.
2022-03-29T13:40:04.970601+03:00 debian systemd[1]: Starting System Logging Service...
2022-03-29T13:40:04.981814+03:00 debian systemd[1]: Started System Logging Service.
2022-03-29T13:41:35.401652+03:00 debian systemd[999]: Started Application launched by gnome-shell.
2022-03-29T13:41:40.778750+03:00 debian systemd[1]: Starting Hostname Service...
2022-03-29T13:41:40.843746+03:00 debian systemd[1]: Started Hostname Service.
2022-03-29T13:42:10.886061+03:00 debian systemd[1]: systemd-hostnamed.service: Succeeded.
2022-03-29T13:42:14.115768+03:00 debian systemd[999]: app-gnome-gnome\x2dcontrol\x2dcenter-3244.scope:
Succeeded.
2022-03-29T13:42:14.116048+03:00 debian systemd[999]: app-gnome-gnome\x2dcontrol\x2dcenter-3244.scope: Consumed
2.955s CPU time.
```

Sending syslog message in Python – Sample Code

```
import socket
from datetime import datetime as dt

SYSLOGFORMAT = { 'BSD': 0, 'IETF': 1 }

LEVEL = {'emerg':   0, 'alert':  1, 'crit': 2, 'err':   3,
         'warning': 4, 'notice': 5, 'info': 6, 'debug': 7
        }

FACILITY = { 'kern':    0, 'user':    1, 'mail':      2, 'daemon':  3,
             'auth':    4, 'syslog':  5, 'lpr':       6, 'news':    7,
             'uucp':    8,  'cron':   9, 'authpriv': 10, 'ftp':    11,
             'local0': 16, 'local1': 17, 'local2':   18, 'local3': 19,
             'local4': 20, 'local5': 21, 'local6':   22, 'local7': 23
           }


#--------------------------------------------------------------------------------
# syslog functions
#--------------------------------------------------------------------------------
def syslog(host='localhost', port=514, format=SYSLOGFORMAT['BSD'], level=LEVEL['notice'],
facility=FACILITY['daemon'], hostname = "myhost", appname = "myapp", message='message'):
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    data = ""
    dteNow      = dt.now()
    timestamp   = dteNow.strftime("%Y-%m-%d %H:%M:%S")
    if format == SYSLOGFORMAT['BSD']:
        timestamp = dteNow.strftime("%b %d %H:%M:%S")
        data = '<%d>%s %s %s: %s' % (level + facility*8, timestamp, hostname, appname, message)
    elif format == SYSLOGFORMAT['IETF']:
        timestamp = dt.strftime("%Y-%m-%dT%H:%M:%S.000Z")
        version = 1
        data = '<%d>%d %s %s %s - - - \xEF\xBB\xBF%s' % (level + facility*8, version, timestamp,
hostname, appname, message)

    if data != "":
        sock.sendto(str.encode(data), (host, port))
        sock.close()
#--------------------------------------------------------------------------------



syslog(host="192.168.1.42", port=514, format=SYSLOGFORMAT['BSD'], level=LEVEL['info'],
facility=FACILITY['daemon'], hostname='MeRasPi4B-VentiWater', appname='VentiWater', message="message
info")
syslog(host="192.168.1.42", port=514, format=SYSLOGFORMAT['BSD'], level=LEVEL['debug'],
facility=FACILITY['daemon'], hostname='MeRasPi4B-VentiWater', appname='VentiWater', message="message
debug")
```