Chapter 1

Configuring Logging

1.1 Understandig Logging In RHEL 7

Due to the introduction of Systemd several services of the older Unix System V now have a counter part in their systemd equivalent. Such is the case for rsyslogd and journald. The former is responsible for logging in System V systems while journald does it in systemd systems. However, due to the concern of backwards compatibility (i.e., being able to use tools written for older versions of Linux which may have used System V utilities), modern distros like RHEL 7 also have rsyslogd installed.

There can be thus, three different ways of logging application information in RHEL 7:

- Directly write to a log file somewhere - no standardized way of accessing logs.
- Write to Systemd's *Journald* - logs are accessible via **journalctl**
- Write to *rsyslogd* - logs are accessible via `/var/log`.

An important point to note here is that **rsyslog** is still the central logging authority, but journald simply adds features to the way that logging is organized. Thus, journald doesn't really replace rsyslog.

This however means that there's scope for confusion on part of the user (or admin) who's handling the system - to understand exactly where a certain programs might write it's logs to. Thus, we can connect the two together to show the same information.

1.2 Connecting Journald to Rsyslogd

We merely need to add a few lines of configuration to have both services report their own logs to each other:

```
1 # To connect Journald to rsyslogd:
2 ## In /etc/rsyslog.conf:
3 $ModLoad imuxsock
4 $OmitLocalLogging off
5 ## In /etc/rsyslog.d/listend.conf:
6 $SystemLogSocketName /run/systemd/journal/syslog
7
```

```
# To connect rsyslogd to journald:
## In /etc/rsyslog.conf:
$ModLoad omjournal
*.* :omjournal:
```

rsyslog messages are sent to **journald**, and vice versa. However, sending to journald is disabled by default in rsyslog.conf. To fix this we add the load the module omjournal (*output module journal*) using `$ModLoad omjournal`. Next, we use rsyslog's notation for indicating the facility, priority and destination. The statement `*.* :omjournal:` means for any facility and any priority, we want the log to be forwarded to *omjournal*.

Receiving from journal is enabled by default in `rsyslog.conf`. This is done via: `$ModLoad imuxsock` (Input Module UniX SOCKet), which instructs rsyslog to listen to a socket. Now, local logging has to be enabled using the `$OmitLocalLogging off` option. Finally, the socket name on which rsyslog will listen will have to be specified in the `/etc/rsyslog.d/listend.conf` file, and has to be set to the value `/run/systemd/journal/syslog`. Everything on the systemd end is already configured and needs no manual intervention. This completes the integration of the two.

1.2.1 Modules

Thus, the connection between rsyslog and journald is facilitated by modules. There are several types of modules, which can be identified and classified by:

Prefix	Type	Description
im*	Input Module	Source of information for the rsyslog journal; Loaded in `/etc/rsyslog.conf` and socket name specified in `/etc/rsyslog.d/listend.conf`.
om*	Output Module	Destination to which data from rsyslog will be sent; Configured in `/etc/rsyslog.conf`
		Other modules such as parser modules, message modification modules, etc.

Together these modules lets us manipulate the log messages any way we want.

1.2.2 Importing text files to log : httpd error log

To import the HTTPD error log to rsyslog, the following needs to be added to the file `/etc/rsyslog.conf`:

```
$ModLoad imfile
$InputFileName /var/log/httpd/error_log
$InputFileTag apache-error:
$InputFileState state-apache-error
$InputRunFileMontitor
```

This takes the error log read and maintained by apache and inserts the data into rsyslog. The `$InputRunFileMonitor` enables monitoring of the specified file.

1.2.3 Exporting data to an output module : exporting to a database

Since rsyslog writes the data to a simple text file, and for managing log information the ability to query is very important, we may choose to export the data to a database. Assuming we're using a MySQL database:

```
$ModLoad ommysql
$ActionOmmysqlServerPort 1234
*.* :ommysql:database-serverName,database-name,database-userid,database-password
```

The first line loads the MySQL Output module for rsyslog. Then, we define the server port on which the logs will be forwarded. Then, finally, we configure the output module using the *facility, priority :destination* method where we send everything (all facilities and every priority[`*.*`]) to the output module, while also providing the database details.

1.3 Setting up Remote Logging

It is via the use of modules that remote logging can be configured. This can be done using two types of protocols: TCP and UDP. UDP is the classical solution and is backwards compatible, but due to the very nature of the protocol, the message transfer isn't connection oriented. What this means is that messages may get lost in transit, and thus data loss may occur. Contrastingly, TCP only does data exchange after a connection has been established, and thus provides more reliable logging. So, if everything in our syslog configuration is compatible with TCP, then we should definitely opt for TCP syslog reception. Example usage for both are provided in a commented section in the `/etc/rsyslog.conf` file:

```
        # Provides UDP syslog reception
        #$ModLoad imudp
        #$UDPServerRun 514

        # Provides TCP syslog reception
        #$ModLoad imtcp
        #$InputTCPServerRun 514
```

The TCP syslog reception needs to be done exactly as the example states and so uncommenting the lines is all we need to do. The **imtcp** module enables the reception of log information via TCP connection and the `InputTCPServerRun` option chooses port 514 to use as incoming port for the messages (the IP is the IP of the server itself).

Now, for the other servers, the configuration has to be such that they send their own logging data to the same IP and port as we just configured. If our logging server is running on 10.0.50.11:514, then the *forwarding rule* configuration (an example of which is present in the `rsyslog.conf` file itself) becomes:

```
        # ### begin forwarding rule ###
        # The statement between the begin ... end define a SINGLE forwarding
        # rule. They belong together, do NOT split them. If you create multiple
        # forwarding rules, duplicate the whole block!
        # Remote Logging (we use TCP for reliable delivery)
        #
        # An on-disk queue is created for this action. If the remote host is
        # down, messages are spooled to disk and sent when it is up again.
        #$ActionQueueFileName fwdRule1 # unique name prefix for spool files
        #$ActionQueueMaxDiskSpace 1g   # 1gb space limit (use as much as possible)
```

```
#$ActionQueueSaveOnShutdown on # save messages to disk on shutdown
#$ActionQueueType LinkedList   # run asynchronously
#$ActionResumeRetryCount -1    # infinite retries if host is down
# remote host is: name/ip:port, e.g. 192.168.0.1:514, port optional
*.* @@10.0.50.11:514
# ### end of the forwarding rule ###
```

Note that the `@@` sign in the line `*.* @@10.0.50.11:514` statement signifies the use of TCP. If UDP is to be used instead, we replace it with a single `@` sign, thus making the statement: `*.* @10.0.50.11:514`. The forwarding rule asks rsyslogd to forward logs from any facility of any priority should be sent to the server over at 10.0.50.11 on port 514 via TCP. Now, after rsyslog service has been restarted on both the server and the client, remote logging should work.

An additional thing to remember is that on the logging server, i.e., the server accepting the logs, the port 514 needs to be unblocked for TCP traffic using:

```
# firewall-cmd --add-port=514/tcp --permanent
# firewall-cmd --reload
```

If SELinux blocks TCP traffic or the port isn't the standard port for remote logging, i.e., port 514, then the associated port needs to be given the right security context of `syslogd_port_t`. For the TCP port 514, the command to allow logging on the server is:

```
# semanage port -a -t syslogd_port_t -p tcp 514
```