Linux System Administration

PM-TRTNB319

Zsolt Schäffer, PTE-MIK

Legal note

These slides form an outline of the curriculum and contain multiple references to the official Red Hat training materials (codes RH124, RH134 and RH254), since students of this subject are also eligible for the named RH learning materials. Some diagrams and pictures originating from the RH courses will show up in this outline as there is an agreement in place which allows for our Faculty to teach this subject based on the Red Hat curriculum.

All other pictures and diagrams sourced from third parties are explicitly marked with a reference to the origin.

Legal note

All students of PTE-MIK who have officially taken the course „Linux System Administration" are eligible to download these slides from the Faculty's internal network, and are eligible for license keys for Red Hat System Administration online learning materials at rhlearn.gilmore.ca as part of their training program.

Do not share, redistribute, copy or otherwise offer these learning support materials, license keys or account information either for money or free of charge to anyone, as these materials contain **intellectual property.**

Unauthorized distribution is both against the law and university policy and will result in an in-campus inquiry, suing for damages and/or criminal prosecution by the University of Pécs, Red Hat Inc. and other parties.

Logging

Lecture 9

Chapter 1

Logs

- There are several reasons to keep logs in a computer system.
  - Troubleshooting: Daemons, as told earlier, don't communicate to a user directly, instead they write log messages. The occasional errors and warnings encountered by a daemon can be read from the logs.
  - Performance evaluation: It is useful to know how many request a daemon serves, what load the system is under in specific time periods, what the bottlenecks are.
  - Security/accountability: It is important to know what happens in a computer, who logged in and did what. Logs can also reveal where an attack originated from.

System logs

- Logging is a feature that also involves the kernel. Programs call a syscall to submit log entries. The entries are buffered in the kernel, and periodically collected and than saved on permanent storage by a logger daemon. Permanent storage can either be a local file or through network, involving another log collector daemon, which in turn usually also stores the logs on permanent storage.
- It is possible that some embedded Linux systems don't have a permanent storage to store the logs on. In this case the logs are stored in memory until the device is rebooted.
- It is possible to have multiple logging daemons, it is also possible for a daemon to record the logs both locally and forward via network.

Traditional system logs

- Traditional logs are **regular text files**, that can be viewed/searched with a pager or standard commands like: **less, cat**, **grep, tail,** …
- Traditional logs don't have standardized storage locations, filenames, methods or formats. There are distribution specific conventions though.
- One of the earliest logger daemons is **syslogd**. (By Eric Allmann, 1980)
- Syslogd defines 24 log facilities and 8 severity levels for entries.
- There are several standards for the text format, like RFC3146, RFC5424.

Log priorities

Number	Severity	Desciption
0	Emergency	Panic condition, system unusable.
1	Alert	Immediate correction required.
2	Critical	Critical condition.
3	Error	Non critical error, non urgent failure (for e.g. notify developer).
4	Warning	Not an error condition, but can indicate an upcoming failure (for e.g. FS is almost full).
5	Notice	Unusual or significant conditions that are not errors. No urgent action required.
6	Informational	Normal operation messages. Can be examined for throughput measurement.
7	Debug	Detailed information used for development and troubleshooting.

Log facilities

- The sources of logs are categorized (based on the emitting process) into facilities:
- Priority = Facility * 8 + Severity

0-kernel messages
1-user-level messages
2-mail system
3-system daemons
4-security/authorization
5-internal Syslog msgs.
6-line printer subsystem
7-network news subsys
8-UUCP subsystem
9-clock daemon
10-security/authorization
11-FTP daemon
12-NTP subsystem
13-log audit
14-log alert
15-clock daemon
16-23 locally def. use (localo-7)

Log format

- Log line format with example (RFC 5424)

2003-10-11T22:14:15.003Z mymachine.example .com su <47> ID: 73214 user.error – BOM 'su root' failed for lonvick on /dev/pts/8

- 2003-… Timestamp RFC8601
- mymachine… Hostname (FQDN)
- su Process name
- <47> Process ID (PID)
- ID: … Message ID
- user.error Syslog facility.severity
- BOM Message marker
- 'su root' fai… Message text

Rsyslog

- Rsyslog is the log handler service in many legacy Linux systems. Systemd has its own built-in logger called **systemd-journald**. Many systemd computers still have **rsyslog** enabled for compatibility reasons. This does not have any drawbacks, other than all messages are being logged and saved at least twice.
- Rsyslog has it's configuration settings in **/etc/rsyslog .conf** and **/etc/rsyslog.d** directory. The settings describe what source (facility) and priority of messages go to which file in the filesystem.

/etc/rsyslog.conf [excerpt]

```
kern.* /var/adm/kernel
kern.crit /dev/console
kern.info;kern.!err /var/adm/kernel-info
*.emerg @syslog-host
mail.* @mail-log-host
```

Conventional log locations

- There is no standardized scheme for sorting and storing logs. RH systems follow the conventions below (this is regulated in rsyslog.conf):
  - /var/log/secure login and auth. related messages
  - /var/log/cron msgs. of scheduled tasks
  - /var/log/boot.log startup related log entries
  - /var/log/maillog mail system log entries
  - /var/log/messages - all other messages that don't go to the first 4 files
- Some daemons implement their very own logging subsystem, unrelated to syslog or standard system calls and log features. En example would be the apache daemon, which writes several log files on it's own to the **/var/log/httpd** directory. (like access.log, error.log)

Logrotate

- Log files are constantly getting lines appended to. Therefore they can grow quiet large, and difficult to handle/backup/search.
- There is an automated mechanism for slicing large log files to smaller pieces, numbering them (by sequence or by date), and removing the oldest slices. This is called logrotate.
- Rotation can be manually triggered by issuing **logrotate**. The automatic scheduling of log rotation is described in **/etc/logrotate.conf** and **/etc/logrotate.d** directory. The logrotate command itself is run periodically by **cron**.
- Look at logrotate.conf, note the following directives: create, missingok, notifempty, size, weekly, daily, compress, rotate, dateext, …
- Look up **RH124 CH 10.1.** and **CH 10.2.** for further.

Analyzing logs

- Use **grep "*expression*" /var/log/messages** to look for entries matching "*expression*".
- Use for e.g. **tail -n 30 /var/log/secure** to print the last N lines of a file.
- Use **tail -f *filename*** to "follow" a file. This will print any new lines added to the file immediately. Press ctlr+c to terminate watching.
- Issue **logger -p *facility.severity* "Your own message"** to create a log entry from a script or command line.
- Use **dmesg** to display kernel messages only. You can use grep or tail or less also with dmesg (using pipes), like **dmesg | grep *something*** or **dmesg | tail -n 20**.
- Complete practice **RH124 CH 10.3.** (find log entries)

systemd-journald

- Systemd has it's own full-featured log service, called journald. It is enabled by default on systemd computers. It is possible to have other logger(s) running along with **systemd-journald**.
- Configuration is stored in **/etc/systemd/journald.conf** file. Journald has its own logrotate mechanism built in. (See SystemMaxUse=, SystemMaxFiles= and related directives.)
- One of the biggest arguments against systemd was that it breaks the convention of storing the logs in plain text format. Journald has a binary database, that can hold the entries more efficiently. It is also easily searchable by time, boot sequence number, systemd unit, …
- The tool for reading systemd logs is **journalctl**. This utility is built in all systemd systems, therefore the binary format does not have any practical drawbacks.

journalctl

- Use **journalctl -f** to follow the journal, apply **-n** switch to limit output to the last N lines of the log, just like with the tail command.
- The output of journalctl will always be automatically piped to a pager, like **less** (if it is present on the system).
- Use **journalctl --since "2016-03-19 15:30:00" --until "2016-03-19 20:00:00"** to narrow down to a time period.
- Use **journalctl -b -1** to see the messages belonging to the previous boot and run of the computer, use -b -2 instead to view the session 2 reboots ago, etc…
- Use **journalctl -p warn** to filter for priority, use **-u sshd.service** to filter to a specific systemd unit.
- Use advanced filters like _PID, _EXE, _SELINUX. (List them with journalctl -o verbose)

systemd-journald

- Issue **journalctl --rotate** to manually trigger a journal rotation.
- By default systemd autoselects the log destination. It first looks for a directory under /var/log/journal.
- If it is found, it will store logs there (/var is usually a disk mount – persistent logs).
- If not it will create /run/log/journal and write logs there. (/run is a tmpfs, it will be lost on reboot – volatile logs)
- Note the Storage= directive in journald.conf
- Read **RH124 CH 10.4.** and **10.6.** for further about journald and journalctl.
- Complete practice **RH 124 CH 10.5.** and **10.7.** (journalctl and persistent journals.)

System clock

Lecture 9

Chapter 2

Time and timezones

- Keeping the system clock accurate is important for many reasons. For e.g. the timestamp of log entries, the timestamp of files synchronized to other computers. Some security protocols require that the clock of computers involved are exactly synchronized, because message signatures have a validity time window.
- To set the computers local RTC chip use **timedate-ctl** command. Linux computers have their clocks set to UTC by default, and calculate local time based on RTC and timezone offset. Windows computer on the other hand tend to set the RTC chip to the local time.
- Use the following format to set the local time: **timedate-ctl set-time 9:00:00** Issue for e.g. **timedate-ctl set-timezone Europe/Budapest** to set timezone. Issue timedate-ctl list-timezones for a list.

NTP

- Issuing timedate-ctl without arguments will print current settings.
- NTP (Network Time Protocol) is a distributed network protocol for keeping the RTC accurate. NTP has a built-in algorithm for compensating varying network delay, that involves multiple network based time sources.
- To enable your system to automatically correct even a small skew or drift of the RTC: issue **timedate-ctl set-ntp true**.
- The daemon involved behind the scenes is called chronyd, this daemon talks the NTP "language". NTP uses a distributed peer-to-peer service model, any NTP synchronized peer can act as a time source itself.
- Issue **chronyc sources -v** to view the configured active time sources with the chrony client utility.

timedate-ctl

/etc/chrony.conf [excerpt]

```
server 1.europe.pool.ntp.org
server 2.europe.pool.ntp.org

maxupdateskew 5
driftfile /etc/chrony.drift
rtcdevice /dev/rtc

### ACTING AS AN NTP SERVER
allow 192.168/16
deny 192.168.100/24
```

- Read **RH124 CH 10.8.** for further details on timedate-ctl and chronyd.
- Complete practice **RH124 CH 10.9.** and **10.10.** (adjust system time, NTP)

Scheduling tasks

Lecture 9

Chapter 3

Scheduled tasks

- In Linux there are three basic concepts for settings execution of commands to a specific time. All of them is controlled by the **crond** daemon in a modern Linux.
- The **at** utility allows to postpone the execution of a command to a very specific time. This will be a one-time execution. The **crontab** mechanism allows for executing recurring tasks with a specific frequency at a specific time. The **anacron** mechanism is also used for scheduling an execution frequency for commands, but it will also consider missed events. This is useful for computers than are not powered-on at all times, but at irregular periods instead (like for e.g. notebooks or home PCs).
- Note that systemd has its own scheduling mechanism in the form of **.timer** units. Transient timers (systemd-run) are the equivalents of at.

at

- The **at** command runs a set of commands or a script file at a specific time, but only once. There are 3 ways to tell at what to do and several formats to tell when to do it:
  - You may pass a filename to be executed with the -f switch. E.g. **at noon -f start-backup.sh**
  - You may redirect a set of commands to it. E.g. **echo "poweroff" | at noon Monday**
  - You may also use the built in editor of at. Finish editing with ctrl+d. Example below:

shell command line

```
# at 9:30 PM Tue
at> if [ -e /root/tuesday_sql_mess ];then
at>  cleanup_sql.sh
at> fi
at> ^D
job 1 at Tue Nov 16 09:30:00 2017
```

at

- The at command has a very flexible time input, it can accept several formats, including natural English language time specifiers. A few examples follow:
- at noon…, at midnight…, at teatime…
- at tomorrow… (the same time tomorrow), at Friday… (same time on Friday), at next Monday… , at next week… (the same day and time as command is given, only next week)
- at 2:34 PM…, at Oct 10…, at 10/21/2017 … (absolute specifiers)
- at now…, at now + 30 minutes … , at now +3 days… (relative specifiers)
- Specifiers can also be combined: at 4PM + 2 days, at 2 PM next Thursday, at 8:23 PM 21/10/2017, at 7:45 tomorrow

at

- You can list currently queued at jobs with **atq** (alias to at -l), remove a job with **atrm** alias to (at -r).
- To list the command(s) in a given job, issue **at -c 4,** where the number is the job number according to atq output.
- The -q switch to at will tell the queue name. At has queues designated by letters starting from **a** to **z**, a is default. The higher the queue letter, the higher the niceness of the process.
- Jobs are stored in text files under **/var/spool/cron** directory.
- Read **RH134 CH 4.1** for further on at.
- Complete practice **RH134 CH 4.2.** (at, atq, atrm)

cron

- The crontab mechanism allows for scheduling recurring tasks.
- All users have their own crontab file. Use **crontab -l** command to print current crontab, use **crontab -e** to open the file in the default editor, like vi.
- Root can use crontab -u *username* to manage a specific users crontab. Regular users can only edit/list their own.
- There is a global crontab file (**/etc/crontab**), which has one additional column, that sets the username to the executed command.
- The global crontab file holds entries (run-parts) for running a set of scripts hourly, daily, weekly. You only have to copy or link executable files into /etc/cron.hourly , /etc/cron.daily, etc… directories respectively to have them run regularly.

crontab format

/etc/crontab

```
# Min  Hour  Day_of_M  Month  D_of_W  User  Command
  0    2     12        *      *       bob   /usr/bin/some
  30   8-16  *         *      1       root  /some/script1
  /5   8-12  1         *      *       joe   /some/script2
  0    1     3/7       1      *       jane  /some/script3
  /7   /2    13        *      5       root  /some/script4
```

- *Commands* can have any number of arguments. Before the command field: fields can be separated with any empty space characters like one or more space(s) or tab(s). Inside the command field: all chars (incl. spaces) are interpreted as they are part of the command line.
- Monday is considered the first day of week (1), Sunday can both be represented with 0 or 7.
- Intervals can be specified with a - (dash), repetitions can be specified with a / (slash). This is actually evaluated by calculating the division remainder. For e.g. /5 will run at 0, 5, 10, 15, etc… for e.g 3/7 will run at 3, 10, 17, 24, 31, etc..

anacron

- Anacron can start jobs after their actual scheduled time if an event was skipped because of a poweroff or hibernation state.
- The **/etc/anacrontab** file contains 4 fileds:
  - frequency in nr. of days.
  - delay in minutes from startup
  - time-tracking file's name
  - command
- Read **RH134 CH 4.3.** for further on cron and **4.5.** for anacron.
- Complete practice **RH134 CH 4.4.** and **4.6.** (crontab)

cleaning temporary files

- Systemd has a service for cleaning up temporary files from both persistent (/var/tmp) and volatile storage (/run). It is called systemd-tmpfiles-clean.service, a .timer unit with the same name ensures, that this service is ran 15 minutes after system startup and then every 24 hours.
- The tasks to do when such an event triggers is described in configuration files under /usr/lib/tmpfiles.d, /run/tmpfiles.d, /etc/tmpfiles.d. If these directories have the same filename then the etc instance overrides those listed before.
- Consider the d, D, Z, and L directives. An example line: **D /home/guest 0700 guest guest 1d**
- Read **RH134 CH 4.7.** for further on systemd's temp file management.
- Complete practice **RH134 CH 4.8.** (temp files)

FTP server

Lecture 9

Chapter 4

FTP

- File Transfer Protocol involves two channels: a control channel (for commands and replies) that is always initiated by the client to tcp port 21 on the server, and a data channel (bulk file content) that depends on mode.
- The active mode is the original FTP mode. In this mode the server initiates the data connection for file transfer from it's tcp/20 port to a dynamic unprivileged port (>1024) on the client side. This can be problematic if the client is behind a firewall or NAT, since this channel is not in the client side NAT table, because it was not initiated by the client.
- In case of the passive FTP mode the server offers a dynamic port above 1024 for the client to connect to. This complicates the server side firewall setup, but does not require any special measures from the client (customer).

vsftpd

- RH repositories only contain the **vsftpd** (Very Secure FTP daemon) as a standalone FTP service.
- Vsftpd **separates** the privileged parent process and spawns un-privileged child processes for client sessions.
- The sensitive operations are handled by the parent process. Children communicate their requests to the parent through a local socket, all requests are verified.
- Child processes are **chroot**-ed to the relevant directory.
- Configuration files can be found under /etc/vsftpd directory. The configuration consists of <directive>= <value> style lines and some other files, that contain lists. (lists of allowed users for e.g.)
- You can manage the service with systemd, the unit name is vsftpd.service. E.g: **systemctl start vsftpd**

vsftpd

- Let's see the most important configuration directives:
  - listen_address= tells what network IFs to listen on.
  - listen_port= defaults to standard ftp port (21)
  - max_clients= the max. number of simultanious connections
  - pasv_enable, pasv_max_port, pasv_min_port: passive mode settings
  - port_enable= allows active mode connections.
  - anon_max_rate= max bandwidth for anonym. users
  - dual_log_enable= allows for both syslog and simultaneous custom vsftp logging.
  - cmds_allowed= holds a list of commands that are allowed over ftp

vsftpd

- Let's see the most important configuration directives:
  - message_file= tells filename for directory messages
  - force_dot_files= show hidden files (starting with a .)
  - guest_enable= enables guest user
  - chroot_local_user, chroot_list_file= a file containing a list of chroot-ed users.
  - anon_upload_enable= allows anon. user to upload
  - ftp_username= the local user mapped to anonymous logins
  - local_enable= enable locally defined users to log in.
  - banner_file= text to display at client side on conn.
  - userlist_deny, userlist_file, userlist_enable= specifies the set of user that are allowed to log in.

SELinux and vsftpd

- Relevant SELinux types are **public_content_t** and **public_content_rw_t**, you should set the file context of FTP accessible directories accordingly. E.g.: **semanage fcontext -a -t public_content_rw_t "/myftp/pub(/.*)?"**
- Relevant SELinux booleans are:
  - allow_ftp_anon_write (allow ftp users to write to FS objects labeled with public_content_rw_t)
  - allow_ftpd_full_access (skips SELinux checks, DAC still in place)
  - allow_ftpd_use_nfs (allow accessing files that are mounted through NFS and labeled nfs_t)
  - ftp_home_dir (allow R/W access to authenticated users to their own home directories)

vsftpd practice

- This example shows how to create a directory that is writeable through ftp. This directory can also be served as read-only web content at the same time. (E.g. this is a way for users to upload their web pages to a server.)

Shell command line

```
# setsebool -P ftp_home_dir on
# mkdir -p /myftp/pub
# chown jane:root /myftp/pub
# chmod 775 /myftp/pub
# semanage fcontext -a -t public_content_t /myftp
# semanage fcontext -a -t public_content_rw_t
"/myftp/pub(/.*)?"
# restorecon -R /myftp/
# ls -dZ /myftp/pub
drwxrwxr-x. jane root unconfined_u:object_r:
                                default_t:s0 /myftp/pub/
# setsebool -P allow_ftpd_anon_write on
# systemctl start vsftpd
# systemctl enable vsftpd
```

vsftpd practice

- Now test the FTP server from a client computer.

Shell command line

```
~]$ ftp ftp.test.net
Connected to ftp.test.net (10.10.0.1).
220 (vsFTPd 2.1.0)
Name (localhost:username):jane
331 Please specify the password.
Password: thisisasecret
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd myftp
250 Directory successfully changed.
ftp> put ftpupload
local: ftpupload remote: ftpupload
227 Entering Passive Mode (127,0,0,1,241,41).
150 Ok to send data.
226 File receive OK.
ftp> quit
221 Goodbye.
```

Recommended reading

Lecture 9

Chapter 5

Relevant chapters in RH:

- RH124 CH10 (Logs)
- RH134 CH4 (Scheduling tasks)

Read by next lecture:

- RH 124 CH9 (SSH)
- RH254 CH10 (Apache/httpd)

Use your rhlearn.gilmore.ca login to access the learning materials.

Thank you for your attention!

Lecture 9, PM-TRTNB319, Linux System Administration

Zsolt Schäffer, PTE-MIK