Key Differences between RHEL 8 and RHEL 9

Chapter 1: Architectures

Red Hat Enterprise Linux 9 is distributed with the kernel version 5.14, which provides support for the following architectures at the minimum required version:

- **AMD and Intel 64-bit architectures (x86-64-v2)**
  Package suffix: x86_64

- **The 64-bit ARM architecture (ARMv8.0-A)**
  Package suffix: aarch64

- **IBM Power Systems, Little Endian (POWER9)**
  Package suffix: ppc64le

- **64-bit IBM Z (z14)**
  Package suffix: s390x

Package name convention:
`<PACKAGE_NAME>-<VER>.<SUB>[.<SUB>]-<RELEASE>[.EL?].<ARCH>.rpm`

```
abc-2-3.4-5.el9.s390x.rpm
```

Chapter 2: Repositories

Red Hat Enterprise Linux 9 is distributed through **two main** repositories, and are available in all RHEL subscriptions.

- **BaseOS**
  Provides the core set of the underlying OS functionality.
  Foundation for all installations.

- **AppStream**
  Additional user-space applications, runtime languages, and databases.

- **CodeReady Linux Builder**
  Provides additional packages used by developers.
  Packages are unsupported.

Both repositories are required for a basic RHEL installation, and are available with all RHEL subscriptions.

Application Streams

Multiple versions of user-space components are delivered as Application Streams and updated more frequently than the core operating system packages. This provides greater flexibility to customize RHEL without impacting the underlying stability of the platform or specific deployments.

Each Application Stream component has a given life cycle, either the same as RHEL 9 or shorter.

Life Cycle:

Red Hat Enterprise Linux
https://access.redhat.com/support/policy/updates/errata

Application Streams
https://access.redhat.com/support/policy/updates/rhel-app-streams-life-cycle

Availability formats:

- RPM Format
- extension to the RPM format called modules
- Software Collections
- Flatpaks

RHEL 9 improves the Application Streams experience by providing initial Application Stream versions that can be installed as RPM packages using the traditional dnf install command.

Some additional Application Stream versions will be distributed as modules with a shorter life cycle in future minor RHEL 9 releases. It is recommended to review the Red Hat Enterprise Linux Application Stream Lifecycle definitions for any content life cycle considerations.

Always determine what version of an Application Stream you want to install and make sure to review the Red Hat Enterprise Linux Application Stream Lifecycle first.

Content that needs rapid updating, such as alternate compilers and container tools, is available in rolling streams that will not provide alternative versions in parallel. Rolling streams can be packaged as RPMs or modules.

For information about Application Streams available in RHEL 9 and their application compatibility level, see the Package manifest. Application compatibility levels are explained in the Red Hat Enterprise Linux 9: Application Compatibility Guide document.

> **Important**
>
> Red Hat Enterprise Linux 9.0 ships without modules. Future versions of RHEL 9 might introduce additional content and later software versions as modules. Furthermore, starting with RHEL 9, you must manually specify default module streams, because they are no longer defined by default. You can define default module streams with configuration files in the `/etc/dnf/modules.defaults.d/` directory.

Module Streams

A module can have:

- Can have one or more ***module streams***, which hold different versions of the content and receive updates independently. Each stream:
  - Can have one or more **profiles**. A profile is a list of packages that will be installed together for a particular use case, such as for a server, client, development, etc.
- Can only have one of the streams enabled.

Manage Modules with DNF

Red Hat Enterprise Linux 9 supports modular features of Application Stream. To handle the modular content, you can use the `dnf module` command. Otherwise, the `dnf` command works with similar modules to regular packages.

See the following list for some important commands to manage modules:

- **`dnf module list`** : List the available modules with the module name, stream, profiles, and a summary.
- **`dnf module list module-name`** : List the module streams for a specific module and retrieve their status.
- **`dnf module info module-name`** : Display details of a module, including the available profiles and a list of the packages that the module installs. Running the

`dnf module info` command without specifying a module stream lists the packages that are installed from the default profile and stream. Use the *module-name:stream* format to view a specific module stream. Add the `--profile` option to display information about packages that each of the module's profiles installed.

- **`dnf module provides`** ***`package`*** : Display which module provides a specific package.

Lab: Exploring AppStream and New Application Versions

1. List all module streams.
   ```
   dnf module list
   ```

2. Display more information for the nodejs module streams.
   ```
   dnf module info nodejs
   ```

3. Check what happens when you enable the module stream.
   ```
   dnf list nodejs
   dnf module enable nodejs:18 -y
   dnf list nodejs
   dnf module reset nodejs -y
   dnf module enable nodejs:20 -y
   dnf list nodejs
   dnf module reset nodejs -y
   dnf list nodejs
   ```

Chapter 3: Installer

Anaconda activates network automatically for interactive installations

Anaconda now activates the network automatically when performing interactive installation, without requiring users to manually activate it in the network spoke. This update does not change the installation experience for Kickstart installations and installations using the `ip=` boot option.

New options to Lock root account and Allow root SSH login with password

RHEL 9 adds following new options to the root password configuration screen:

Lock root account: To lock the root access to the machine.
Allow root SSH login with password: To enable password-based SSH root logins.

During Kickstart installations, you can enable root access via SSH with password by using the `--allow-ssh` option of the `rootpw` Kickstart command. For more information, see rootpw (required).

Licensing, system, and user setting configuration screens have been disabled post standard installation

Previously, RHEL users configured Licensing, System (Subscription manager), and User Settings prior to `gnome-initial-setup` and `login` screens. Starting with RHEL 9, the initial setup screens have been **disabled by default** to improve user experience. If you must run the initial setup for user creation or license display, install the following packages based on the requirements.

To install initial setup packages:

```
# dnf install initial-setup initial-setup-gui
```

To enable initial setup after the next reboot of the system.

```
# systemctl enable initial-setup
```

Reboot the system to view initial setup.

For **Kickstart** installations, add `initial-setup-gui` to the packages section and enable the initial-setup service.

```
firstboot --enable
%packages
@^graphical-server-environment
initial-setup-gui
%end
```

The rhsm command for machine provisioning through Kickstart for Satellite is now available

The `rhsm` command replaces the `%post` scripts for machine provisioning on RHEL 9. The `rhsm` command helps with all provisioning tasks such as registering the system, attaching RHEL subscriptions, and installing from a Satellite instance.

New Kickstart command - timesource

The new `timesource` Kickstart command is optional and it helps to set NTP, NTS servers, and NTP pools that provide time data. It also helps to control enabling or disabling the NTP services on the system. The --ntpservers option from the timezone command has been deprecated and has been replaced with this new command.

Support for Anaconda boot arguments without inst. prefix is no longer available

Anaconda boot arguments without the `inst.` prefix have been deprecated since RHEL 7. Support for these boot arguments has been removed in RHEL 9. To continue using these options, use the `inst.` prefix

For example, to force the installation program to run in the text mode instead of the graphical mode, use the following option:

```
inst.text
```

Removed Kickstart commands and options

The following Kickstart commands and options have been removed from RHEL 9. Using them in Kickstart files causes an error.

- `device`
- `deviceprobe`
- `dmraid`
- `install` - use the subcommands or methods directly as commands
- `multipath`
- `bootloader --upgrade`
- `ignoredisk --interactive`
- `partition --active`
- `harddrive --biospart`
- `autostep`

Where only specific options and values are listed, the base command and its other options are still available and not removed.

Removed boot options

The following boot options have been removed from Red Hat Enterprise Linux:

- `inst.zram`

  RHEL 9 does not support the zram service. See the zram-generator(8) man page for more information.

- `inst.singlelang`

  The single language mode is not supported on RHEL 9.

- `inst.loglevel`

  The log level is always set to debug.

Chapter 4: SELinux

Support for disabling SELinux through /etc/selinux/config has been removed

With the RHEL 9.0 release, support for disabling SELinux through the SELINUX=disabled option in the /etc/selinux/config file has been removed from the kernel. When you disable SELinux only through /etc/selinux/config, the system starts with SELinux enabled but with no policy loaded, and SELinux security hooks remain registered in the kernel. This means that SELinux disabled by using /etc/selinux/config still requires some system resources, and you should instead disable SELinux by using the kernel command line in all performance-sensitive scenarios.

Furthermore, the Anaconda installation program and the corresponding man pages have been updated to reflect this change. This change also enables read-only-after-initialization protection for the Linux Security Module (LSM) hooks.

If your scenario requires disabling SELinux, add the `selinux=0` parameter to your kernel command line.

See the Remove support for SELinux run-time disable Fedora wiki page for more information.

Lab 3: Disable SELinux

1. Append selinux=0 to GRUB_CMDLINE_LINUX in /etc/default/grub
   ```
   sudo vi /etc/default/grub
   ```

2. Verify the contents updated.
   ```
   cat /etc/default/grub
   GRUB_CMDLINE_LINUX=”.... selinux=0”
   GRUB_DEFAULT=saved
   ```

3. Generate new configuration.
   ```
   sudo grub2-mkconfig -o /boot/grub2/grub.cfg
   ```

4. Reboot to verify selinux is disabled.
   ```
   sudo reboot
   getenforce

   IF this does not work. Use grubby to update the grubenv file
   sudo grubby --update-kernel ALL --args selinux=0
   ```

Additional services confined in the SELinux policy

The RHEL 9.3 release added additional rules to the SELinux policy that confine the following systemd services:

- qat
- systemd-pstore
- boothd
- fdo-manufacturing-server
- fdo-rendezvous-server
- fdo-client-linuxapp
- fdo-owner-onboarding-server

As a result, these services do not run with the `unconfined_service_t` SELinux label anymore, and run successfully in SELinux enforcing mode.

The glusterd SELinux module moved to a separate glusterfs-selinux package

With this update, the glusterd SELinux module is maintained in the separate `glusterfs-selinux` package. The module is therefore no longer part of the `selinux-policy` package. For any actions that concern the glusterd module, install and use the `glusterfs-selinux` package.

Chapter 5: Containers

The container-tools meta-package is now available

The *container-tools* RPM meta-package, which includes Podman, Buildah, Skopeo, CRIU, Udica, and all required libraries, is available in RHEL 9. The stable streams are not available on RHEL 9. To receive stable access to Podman, Buildah, Skopeo, and others, use the RHEL EUS subscription.

Improved control group performance

The previous version of control groups, cgroup version 1 (cgroup v1), caused performance problems with a variety of applications. The latest release of control groups, cgroup version 2 (cgroup v2) enables system administrators to limit resources for any application without causing performance problems.

In RHEL 9, the new version of control groups, cgroups v2, is enabled by default.

Podman now supports secure short names

Short-name aliases for images can now be configured in the *registries.conf* file in the `[aliases]` table. The short-names modes are:

- ***Enforcing***: If no matching alias is found during the image pull, Podman prompts the user to choose one of the unqualified-search registries. If pulled successfully, Podman automatically records a new short-name alias in:
    - `$HOME/.cache/containers/short-name-aliases.conf` (rootless)
    - `/var/cache/containers/short-name-aliases.conf` (root user).

  If the user cannot be prompted (for example, stdin or stdout are not a TTY), Podman fails. Note that the `short-name-aliases.conf` file has precedence over `registries.conf` file if both specify the same alias. The enforcing mode is **default** in RHEL 9.

- ***Permissive***: Similar to enforcing mode, but Podman does not fail if the user cannot be prompted. Instead, Podman searches in all unqualified-search registries in the given order. Note that no alias is recorded. The permissive mode is **default** in RHEL 8.

Example:

```
unqualified-search-registries=["registry.fedoraproject.org", "quay.io"]
[aliases]
"fedora"="registry.fedoraproject.org/fedora"
short-name-mode = "enforcing"
```

Default OCI runtime change

The crun OCI runtime is now available for the container-tools:rhel8 module. The crun container runtime supports an annotation that enables the container to access the rootless user's additional groups. This is useful for container operations when volume mounting in a directory where setgid is set, or when the user only has group access.

- The default container runtime in RHEL 8 is runc.
- The default container runtime in RHEL 9 is crun.

Running RHEL 9 containers on a RHEL 7 host is not supported.

For more information, see Red Hat Enterprise Linux Container Compatibility Matrix.

Default network stacks

Default network stack for Podman:

RHEL 8: **CNI**

RHEL 9: **Netavark**

If you perform an in-place upgrade from RHEL 8 to RHEL 9, Podman's network stack is set as:

Netavark if the network_backend parameter in the configuration file `/etc/containers/containers.conf` is not set or if you manually upgraded Podman's network stack in RHEL 8 to Netavark.

CNI if there are containers, images, pods, or networks presented when Podman is first run after an upgrade. You can then manually upgrade to the new Netavark network stack. CNI network stack is deprecated and will be removed in future RHEL release.

Switching network stack from CNI to Netavark

https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/9/html/building_running_and_managing_containers/assembly_setting-container-network-modes_building-running-and-managing-containers#proc_switching-the-network-stack-from-cni-to-netavark_assembly_setting-container-network-modes

Switching the network stack from Netavark to CNI

https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/9/html/building_running_and_managing_containers/assembly_setting-container-network-modes_building-running-and-managing-containers#proc_switching-the-network-stack-from-netavark-to-cni_assembly_setting-container-network-modes

Red Hat recommends explicitly specifying the network_backend parameter to ensure that the correct backend is selected.

Warning

You cannot migrate the existing containers to a different network stack system using the `podman container checkpoint` and the `podman container restore` commands. If you want to switch from the CNI network stack to the Netavark network stack, recreate the container from the container image.

Lab: Creating an Application Server Container

Refer to podman-lab.txt

Chapter 6: Performance

Performance Co-Pilot rebased to version 6.0

Starting in RHEL 9.2, **Performance Co-Pilot** (PCP) has been updated to version 6.0. Notable improvements include:

1. Version 3 PCP archive support:
   This includes support for instance domain change-deltas, Y2038-safe timestamps, nanosecond-precision timestamps, arbitrary timezones support, and 64-bit file offsets used throughout for larger (beyond 2GB) individual volumes.

   This feature is currently opt-in via the `PCP_ARCHIVE_VERSION` setting in the `/etc/pcp.conf` file.

   Version 2 archives remain the default.

2. Only OpenSSL is used throughout PCP. Mozilla NSS/NSPR use has been dropped:

   This impacts `libpcp`, `PMAPI` clients and `PMCD` use of encryption. These elements are now configured and used consistently with `pmproxy` HTTPS support and `redis-server`, which were both already using OpenSSL.

3. New nanosecond precision timestamp `PMAPI` calls for `PCP` library interfaces that make use of timestamps.

   These are all optional, and full backward compatibility is preserved for existing tools

4. The following tools and services have been updated:
   `pcp2elasticsearch` Implemented authentication support.
   `pcp-dstat` Implemented support for the `top-alike` plugins.
   `pcp-htop` Updated to the latest stable upstream release.
   `pmseries` Added `sum`, `avg`, `stdev`, `nth_percentile`, `max_inst`, `max_sample`, `min_inst` and `min_sample` functions.
   `pmdabpf` Added CO-RE (Compile Once - Run Everywhere) modules and support for AMD64, Intel 64-bit, 64-bit ARM, and IBM Power Systems.
   `pmdabpftrace` Moved example autostart scripts to the `/usr/share` directory.
   `pmdadenki` Added support for multiple active batteries.
   `pmdalinux` Updates for the latest `/proc/net/netstat` changes.
   `pmdaopenvswitch` Added additional interface and coverage statistics.
   `pmproxy` Request parameters can now be sent in the request body.
   `pmieconf` Added several `pmie` rules for Open vSwitch metrics.
   `pmlogger_farm` Added a default configuration file for farm loggers.
   `pmlogger_daily_report` Some major efficiency improvements.

The sadf(1) tool now generates PCP archives from native sadc(1) archives

Starting in RHEL 9, the `sadf(1)` tool provided by the `sysstat` package can generate PCP archives from native `sadc(1)` archives.

By default, when the `-` flag is used with `sadc(1)`, `sadc(1)` writes the data to the standard system activity daily data file. This file is named `saDD` and is located in the `/var/log/sa` directory by default. Conversely, when no input datafile is specified, the `sadf(1)` tool uses the standard system activity daily data file to generate archives. Pass numbers as an argument to tell `sadf(1)` to generate archives from data recorded that specified number of days in the past

- To generate a PCP archive from an `sadc(1)` archive recorded 2 days ago, run:

  # sadf -l -O pcparchive=/tmp/recording -2

- To show a list of metrics in the PCP archive generated from an `sadc(1)` archive archive, run:

  $ pminfo --archive /tmp/recording
  Disk.dev.avactive
  Disk.dev.read
  Disk.dev.write
  Disk.dev.blkread
  [...]

- To show the timespace of the archive and hostname of the PCP archive generated from an `sadc(1)` archive

  ```
  $ pmdumplog --label /tmp/recording
  Log Label (Log Format Version 2)
  Performance metrics from host shard
        commencing Tue Jul 20 00:10:30.642477 2021
        ending     Wed Jul 21 00:10:30.222176 2021
  ```

- You can then analyze the PCP archive generated from an `sadc(1)` archive by using PCP commands, for example:

  ```
  $ pmchart --archive /tmp/recording
  ```

New PCP PMDA - `pmdabpf`

RHEL 9 is distributed with the `pcp-pmda-bpf` package, which provides the `pmdabpf` Performance Co-Pilot (PCP) Performance Metric Domain Agent (PMDA).

The `pmdabpf` PMDA extracts live performance data from `eBPF` programs utilizing `BPF CO-RE` (Compile Once - Run Everywhere), that is `libbpf` and `BTF`.

Chapter 7: Security

CIS and DISA STIG profiles provided as DRAFT

The profiles based on benchmarks from the *Center for Internet Security (CIS)* and *Defence Industry Security Association Security Technical Implementation Guides (DISA STIG)* are provided as DRAFT because the issuing authorities have not yet published an official benchmark for RHEL 9. In addition, the OSSP profile is in DRAFT because it is being implemented.

For a complete list of profiles available in RHEL 9, see SCAP Security Guide profiles supported in RHEL 9.

OpenSCAP no longer supports SHA-1 and MD5

Due to removal of **SHA-1** and **MD5** hash functions in Red Hat Enterprise Linux 9, support for OVAL `filehash_test` has been removed from OpenSCAP. Also, support for SHA-1 and MD5 hash functions has been removed from OVAL `filehash58_test` implementation in OpenSCAP. As a result, OpenSCAP evaluates rules in SCAP content that use the OVAL `filehash_test` as `notchecked`. In addition, OpenSCAP returns `notchecked` also when evaluating OVAL `filehash58_test` with the `hash_type` element within `filehash58_object` set to SHA-1 or MD5.

To update your OVAL content, rewrite the affected SCAP content so that it uses `filehash58_test` instead of `filehash_test` and use one of `SHA-224, SHA-256, SHA-384, SHA-512` in the `hash_type` element within `filehash58_object`.

OpenSCAP uses the data stream file instead of the XCCDF file

The SCAP source data stream file (`ssg-rhel9-ds.xml`) contains all the data that in previous versions of RHEL were contained in the XCCDF file (`ssg-rhel9-xccdf.xml`). The SCAP source data stream is a container file that includes all the components (XCCDF, OVAL, CPE) needed to perform a compliance scan. Using the SCAP source data stream instead of XCCDF has been recommended since RHEL 7. In previous versions of RHEL, the data in the XCCDF file and SCAP source data stream was duplicated. In RHEL 9, this duplication is removed to reduce the RPM package size. If your scenario requires using separate files instead of the data stream, you can split the data stream file by using this command:

```
# oscap ds sds-split /usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml
output_directory.
```

Crypto-policies, RHEL core cryptographic components, and protocols

Continuing SHA-1 deprecation

In RHEL 9, SHA-1 usage for signatures is restricted in the DEFAULT system-wide cryptographic policy. Except for HMAC, SHA-1 is no longer allowed in TLS, DTLS, SSH, IKEv2, DNSSEC, and Kerberos protocols. Individual applications not controlled by the RHEL system-wide crypto policies are also moving away from using SHA-1 hashes in RHEL 9.

If your scenario requires the use of SHA-1 for verifying existing or third-party cryptographic signatures, you can enable it by entering the following command:

```
# update-crypto-policies --set DEFAULT:SHA1
```

Alternatively, you can switch the system-wide crypto policies to the LEGACY policy. Note that LEGACY also enables many other algorithms that are not secure. See the Re-enabling SHA-1 section in the RHEL 9 Security hardening document for more information.

For solutions of compatibility problems with systems that still require SHA-1, see following Red Hat Knowledgebase solutions:

SSH from RHEL 9 to RHEL 6 systems does not work
Packages signed with SHA-1 cannot be installed or upgraded
Failed connection with SSH servers and clients that do not support the 'server-sig-algs' extension
DNSSEC records signed with RSASHA1 fail to verify

Algorithms disabled in all policy levels

The following algorithms are disabled in the LEGACY, DEFAULT and FUTURE crypto policies provided with RHEL 9:

TLS older than version 1.2 (since RHEL 9, was < 1.0 in RHEL 8)
DTLS older than version 1.2 (since RHEL 9, was < 1.0 in RHEL 8)
DH with parameters < 2048 bits (since RHEL 9, was < 1024 bits in RHEL 8)
RSA with key size < 2048 bits (since RHEL 9, was < 1024 bits in RHEL 8)
DSA (since RHEL 9, was < 1024 bits in RHEL 8)
3DES (since RHEL 9)
RC4 (since RHEL 9)
FFDHE-1024 (since RHEL 9)

DHE-DSS (since RHEL 9)
Camellia (since RHEL 9)
ARIA
SEED
IDEA
Integrity-only cipher suites
TLS CBC mode cipher suites using SHA-384 HMAC
AES-CCM8
All ECC curves incompatible with TLS 1.3, including secp256k1
IKEv1 (since RHEL 8)
NSEC3DSA in the BIND configuration (since RHEL 9.2)

> Warning
>
> If your scenario requires a policy that has been disabled, you can enable it by applying a custom cryptographic policy or by an explicit configuration of individual applications, but the resulting configuration will not be supported.

Changes to TLS

In RHEL 9, TLS configuration is performed using the system-wide cryptographic policies mechanism. TLS versions below 1.2 are not supported anymore. DEFAULT, FUTURE and LEGACY cryptographic policies allow only TLS 1.2 and 1.3. See Using system-wide cryptographic policies for more information.

The default settings provided by libraries included in RHEL 9 are secure enough for most deployments. The TLS implementations use secure algorithms where possible while not preventing connections from or to legacy clients or servers. Apply hardened settings in environments with strict security requirements where legacy clients or servers that do not support secure algorithms or protocols are not expected or allowed to connect.

Lab: Checking system-wide cryptographic policy

1. Check current cryptographic policy.
   ```
   update-crypto-policies --show
   ```

2. Set the new cryptographic policy.
   ```
   sudo update-crypto-policies --set LEGACY
   ```

3. Restart the system.

```
sudo reboot
```

4. Verify it is changed, then change it back to DEFAULT and reboot.
```
update-crypto-policies --show
sudo update-crypto-policies --set DEFAULT
sudo reboot
```

The Extended Master Secret TLS Extension is now enforced on FIPS-enabled systems

With the release of the RHSA-2023:3722 advisory, the TLS Extended Master Secret (EMS) extension (RFC 7627) is mandatory for TLS 1.2 connections on FIPS-enabled RHEL 9 systems. This is in accordance with FIPS-140-3 requirements. TLS 1.3 is not affected.

Legacy clients that do not support EMS or TLS 1.3 now cannot connect to FIPS servers running on RHEL 9. Similarly, RHEL 9 clients in FIPS mode cannot connect to servers that only support TLS 1.2 without EMS. This in practice means that these clients cannot connect to servers on RHEL 6, RHEL 7 and non-RHEL legacy operating systems. This is because the legacy 1.0.x versions of OpenSSL do not support EMS or TLS 1.3.

SCP not supported in RHEL 9

The secure copy protocol (SCP) protocol is no longer supported because it is difficult to secure. It has already caused security issues, for example CVE-2020-15778 . In RHEL 9, SCP is replaced by the SSH File Transfer Protocol (SFTP) by default.

> Warning
>
> By default, SSH cannot connect from RHEL 9 systems to older systems (for example, RHEL 6) or from older systems to RHEL 9. This is because the cryptographic algorithms used in older versions are now considered insecure. If your scenario requires connecting with older systems, you can either use the ECDSA and ECDH algorithms as keys on the legacy system or use the legacy cryptographic policy on the RHEL 9 system. For additional details, see the solutions SSH from RHEL 9 to RHEL 6 systems does not work and Failed connection with SSH servers and clients that do not support the server-sig-algs extension.

OpenSSH servers now use `/etc/ssh/sshd_config.d/` instead of the `CRYPTO_POLICY=` line in `/etc/sysconfig/sshd`

In RHEL 9, if you want to opt out of your SSH server from following the system-wide cryptographic policies, you must use the new /etc/ssh/sshd_config.d/ drop-in directory. The CRYPTO_POLICY= directive in the /etc/sysconfig/sshd configuration file used for specifying allowed cryptographic algorithms and ciphers in RHEL 8 is now ignored. See the Examples of opting out of the system-wide cryptographic policies section in the Security hardening document and the sshd ignores Ciphers, MACs, and KexAlgorithms in /etc/ssh/sshd_config on RHEL 9 solution (Red Hat Knowledgebase) for more information.

Interoperability of FIPS:OSPP hosts impacted due to CNSA 1.0

The `OSPP` subpolicy has been aligned with Commercial National Security Algorithm (CNSA) 1.0. This affects the interoperability of hosts that use the FIPS:OSPP policy-subpolicy combination, with the following major aspects:

- Minimum RSA key size is mandated at 3072 bits.
- Algorithm negotiations no longer support AES-128 ciphers, the secp256r1 elliptic curve, and the FFDHE-2048 group.

OpenSSH root password login disabled by default

The default configuration of OpenSSH in RHEL 9 disallows users to log in as root with a password to prevent attackers from gaining access through brute-force attacks on passwords.

OpenSSH further enforces SHA-2

As part of the effort to migrate further from the less secure SHA-1 message digest for cryptographic purposes, the following changes were made in OpenSSH:

- Added a check on sshd startup whether using SHA-1 is configured on the system. If it is not available, OpenSSH does not try to use SHA-1 for operations. This eliminates loading DSS keys when they are present and also enforces advertising rsa-sha2 combinations when they are available.
- On SSH private key conversion, OpenSSH explicitly uses SHA-2 for testing RSA keys.
- When SHA-1 signatures are unavailable on the server side, sshd uses SHA-2 to confirm host key proof. This might be incompatible with clients on RHEL 8 and earlier versions.
- When the SHA-1 algorithm is unavailable on the client side, OpenSSH uses SHA-2.

- On the client side, OpenSSH permits SHA-2-based key proofs from the server when SHA-1 was used in key proof request or when the hash algorithm is not specified (assuming default). This is aligned with the already present exception for RSA certificates, and allows connecting by using modern algorithms when supported.

GnuTLS requires EMS with TLS 1.2 in FIPS mode

To comply with the FIPS-140-3 standard, GnuTLS servers and clients require the Extended Master Secret (EMS) extension (RFC 7627) for all TLS 1.2 connections negotiated in FIPS mode. If your scenario requires preserving compatibility with older servers and clients that do not support EMS and you cannot use TLS 1.3, you can apply the NO-ENFORCE-EMS system-wide cryptographic subpolicy:

```
# update-crypto-policies --set FIPS:NO-ENFORCE-EMS
```

Warning

If you allow TLS 1.2 connections without EMS, your system no longer meets the FIPS-140-3 requirements.

GnuTLS no longer supports TPM 1.2

The GnuTLS library no longer supports the Trusted Platform Module (TPM) 1.2 technology. Your applications using TPM through the GnuTLS API must support TPM 2.0.

GnuTLS support for GOST has been removed

In RHEL 8, the GOST ciphers have been disabled through the system-wide cryptographic policies. In RHEL 9, support for these ciphers has been removed from the GnuTLS library.

cyrus-sasl now uses GDBM instead of Berkeley DB

The cyrus-sasl package is now built without the libdb dependency, and the sasldb plugin uses the GDBM database format instead of Berkeley DB. To migrate your existing Simple Authentication and Security Layer (SASL) databases stored in the old Berkeley DB format, use the cyrusbdb2current tool with the following syntax:

```
$ cyrusbdb2current <sasldb_path> <new_path>
```

NSS now enforce EMS in FIPS mode

The Network Security Services (NSS) libraries now contain the TLS-REQUIRE-EMS policy to require the Extended Master Secret (EMS) extension (RFC 7627) for all TLS 1.2 connections as mandated by the FIPS 140-3 standard. NSS use the new policy when the system-wide cryptographic policies are set to FIPS.

If your scenario requires interoperating with legacy systems without support for EMS or TLS 1.3, you can apply the NO-ENFORCE-EMS system-wide cryptographic subpolicy. Such a change violates the FIPS-140-3 requirements.

NSS no longer support DBM and pk12util defaults changed

The Network Security Services (NSS) libraries no longer support the DBM file format for the trust database. In RHEL 8, the SQLite file format became the default format, and the existing DBM databases were opened on read-only mode and automatically converted to SQLite. Before you upgrade to RHEL 9, update all trust databases from DBM to SQLite.

See the Updating NSS databases from DBM to SQLite procedure for detailed instructions.

NSS pk12util no longer uses DES-3 and SHA-1 by default

The pk12util tool now uses the AES and SHA-256 algorithms instead of DES-3 and SHA-1 by default when exporting private keys.

Note that SHA-1 is disabled by the default system-wide cryptographic policy for all signatures in RHEL 9.

NSS no longer support RSA keys shorter than 1023 bits

The update of the Network Security Services (NSS) libraries changes the minimum key size for all RSA operations from 128 to 1023 bits. This means that NSS no longer perform the following functions:

- Generate RSA keys shorter than 1023 bits.
- Sign or verify RSA signatures with RSA keys shorter than 1023 bits.
- Encrypt or decrypt values with RSA key shorter than 1023 bits.

OpenSSL ENGINE extension API is not supported in FIPS mode

The legacy extension system to OpenSSL, the ENGINE API, is not compatible with the new provider API. Therefore, applications that depend on functionality provided by

OpenSSL engines, such as the openssl-pkcs11 and openssl-ibmca modules, cannot be used in FIPS mode.

FIPS mode in OpenSSL must be enabled to work correctly

If you are using non-default values in the openssl.cnf configuration file with FIPS mode enabled, and especially when using a third-party FIPS provider, add fips=yes to the openssl.cnf file.

OpenSSL does not accept explicit curve parameters in FIPS mode

Elliptic curve cryptography parameters, private keys, public keys, and certificates that specified explicit curve parameters no longer work in FIPS mode. Specifying the curve parameters using ASN.1 object identifiers, which use one of the FIPS-approved curves, still works in FIPS mode.

OpenSSL no longer creates X.509 v1 certificates

With the OpenSSL TLS toolkit 3.2.1 introduced in RHEL 9.5, you can no longer create certificates in the X.509 version 1 format using the openssl CA tool. The X.509 v1 format does not meet current web requirements.

Libreswan now requests ESN by default

In Libreswan, the default value for the configuration option `esn=` has changed from no to either. This means that when initiating connections, Libreswan requests the use of Extended Serial Number (ESN) by default. In particular, when hardware offload is used, this new behavior prevents certain network interface cards (NIC) from establishing IPsec connection if they do not support ESN. To disable ESN, set esn= to no and the replay_window= option to a value of 32 or lower. For example:

```
esn=no
replay_window=32
```

The replay_window= option is necessary because a different mechanism uses ESN for anti-replay protection with window sizes larger than 32.

Chapter 8: Shells and command-line tools

Data Encryption Standard (DES) algorithm is not available for net-snmp communication in Red Hat Enterprise Linux 9

In previous versions of RHEL, DES was used as an encryption algorithm for secure communication between net-snmp clients and servers. In RHEL 9, the DES algorithm isn't supported by the OpenSSL library. The algorithm is marked as *insecure* and the DES support for net-snmp has therefore been removed.

The ABRT tool has been removed

The Automatic Bug Reporting Tool (ABRT) for detecting and reporting application crashes is not available in RHEL 9.

As a replacement, use the `systemd-coredump` tool to log and store core dumps, which are automatically generated files after a program crashes.

The `hidepid=n` mount option is not supported in RHEL 9 systemd

The mount option `hidepid=n`, which controls who can access information in `/proc/[pid]` directories, is not compatible with systemd infrastructure provided in RHEL 9.

In addition, using this option might cause certain services started by `systemd` to produce SELinux AVC denial messages and prevent other operations from being completed.

The dump utility from the dump package has been removed

The dump utility used for backup of file systems has been deprecated in Red Hat Enterprise Linux 8 and is not available in RHEL 9.

In RHEL 9, Red Hat recommends using the *`tar`, or `dd` as a backup tool for ext2, ext3, and ext4 file systems*. The `dump` utility will be a part of the EPEL 9 repository.

Note that the restore utility from the `dump` package remains available and supported in RHEL 9 and is available as the restore package.

RHEL 9 does not contain ReaR crontab

The `/etc/cron.d/rear` crontab in the rear package, which runs `rear mkrescue` after the disk layout changes, has been removed in RHEL 9.

If you relied on the `/etc/cron.d/rear` crontab to run `rear mkrescue`, you can manually configure periodic runs of ReaR instead.

Note

The rear package in RHEL contains the following examples for scheduling jobs:

- the `/usr/share/doc/rear/rear.cron` example crontab
- the `/usr/share/doc/rear/rear.{service,timer}` example systemd unit

Do not use these examples without site-specific modifications or other actions to take updated backups for system recovery. You must take regular backups in addition to re-creating the rescue image. The steps to take a backup depend on the local configuration. If you run the rear mkrescue command without taking an updated backup at the same time, the system recovery process would use a previous backup that might be inconsistent with the saved layout.

Support for the raw command-line tool has been removed

With this release, the raw (`/usr/bin/raw`) command-line tool has been removed from the `util-linux` package, because Linux kernel does not support raw devices since version 5.14.

Currently, there is no replacement available.

cgroupsv1 is deprecated in RHEL 9

cgroups is a kernel subsystem used for process tracking, system resource allocation and partitioning. Systemd service manager supports booting in the cgroups v1 mode and in cgroups v2 mode. In Red Hat Enterprise Linux 9, the default mode is v2. In the next major release, systemd will not support booting in the cgroups v1 mode and only cgroups v2 mode will be available.

The lsb-release binary is not available in RHEL 9

The information in the `/etc/os-release` file was previously available by calling the lsb-release binary. This binary was included in the redhat-lsb package, which was removed in RHEL 9. Now, you can display information about the operating system, such as the distribution, version, code name, and associated metadata, by reading the `/etc/os-release` file. This file is provided by Red Hat and any changes to it are overwritten with each update of the `redhat-release` package. The format of the file is `KEY=VALUE`, and you can safely source the data for a shell script.

Software management

Package management with DNF/YUM

In Red Hat Enterprise Linux 9, software installation is ensured by DNF. Red Hat continues to support the usage of the yum term for consistency with previous major versions of RHEL. If you type dnf instead of yum, the command works as expected because both are aliases for compatibility.

Although RHEL 8 and RHEL 9 are based on DNF, they are compatible with YUM used in RHEL 7.

For more information, see Managing software with the DNF tool.

Notable RPM features and changes

Red Hat Enterprise Linux 9 is distributed with RPM version **4.16**. This version introduces many enhancements over its previous versions.

New SPEC features:

- Fast macro-based dependency generators

  It is now possible to define dependency generators as regular RPM macros. This is especially useful in combination with the embedded Lua interpreter (%{lua:…}) because it enables writing sophisticated yet fast generators and avoiding redundant forking and executing a shell script.

  Example:

```
%__foo_provides()   %{basename:%{1}}
```

- The `%generate_buildrequires` section that enables generating dynamic build dependencies

  Additional build dependencies can now be generated programmatically at RPM build time, using the newly available `%generate_buildrequires` section. This is useful when packaging software written in a language in which a specialized utility is commonly used to determine run-time or build-time dependencies, such as Rust, Golang, Node.js, Ruby, Python or Haskell.

- Meta (unordered) dependencies

  A new dependency qualifier called meta enables expressing dependencies that are not specifically install-time or run-time dependencies. This is useful for avoiding unnecessary dependency loops that could otherwise arise from the normal dependency ordering, such as when specifying the dependencies of a meta package.

  Example:

```
Requires(meta): <pkgname>
```

- Native version comparison in expressions

  It is now possible to compare arbitrary version strings in expressions by using the newly supported v"…" format.

  Example:

```
%if v"%{python_version}" < v"3.9"
```

- Caret version operator, opposite of tilde

  The new caret (^) operator can be used to express a version that is higher than the base version. It is a complement to the existing tilde (~) operator which has the opposite semantics.
  %elif, %elifos and %elifarch statements

- Optional automatic patch and source numbering

  Patch: and Source: tags without a number are now automatically numbered based on the order in which they are listed.

- %autopatch now accepts patch ranges

  The %autopatch macro now accepts the -m and -M parameters to limit the minimum and maximum patch number to apply, respectively.

- %patchlist and %sourcelist sections

  It is now possible to list patch and source files without preceding each item with the respective Patch: and Source: tags by using the newly added %patchlist and %sourcelist sections.

- A more intuitive way to declare build conditionals

  Starting from RHEL 9.2, you can use the new %bcond macro to build conditionals. The %bcond macro takes a build conditional name and the default value as arguments. Compared to the old %bcond_with and %bcond_without macros, %bcond is easier to understand and allows you to calculate the default value at build time. The default value can be any numeric expression.

  Example:

  To create a gnutls build conditional, enabled by default:

```
%bcond gnutls 1
```

  To create a bootstrap build conditional, disabled by default:

```
%bcond bootstrap 0
```

  To create an openssl build conditional, defaulting to opposite of gnutls:

```
%bcond openssl %{without gnutls}
```

- The RPM database is now based on the sqlite library. Read-only support for BerkeleyDB databases has been retained for migration and query purposes.

- A new rpm-plugin-audit plug-in for issuing audit log events on transactions, previously built into RPM itself

- Increased parallelism in package builds

  There have been numerous improvements to the way the package build process is parallelized. These improvements involve various buildroot policy scripts and sanity checks, file classification, and subpackage creation and ordering. As a result, package builds on multiprocessor systems, particularly for large packages, should now be faster and more efficient.

- Enforced UTF-8 validation of header data at build-time

- RPM now supports the Zstandard (zstd) compression algorithm

In RHEL 9, the default RPM compression algorithm has switched to Zstandard (zstd). As a result, packages now install faster, which can be especially noticeable during large transactions.

Podman v5.0 deprecations

In RHEL 9.5, the following is deprecated in Podman v5.0:

- The system connections and farm information stored in the containers.conf file are now read-only. The system connections and farm information will now be stored in the podman.connections.json file, managed only by Podman. Podman continues to support the old configuration options such as [engine.service_destinations] and the [farms] section. You can still add connections or farms manually if needed; however, it is not possible to delete a connection from the containers.conf file with the podman system connection rm command.

- The slirp4netns network mode is deprecated and will be removed in a future major release of RHEL. The pasta network mode is the default network mode for rootless containers.

- The cgroups v1 for rootless containers is deprecated and will be removed in a future major release of RHEL.

The runc container runtime has been deprecated

The runc container runtime is deprecated and will be removed in a future major release of RHEL. The default container runtime is crun.

RED HAT CUSTOMER PORTAL LABS

Red Hat Customer Portal Labs is a set of tools in a section of the Customer Portal available at https://access.redhat.com/labs/. The applications in Red Hat Customer Portal Labs can help you improve performance, quickly troubleshoot issues, identify security problems, and quickly deploy and configure complex applications. Some of the most popular applications are:

- Registration Assistant
- Kickstart Generator
- Red Hat Product Certificates
- Red Hat CVE Checker
- Kernel Oops Analyzer
- Red Hat Code Browser
- VNC Configurator
- Red Hat OpenShift Container Platform Update Graph
- Red Hat Satellite Upgrade Helper
- JVM Options Configuration Tool
- Load Balancer Configuration Tool
- Red Hat OpenShift Data Foundation Supportability and Interoperability Checker
- Ansible Automation Platform Upgrade Assistant
- Ceph Placement Groups (PGs) per Pool Calculator
- Yum Repository Configuration Helper
- Red Hat Out of Memory Analyzer