aws

User Guide

Amazon Linux 2027

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon Linux 2027: User Guide

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

What is Amazon Linux 2027? 1
Release cadence 2
Major and minor releases 2
Consuming new releases 2
Long-term support policy 3
Naming and versioning 4
Performance and operational optimizations 5
Relationship to Fedora 7
Customized cloud-init 7
SELinux 9
Security updates and features 9
SELinux enforcing by default 10
Post-Quantum Cryptography (PQC) by Default 10
SSH server default 10
Manage updates 11
Security in the cloud 11
Networking service 11
Core toolchain packages glibc, gcc, binutils 13
Package management tool 13
Compatibility with AL2023 commands 14
Migrating to DNF5 15
RPM 6.0 20
Default SSH server configuration 20
Deprecated functionality 23
`compat-` library packages 23
Deprecated in AL2023 23
DNF version 4 and its Python API 24
32bit x86 (i686) runtime support 25
`aspell` 25
Berkeley DB (`libdb`) 25
`cron` 25
IMDSv1 26
`pcre` version 1 26
System V init (`sysvinit`) 26

Control groups version 1 (cgroup v1) ... 27
ksh ... 27
opensmtpd ... 27
ftp client ... 27
EOL Packages are deprecated ... 27
Deprecated in AL2027 ... 28
EOL packages are deprecated ... 28
Comparing AL2023 and AL2027 ... 29
Toolchain considerations ... 30
AMI package changes ... 30
AL2023 and AL2027 AMI disk usage ... 31
AL2023 and AL2027 AMI comparison ... 31
AL2023 and AL2027 Minimal AMI comparison ... 59
System Requirements ... 78
CPU requirements for running AL2027 ... 78
ARM CPU requirements for AL2027 ... 78
x86-64 CPU requirements for AL2027 ... 79
Memory (RAM) requirements for running AL2027 ... 80
Amazon Linux kernel ... 81
Using Multi-Gen LRU (MGLRU) on AL2027 kernels ... 81
Configuration and Tuning ... 82
Graphical desktop ... 84
Install GNOME ... 84
Prerequisites ... 85
Step 1: Install the GNOME desktop environment ... 85
Step 2: Start the desktop ... 86
Step 3: Clean up resources ... 86
Related topics ... 87
Configure RDP on AL2027 ... 87
Prerequisites ... 88
Step 1: Install the desktop environment ... 88
Step 2: Create a TLS certificate ... 88
Step 3: Configure and enable the RDP service ... 89
Step 4: Connect using an RDP client ... 91
(Optional) Start the service at boot ... 92
(Optional) Disable RDP access ... 92

(Optional) Use your own TLS certificate ... 92
Related topics ... 93
Install LibreOffice on AL2027 ... 93
Prerequisites ... 94
Step 1: Determine your system architecture ... 94
Step 2: Download the LibreOffice RPM package ... 95
Step 3: Extract the archive ... 95
Step 4: Install the RPM packages ... 95
Step 5: Verify the installation ... 96
Launching LibreOffice ... 96
Clean up ... 97
Troubleshooting ... 97
Using AL2027 on AWS ... 99
Getting started with AWS ... 99
Sign up for an AWS account ... 99
Granting programmatic access ... 99
AL2027 on Amazon EC2 ... 101
Launching AL2027 using the Amazon EC2 console ... 102
Launching AL2027 using the SSM parameter and AWS CLI ... 102
Launching AL2027 using a specific AMI ID ... 104
Connect to your instance ... 104
Connecting to AL2027 instances ... 104
Standard and minimal AMIs ... 105
AL2027 in containers ... 125
AL2027 base container image ... 126
AL2027 minimal container image ... 127
Bare-bones AL2027 container images ... 128
AL2027 container image package list comparison ... 132
AL2027 Minimal AMI compared to container images ... 136
Amazon EFS on AL2027 ... 152
`amazon-efs-utils` ... 152
Mounting Amazon EFS file system ... 153
Identifying Amazon Linux versions ... 154
`/etc/os-release` ... 154
Key differences ... 155
Field types ... 155

/etc/os-release examples ... 157
Comparison with other distributions ... 159
Amazon Linux-specific identification ... 161
/etc/system-release ... 161
/etc/image-id ... 162
Amazon Linux-specific examples ... 162
Example code ... 165
Filesystem layout ... 179
/ ... 179
/boot ... 180
/boot/efi ... 180
/etc ... 180
/home ... 180
/root ... 181
/srv ... 181
/tmp ... 181
/run ... 182
/usr ... 182
Unified /usr/bin and /usr/sbin ... 183
/usr/bin ... 183
/usr/include ... 183
/usr/lib and /usr/lib64 ... 183
/usr/local ... 183
/usr/share ... 183
/var ... 184
/var/cache ... 184
/var/lib ... 184
/var/log ... 184
/var/spool ... 184
/var/tmp ... 184
Updating AL2027 ... 185
Best practices for safely deploying updates ... 186
Preparing for minor updates ... 188
Preparing for major updates ... 188
Receive notifications on new updates ... 188
Deterministic upgrades through versioned repositories ... 189

Differences between minor and major version upgrades ... 191
Knowing when updates are available ... 191
Control the package updates available from the AL2027 repositories ... 191
Instance replacement ... 192
In-place deterministic upgrades ... 192
Managing updates ... 198
Checking for available package updates ... 199
Applying updates using DNF and repository versions ... 200
Checking for newer repository versions with `dnf check-release-update` ... 200
Getting package support information ... 201
Launching an instance with a specific repository version enabled ... 202
Adding, enabling, or disabling new repositories ... 203
Adding repositories with cloud-init ... 204
Automatic service restart after updates ... 204
When is a reboot required to apply updates? ... 205
Kernel updates ... 205
Support info plugin ... 206
Using `dnf supportinfo` ... 206
Prerequisites ... 207
Getting started ... 207
Query packages ... 208
Filter packages ... 208
Structured output ... 210
Manage the cache ... 211
Non-root usage ... 211
Configuration reference ... 211
Output fields reference ... 212
Troubleshooting ... 212
Additional resources ... 213
SupportInfo XML ... 213
How the plugin uses this file ... 214
Example ... 214
Element reference ... 215
Referential integrity ... 216
Programming languages and runtimes ... 217
Databases ... 218

Python ... 219
Migrating to Python 3.14 ... 219
Python modules ... 220
Package Installation Security ... 220
Java ... 220
Node.js ... 221
Security best practices ... 222
TypeScript ... 222
Go ... 225
Rust ... 226
Rust versions ... 226
.NET ... 227
Migrating to .NET 10 ... 227
Installing .NET ... 227
PHP ... 227
Migrating to new PHP versions ... 228
PHP modules ... 228
Ruby ... 228
Swift ... 230
AL2027 Function: Swift ... 231
C/C++ and Fortran ... 231
LLVM/Clang ... 232
Language standards ... 233
Perl ... 234
Perl modules ... 235
AL2027 reserved users and groups ... 236
List of AL2027 reserved users ... 236
List of AL2027 reserved groups ... 245
Running applications ... 259
Resource control with systemd ... 259
Resource control with `systemd-run` for running one-off commands ... 260
Resource control in a `systemd` service ... 262
Using cgroups utilities ... 267
Codecs available in AL2027 ... 269
Security and Compliance ... 273
Security advisories ... 274

ALAS Announcements ... 275
ALAS FAQs ... 275
ALAS Advisories ... 275
Advisories and RPM repositories ... 275
Advisory IDs ... 276
Advisory Released Date and Advisory Updated Date ... 276
Advisory Types ... 277
Advisory Severities ... 277
Advisories and Packages ... 278
Advisories and CVEs ... 278
Advisory Text ... 279
Kernel Live Patch Advisories ... 279
`updateinfo.xml` schema ... 280
Listing applicable advisories ... 281
In-place updates ... 283
Setting SELinux modes for AL2027 ... 285
Default SELinux status and modes for AL2027 ... 286
Change to `permissive` mode ... 287
Option to disable SELinux ... 288
Post-Quantum Cryptography (PQC) on AL2027 ... 289
How to disable Post-Quantum Cryptography (PQC) on AL2027 ... 290
Enable FIPS Mode on AL2027 ... 291
Enable FIPS Mode in an AL2027 Container ... 293
Swap OpenSSL FIPS providers on AL2027 ... 295
Kernel hardening ... 297
Repository metadata signing ... 306
How repository metadata signing works ... 307
Difference between `gpgcheck` and `repo_gpgcheck` ... 307
Enabling repository metadata verification ... 308
Verifying that repository metadata signing is working ... 308
Use repository metadata verification in automation ... 309
Commands that refresh repository metadata ... 310
Detect a skipped repository ... 312
Pinned versions ... 312
GPG public keys for AL2027 repositories ... 313
Security patching during preview ... 313

FIPS validation ... 313
UEFI Secure Boot on AL2027 ... 314
Enable UEFI Secure Boot on AL2027 ... 314
Enrollment of an existing instance ... 315
Register image from snapshot ... 315
Revocation updates ... 316
How UEFI Secure Boot works on AL2027 ... 316
Enrolling your own keys ... 317
Known issues and preview limitations ... 318
NVIDIA drivers ... 318
On-premises VM images ... 318
SSM Patch Manager ... 318
Attestable image examples ... 319
Reporting issues and feedback ... 319
Document history ... 320

What is Amazon Linux 2027?

AL2027 Preview

AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads. We encourage preview users to migrate to the latest version of the AMI and to keep their systems updated. By default, AL2027 instances do not automatically receive any updates, including critical and important security updates.

Amazon Linux 2027 (AL2027) is the next generation of Amazon Linux from Amazon Web Services (AWS). With AL2027, you can develop and run cloud and enterprise applications in a secure, stable, and high-performance runtime environment. AL2027 offers long-term support with access to the latest innovations in Linux. AL2027 is provided at no additional charge.

AL2027 is the successor to Amazon Linux 2023 (AL2023). For information about the differences between AL2023 and AL2027, see Comparing AL2023 and AL2027 and Package changes in Amazon Linux 2027.

Topics

- Release cadence
- Naming and versioning
- Performance and operational optimizations
- Relationship to Fedora
- Customized cloud-init
- SELinux
- Security updates and features
- Networking service
- Core toolchain packages glibc, gcc, binutils
- Package management tool
- Default SSH server configuration

Release cadence

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

AL2027 follows the same release cadence model as AL2023. AL2027 will be supported until 2032. There are two phases of support:

- **Standard support** – During this phase, the release receives quarterly minor version updates with new features, security updates, and bug fixes.
- **Maintenance** – During this phase, the release receives only security updates and critical bug fixes. These updates are published as soon as they are available.

Major and minor releases

With every Amazon Linux release (major version, minor version, or a security release), we release a new AMI.

- **Major version release** – Includes new features and improvements in security and performance across the stack. The improvements might include major changes to the kernel, toolchain, glibc, OpenSSL, and other system libraries and utilities. Major releases of Amazon Linux are based in part on the current version of the upstream Fedora Linux distribution. AWS might add or replace specific packages from other non-Fedora upstreams. For more information, see Relationship to Fedora.
- **Minor version release** – A quarterly update that includes security updates, bug fixes, and new features and packages. Each minor version is a cumulative list of updates that includes security and bug fixes in addition to new features and packages. These releases might include the latest language runtimes and other popular software packages.

Consuming new releases

Updates are provided through a combination of new AMI releases and corresponding new repository versions. By default, a new AMI and the repository version that it points to are coupled:

each release is locked to its own repository version. You can point your running Amazon EC2 instances to newer repository versions over time to apply updates in place, or you can update by launching new instances from the latest AMIs. For more information, see Deterministic upgrades through versioned repositories on AL2027.

Long-term support policy

Amazon Linux provides updates for all of your packages and maintains compatibility within a major version for your applications that are built on Amazon Linux. Core packages such as the glibc library, OpenSSL, OpenSSH, and the DNF5 package manager receive support for the lifetime of the major AL2027 release. Packages that are not part of the core packages are supported based on their specific upstream sources. You can see the support status and dates of individual packages by running the following command.

```
$ sudo dnf supportinfo --pkg packagename
```

You can get information on all currently installed packages by running the following command.

```
$ sudo dnf supportinfo --show installed
```

> **Note**
>
> During the preview, the support timeline reported for each package extends only to the end of the preview period. Support timelines for the AL2027 releases will be published with the public releases.

For all plugin commands, output formats, and configuration, see Query package support with the AL2027 support info plugin.

The full list of core packages is finalized during the preview. If you want to see more packages included as core packages, tell us. We evaluate all feedback as we receive it. Provide feedback through your designated AWS representative or file an issue in the `amazon-linux-2027` repo on GitHub.

Naming and versioning

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

AL2027 provides a minor release every three months during standard support. Each release is identified by an increment from 0 to N. The value 0 refers to the original major release for that iteration. All releases will be called Amazon Linux 2027. When the next version of Amazon Linux is released, AL2027 will enter the maintenance phase and receive security updates and critical bug fixes.

For example, minor releases of AL2027 have the following format:

- `2027.0.20260903`
- `2027.1.20261203`
- `2027.2.20270303`

The corresponding AL2027 AMIs have the following format:

- `al2027-preview-ami-2027.0.20260903.0-kernel-7.1-x86_64`
- `al2027-preview-ami-2027.1.20261203.0-kernel-7.1-x86_64`
- `al2027-preview-ami-minimal-2027.0.20260903.0-kernel-7.1-arm64`

Within a specific minor version, regular AMI releases occur with a timestamp of the date of the AMI release.

- `al2027-preview-ami-2027.0.20260903.0-kernel-7.1-x86_64`
- `al2027-preview-ami-2027.0.20261012.0-kernel-7.1-x86_64`
- `al2027-preview-ami-2027.0.20261120.0-kernel-7.1-x86_64`

To identify an Amazon Linux instance, read the Common Platform Enumeration (CPE) string from `/etc/system-release-cpe`. Then split the string into its fields and read the platform and version values. On AL2027, the CPE string has the following form:

```
cpe:2.3:o:amazon:amazon_linux:2027:2027.0.20260903
```

AL2027 also provides two files for platform identification:

- `/etc/amazon-linux-release` points to the same file as `/etc/system-release`
- `/etc/amazon-linux-release-cpe` points to the same file as `/etc/system-release-cpe`

These two files indicate that an instance is Amazon Linux. There is no need to read a file or split the string into fields, unless you want to know the specific platform and version values. For all identification methods and examples, see Identifying Amazon Linux instances and versions.

Performance and operational optimizations

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

Amazon Linux 7.1 kernel

- AL2027 uses the latest drivers for Elastic Network Adapter (ENA) and Elastic Fabric Adapter (EFA) devices. AL2027 focuses on performance and functionality backports for hardware in Amazon EC2 infrastructure.
- All kernel build and runtime configurations include many of the same performance and operational optimizations of AL2023.

Base toolchain selection and default build flags

- AL2027 packages are built with compiler optimizations (`-02`) enabled by default.
- AL2027 packages are built requiring `x86-64-v3` for `x86-64` systems (`-march=x86-64-v3`), and Graviton 2 or higher for `aarch64` (`-march=armv8.2-a+crypto -mtune=neoverse-n1`). This enables the compiler to use AVX and AVX2 instructions for improved performance on `x86-64` systems.
- AL2027 packages are built with auto-vectorization enabled (`-ftree-vectorize`).

- AL2027 packages are built with Link Time Optimization (LTO) enabled.
- AL2027 uses GCC 16.1 as the default compiler, along with updated versions of Rust, Clang/LLVM 22, and Go.
- On `aarch64` (Graviton) instances, AL2027 enables transparent huge pages (THP) for `malloc` heap arenas by default, with glibc using 2 MB huge pages for heap arenas. For workloads with a large memory working set and frequent random memory accesses, this can improve performance. Workloads running in memory-constrained cgroups (such as containers with low memory limits) might experience increased memory consumption. To disable THP for `malloc`, set the environment variable `GLIBC_TUNABLES=glibc.malloc.hugetlb=0`.
- Some workloads, such as certain database engines, perform better with transparent huge pages disabled system-wide. To disable THP at runtime, run `echo never > /sys/kernel/mm/transparent_hugepage/enabled` as root. To persist the setting across reboots, add `transparent_hugepage=never` to the kernel command line.

Package selection and versions

- AL2027 uses DNF5 as its package manager. DNF5 is a C++ rewrite of DNF (version 4) that provides faster package operations, lower memory usage, and quicker repository metadata handling. For more information, see Package management tool.
- Select backports to major system components include several performance improvements for running on Amazon EC2 infrastructure, especially Graviton instances.
- AL2027 is integrated with several AWS services and features. This includes the AWS CLI and SSM Agent.
- AL2027 uses Amazon Corretto as the Java Development Kit (JDK).
- AL2027 provides database engines and programming language runtime updates to newer versions as they're released by upstream projects. Programming language runtimes with new versions are added when they're released.

Deployment in a cloud environment

- The base AL2027 AMI and container images are frequently updated to support patching instance replacement.
- Kernel updates are included in AL2027 AMI updates. This means that you don't need to use commands such as `dnf upgrade` and `reboot` to update your kernel.

- In addition to the standard AL2027 AMI, a minimal AMI and container image is also available. Choose the minimal AMI to run an environment with the minimal number of packages that's required to run your service.
- By default, AL2027 AMIs and containers are locked to a specific version of the package repositories. There's no auto-update when they're launched. This means that you're always in control of when you ingest any package update. You can always test in a beta/gamma environment before rolling out to production. If there's a problem, you can use the pre-validated rollback path. For more information, see Deterministic upgrades through versioned repositories on AL2027.

Relationship to Fedora

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

AL2027 maintains its own release and support lifecycles independent of Fedora. AL2027 provides updated versions of open-source software, a large variety of packages, and frequent releases. This preserves the familiar RPM-based operating system.

AL2027 is not directly comparable to any specific Fedora release. AL2027 is based in part on Fedora 44 and Fedora 45. Some of the components are the same as the components in Fedora and some are modified. Other components were developed independently. The Amazon Linux kernel is sourced from upstream kernel.org releases, chosen independently from Fedora.

AL2027 is not a Fedora derivative or remix. Packages might differ in version, configuration, or availability from the corresponding Fedora release.

Customized cloud-init

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

The cloud-init package is an open-source application that bootstraps Linux images in a cloud computing environment. AL2027 includes cloud-init 26.1. For more information, see the cloud-init documentation on the cloud-init website.

AL2027 contains a customized version of cloud-init. With cloud-init, you can specify what occurs to your instance at boot time.

When you launch an instance, you can use the user-data fields to pass actions to cloud-init. This means that you can use common AMIs for many use cases and configure them dynamically when you start an instance. AL2027 also uses cloud-init to configure the `ec2-user` account.

AL2027 uses the cloud-init actions in `/etc/cloud/cloud.cfg.d` and `/etc/cloud/cloud.cfg`. You can create your own cloud-init action files in the `/etc/cloud/cloud.cfg.d` directory. Cloud-init reads all the files in this directory in lexicographical order. Later files overwrite values in earlier files. When cloud-init launches an instance, the cloud-init package does the following configuration tasks:

- Sets the default locale
- Sets the hostname
- Parses and handles user-data
- Generates host private SSH keys
- Adds a user's public SSH keys to `.ssh/authorized_keys` for easy login and administration
- Prepares the repositories for package management
- Handles package actions that are defined in user-data
- Runs user scripts that are in user-data
- Mounts instance store volumes, if applicable
  - Instance store volumes that support TRIM aren't formatted when an instance launches. Before you can mount instance store volumes, you must partition and format them.

    For more information about TRIM support, see Instance store volume TRIM support in the *Amazon EC2 User Guide*.
  - When you launch your instances, you can use the `disk_setup` module to partition and format your instance store volumes.

    For more information about the module, see Disk Setup on the cloud-init website.

- You can use the `mounts` module to mount instance store volumes by device name. On AL2027 instances, instance store volumes are exposed as NVMe block devices. For example, the following cloud-config directive mounts the instance store volume at `/dev/nvme1n1` to `/mnt`:

  ```
  #cloud-config
  mounts:
   - [ /dev/nvme1n1, /mnt ]
  ```

  The device name and enumeration order for instance store volumes can vary by instance type. For more information about controlling mounts, see Mounts on the cloud-init website.

For more information about cloud-init user-data formats, see Configuration formats on the cloud-init website.

SELinux

In AL2027, SELinux is set to **enforcing** mode by default. This is a change from AL2023, which used permissive mode by default. For more information about the default status and modes, see Default SELinux status and modes for AL2027.

In enforcing mode, SELinux actively enforces the loaded security policy on the entire system. Applications that relied on permissive behavior might need policy adjustments when you migrate to AL2027. To relax enforcement while you make those adjustments, you can change to permissive mode. For more information, see Change to `permissive` mode.

To change the SELinux mode or disable SELinux completely, see Option to disable SELinux.

Security updates and features

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

AL2027 includes several security enhancements over AL2023

Topics

- SELinux enforcing by default
- Post-Quantum Cryptography (PQC) by Default
- SSH server default
- Manage updates
- Security in the cloud

SELinux enforcing by default

The SELinux mandatory access control policies define permissions for users, processes, programs, files, and devices.

By default, SELinux is enabled and set to `enforcing` mode in AL2027. This means that SELinux security policy is fully enforced, in contrast with permissive mode (default in AL2023) where permission denials are logged but not enforced.

For more information about SELinux modes and policy, see Setting SELinux modes for AL2027 and the SELinux Project.

Post-Quantum Cryptography (PQC) by Default

The system-wide cryptographic policies on AL2027 now enables post-quantum cryptography (PQC) by default.

- **OpenSSH** enables Post-Quantum Cryptography (PQC) key exchange by default.
- **OpenSSL 3.5** added support for the ML-KEM hybrid key exchange algorithm, and ML-DSA (ML-DSA-44, ML-DSA-65, and ML-DSA-87) and SLH-DSA signature algorithms.
- **GnuTLS 3.8.10** supports ML-KEM hybrid key exchange algorithms and ML-DSA-44, ML-DSA-65, and ML-DSA-87 signature algorithms for TLS communications.
- **NSS 3.124** supports the ML-KEM hybrid key exchange algorithm and ML-DSA-44, ML-DSA-65, and ML-DSA-87 signature algorithms for TLS communications.

For more information about Post-Quantum Cryptography on AWS, see: AWS Cloud Security > Post-Quantum Cryptography

SSH server default

AL2027 includes **OpenSSH 9.9p1**.

- Post-Quantum Cryptography (PQC) key exchange is enabled by default
- Additional default configurations have been added to `/etc/ssh/sshd_config.d/10-amazon-hardening.conf`
- DSA signature algorithm support is no longer available. If you have existing DSA host keys or client keys, you must migrate to a supported key type (RSA, ECDSA, or Ed25519) before upgrading to AL2027.
- RSA keys <2048 bits are not allowed by default

For more information, see Default SSH server configuration.

Manage updates

Apply security updates using DNF and repository versions. For more information, see Manage package and operating system updates in AL2027.

Security in the cloud

Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security of the cloud and security in the cloud. For more information, see Security and Compliance in AL2027.

Networking service

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

The open-source project `systemd-networkd` is widely available in modern Linux distributions. The project uses a declarative configuration language that's similar to the rest of the `systemd` framework. Its primary configuration file types are `.network` and `.link` files.

The `amazon-ec2-net-utils` package configures each interface in the `/run/systemd/network` directory. For each Elastic Network Interface (ENI) that's attached to an instance, the package creates a per-interface configuration named `70-`*`interface_name`*`.network` that enables both IPv4 and IPv6 networking through DHCP.

Interface-specific settings are written to a drop-in directory named `70-interface_name.network.d` that's associated with the per-interface configuration. The `policy-routes@interface_name.service` unit generates an `eni.conf` drop-in in this directory. The drop-in installs policy routing rules that help ensure that locally sourced traffic is routed to the network through the corresponding instance's network interface. These rules ensure that the right traffic is routed through the ENI from the associated addresses or prefixes. For more information about using ENI, see Elastic network interfaces in the *Amazon EC2 User Guide*.

You can customize this networking behavior by placing a custom configuration file in the `/etc/systemd/network` directory to override the default configuration settings contained in `/run/systemd/network`.

Important

If you supply a drop-in configuration file, ensure that `amazon-ec2-net-utils` has finished generating the base `70-interface_name.network` configuration and its associated `eni.conf` drop-in in `/run/systemd/network` and allow time for the default configuration to be fully applied before the drop-in takes effect. If the drop-in is applied at approximately the same time as the base configuration, the rapid succession of configuration changes can cause the interface link to go down. To avoid this race condition, make sure to allow enough time for the default configuration to take place before applying the custom configuration.

The systemd.network documentation on the freedesktop.org website describes how the `systemd-networkd` service determines the configuration that applies to a specific interface. It also generates alternative names, known as altnames, for the ENI-backed interfaces to reflect the properties of various AWS resources. These ENI-backed interface properties are the `ENI ID` and the `DeviceIndex` field of the ENI attachment. You can refer to these interfaces using their properties when using various tools, such as the `ip` command.

AL2027 instance interface names are generated using the `systemd` slot naming scheme. AL2027 ships with `systemd` version 260 and uses the `v260` net naming scheme by default. For more information, see systemd.net naming scheme on the freedesktop.org website.

Additionally, AL2027 uses the `fq_codel` active queue management network transmission scheduling algorithm by default. For more information, see CoDel overview on the Bufferbloat website.

Core toolchain packages glibc, gcc, binutils

AL2027 includes the following core toolchain versions:

- **GCC 16.1** (up from 11.5 in AL2023)
- **glibc 2.44** (up from 2.34 in AL2023)
- **binutils 2.46** (up from 2.41 in AL2023)

The GCC 11 to 16 upgrade is the largest toolchain change. Code compiled on AL2023 will run on AL2027, but recompilation may expose new warnings or errors due to stricter defaults.

LLVM/Clang 22 is built from a single unified SRPM. The separate `clang`, `lld`, and `lldb` source packages no longer exist.

Package management tool

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

The default software package management tool in AL2027 is DNF5. DNF5 is the successor to DNF (version 4), the package management tool in AL2023. DNF5 is faster, uses less memory, and is where upstream package management development now happens.

Both the `dnf` and `yum` commands run DNF5 in AL2027. Day-to-day commands are unchanged from AL2023:

```
sudo dnf install packagename
sudo dnf search packagename
sudo dnf remove packagename
sudo dnf upgrade
```

For a complete command reference, run `man dnf5` or see the DNF5 documentation on the dnf5.readthedocs.io website.

Topics

- Compatibility with AL2023 commands
- Migrating to DNF5
- RPM 6.0

Compatibility with AL2023 commands

AL2027 ships compatibility aliases so that common DNF (version 4) command spellings continue to work. This includes commands such as `dnf check-update`, `dnf updateinfo`, `dnf list installed`, `dnf groupinstall`, and `dnf erase`. Existing scripts that use these spellings run unchanged. The aliases are a standard part of AL2027, not a temporary migration aid.

Common DNF (version 4) option spellings also continue to work, such as `--sec-severity`, `--skip-broken`, and `--downloaddir`. A `--setopt` option that DNF5 does not recognize prints a warning instead of stopping the command. Scripts that pass settings specific to DNF (version 4) keep running.

AL2027 includes the command-line tools from the AL2023 `yum-utils` package. They run the equivalent DNF5 commands:

- `repoquery`
- `yumdownloader`
- `yum-builddep`
- `needs-restarting`
- `reposync`
- `repoclosure`
- `repomanage`
- `repotrack`
- `package-cleanup`
- `debuginfo-install`
- `yum-config-manager`
- `find-repos-of-install`

Installing the `yum-utils` or `dnf-utils` package name resolves to the `dnf5-plugins` package, which provides these tools.

Some differences remain. Command output formats differ. DNF5 renamed some options. For example, `--advisory` is now `--advisories`, with the old spelling kept as an alias. Options that took values in DNF (version 4) often became subcommands. For example, `dnf updateinfo --list` is now `dnf advisory list`. For the complete list, see Changes in DNF5 compared to DNF on the dnf5.readthedocs.io website.

Configuration carries over. DNF5 reads settings from `/etc/dnf/dnf.conf` and repository definitions from `/etc/yum.repos.d/`. The Amazon Linux defaults ship in a separate vendor file, so `/etc/dnf/dnf.conf` starts empty.

To define your own command aliases, add drop-in files under `/etc/dnf/dnf5-aliases.d/`.

For the changes that can break scripts and the steps to move, see Migrating to DNF5.

Migrating to DNF5

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

AL2027 replaces DNF (version 4) with DNF5. The `dnf` and `yum` commands run DNF5. AL2027 does not include DNF (version 4), and you cannot install it. AL2027 ships a compatibility layer for the common DNF (version 4) spellings. Most commands and scripts written for AL2023 run unchanged. This page lists what that layer covers, the changes that can break scripts and tools, and the steps to move.

Topics

- What does not change
- Command changes
- Automatic updates
- Changes that can break scripts and tools
- Python API
- Plugins
- Steps to move

What does not change

The following AL2023 package management behaviors are unchanged in AL2027:

- The configuration carries over. Repository definitions stay in `/etc/yum.repos.d/`, and settings you add to `/etc/dnf/dnf.conf` apply as before. The Amazon Linux defaults moved to a vendor file, so `/etc/dnf/dnf.conf` starts empty.
- The update model is the same: versioned repositories, a locked `releasever`, and `dnf upgrade --releasever` to move between release versions. For more information, see Deterministic upgrades through versioned repositories on AL2027.
- Installed-package queries with `rpm`, such as `rpm -q` and `rpm -ql`, work unchanged.
- The AL2023 package manager configuration defaults carry over, including package signature checking being on by default and unreachable repositories being skipped rather than failing the command.

Command changes

The following DNF (version 4) spellings keep working through the compatibility layer. Adopt the native DNF5 spelling when you update a script.

DNF (version 4) command	Native DNF5 command
`dnf erase`	`dnf remove`
`dnf localinstall`	`dnf install`
`dnf groupinstall` , `dnf groupremove` , `dnf grouplist`	`dnf group install`, `dnf group remove`, `dnf group list`
`dnf list installed`	`dnf list --installed`
`dnf whatprovides`	`dnf provides`
`dnf deplist`	`dnf repoquery --providers-of=requires`
`dnf updateinfo list`	`dnf advisory list`

DNF (version 4) command	Native DNF5 command
`dnf repolist enabled`	`dnf repolist --enabled`
`dnf history userinstalled`	`dnf repoquery --userinstalled`
`dnf update-minimal`	`dnf upgrade-minimal`

Group names keep working. AL2027 resolves group specs by display name in addition to the group ID, so `dnf group install "Development Tools"` and `dnf install "@Development Tools"` work. The match is case-sensitive. Spell the name exactly as `dnf group list` shows it.

AL2027 also includes the `yum-utils` command-line tools, such as `repoquery` and `yumdownloader`. They run the equivalent DNF5 commands. For the full compatibility surface, see Compatibility with AL2023 commands.

Automatic updates

The `dnf5-plugin-automatic` package replaces the `dnf-automatic` tool from AL2023. Installing the `dnf-automatic` package name resolves to it, and the AL2023 timer spelling still works:

```
sudo dnf install dnf-automatic
sudo systemctl enable --now dnf-automatic.timer
```

Both the `dnf-automatic.timer` and `dnf5-automatic.timer` units run `dnf5 automatic`. Configure it in `/etc/dnf/automatic.conf`. The packaged defaults are documented in `/usr/share/dnf5/dnf5-plugins/automatic.conf`. If you carry an AL2023 `automatic.conf`, review it against the packaged defaults. DNF5 added options and removed others.

Changes that can break scripts and tools

The following changes in AL2027 can break existing scripts and tools:

- The DNF (version 4) Python API (`python3-dnf`, imported as `import dnf`) is not available. This is the largest change. See Python API.
- DNF (version 4) plugins do not load. See Plugins.
- The package is named `dnf5`. `rpm -q dnf` reports that `dnf` is not installed. Query `dnf5` instead.

- DNF5 has changed most exit codes from DNF (version 4). For example, a command line that DNF5 cannot parse exits with code 2, whereas DNF (version 4) returned 1 for many of these errors. If your scripts or automation check for specific exit code values, they might get unexpected results. We recommend that you treat any non-zero exit code as failure instead of testing for a specific value.
- `dnf history` requires a subcommand and exits with an error without one. Use `dnf history list`.
- Group name matching is case-sensitive in `dnf group install`, `dnf group remove`, `dnf group upgrade`, and `@name` specs on `dnf install`. For example, `"Development Tools"` matches and `"development tools"` does not. DNF (version 4) matched both forms. Group IDs avoid this, for example, `dnf group install development`. The `dnf group list` output shows the ID for every group, and `dnf group list` and `dnf group info` still accept name variants in any case.
- Output text changed in places. Do not parse human-readable output. Many commands support the `--json` option, for example, `dnf advisory list --json`, `dnf repolist --json`, and `dnf list --installed --json`. When you use a custom `--queryformat`, end the format string with `\n`. DNF5 does not add the newline.
- DNF5 logs to `/var/log/dnf5.log`. It does not write to `/var/log/dnf.log` or `/var/log/dnf.librepo.log`. Update log collection tools that read them.
- AL2027 does not provide module streams, and the `dnf module` commands are not included.
- A small set of DNF (version 4) commands was removed without a replacement, such as `dnf shell` and `dnf alias`. They fail with a clear error message rather than changing behavior silently.
- `yum-config-manager --add-repo` accepts repository definition (`.repo`) files. To add a repository from a plain URL, use `dnf config-manager addrepo --id=myrepo --set=baseurl=URL` or point `--from-repofile` at a `.repo` file.

Python API

The DNF (version 4) Python API is not available in AL2027. Port scripts that use `import dnf` to the DNF5 Python bindings in the `python3-libdnf5` package. For more information, see the DNF5 API documentation on the dnf5.readthedocs.io website.

For many scripts, calling the command line and reading JSON output is the smaller change: `dnf advisory list --json`, `dnf repolist --json`, or `dnf repoquery --queryformat json`.

Configuration management tools that use the DNF (version 4) Python API are affected the same way. For Ansible, install `python3-libdnf5` on the managed node and use the `dnf5` module that comes with `ansible-core`. Both packages are in the AL2027 repositories.

Plugins

DNF (version 4) plugins do not load in DNF5. The plugin interface changed.

> **Important**
>
> A plugin that is not ported stops working silently. Commands run, exit with code 0, and print no error message while the plugin's features are missing. Check for the features you rely on.

DNF5 command plugins are configured under `/etc/dnf/dnf5-plugins` and library plugins under `/etc/dnf/libdnf5-plugins`. The `/etc/dnf/plugins` path remains as a symbolic link, so tools that write there keep working, but configuration written for DNF (version 4) plugins has no effect.

The Amazon Linux plugin commands from AL2023 are provided as native DNF5 plugins, including `dnf check-release-update` and `dnf supportinfo`.

If you maintain your own DNF (version 4) plugin, port it to the DNF5 plugin interface. The development headers ship in the `dnf5-devel` and `libdnf5-devel` packages. The upstream Tutorial: Writing Plugins on the dnf5.readthedocs.io website walks through both plugin types: DNF5 command plugins and libdnf5 library plugins.

Steps to move

To move your scripts and tools from AL2023 to AL2027, complete the following steps:

1. Launch the latest AL2027 preview AMI and run your scripts, tools, and services as they are. Most run unchanged. Compare the results with the same run on AL2023. For launch instructions, see AL2027 on Amazon EC2.
2. Add `-y` to commands that must not prompt.

3. Treat error exit codes as zero or non-zero, not as exact values.
4. Replace output text parsing with `--json` output.
5. End custom `--queryformat` strings with `\n`.
6. Prefer group IDs in `dnf group` commands. Group names also work, but the name form is case-sensitive and names can change. IDs stay stable.
7. Point log collection at `/var/log/dnf5.log`.
8. Port code that uses `import dnf` to `python3-libdnf5`, or call the command line and read JSON output.
9. Port your own DNF (version 4) plugins to the DNF5 plugin interface.
10. Confirm the result. Run `dnf --version` and check that it reports `dnf5`.
11. If a command behaves differently than expected, see Changes that can break scripts and tools.

RPM 6.0

AL2027 uses RPM 6.0, upgraded from RPM 4.16 in AL2023. Package signature verification uses the `rpm-sequoia` OpenPGP implementation. If you build and distribute your own RPM packages, test your spec files and package signing against RPM 6.0.

Default SSH server configuration

If you have SSH clients from several years ago, you might see an error when you connect to an instance. If the error tells you there's no matching host key type found, update your SSH host key to troubleshoot this issue.

AL2027 includes OpenSSH 9.9 (up from 8.7 in AL2023).

Hardened server defaults

AL2027 ships hardened OpenSSH server defaults in the `/etc/ssh/sshd_config.d/10-amazon-hardening.conf` file, which disables root login, X11 forwarding, Kerberos authentication, and Active Directory login by default.

Default disabling of `ssh-rsa` signatures

AL2027 includes a default configuration that disables the legacy `ssh-rsa` host key algorithm and generates a reduced set of host keys. Clients must support the `ssh-ed25519` or the `ecdsa-sha2-nistp256` host key algorithm.

AL2027 also requires RSA keys to be at least 2048 bits.

The default configuration accepts any of these key exchange algorithms:

- `mlkem768x25519-sha256`
- `mlkem768nistp256-sha256`
- `mlkem1024nistp384-sha384`
- `curve25519-sha256`
- `curve25519-sha256@libssh.org`
- `ecdh-sha2-nistp256`
- `ecdh-sha2-nistp384`
- `ecdh-sha2-nistp521`
- `diffie-hellman-group-exchange-sha256`
- `diffie-hellman-group14-sha256`
- `diffie-hellman-group16-sha512`
- `diffie-hellman-group18-sha512`

The first three algorithms in this list are the NIST-standardized Module-Lattice Key Encapsulation Mechanism (ML-KEM) hybrid methods. AL2027 adds these post-quantum key exchange methods and prefers them ahead of the classical algorithms.

By default, AL2027 generates `ed25519` and ECDSA host keys. Clients support either the `ssh-ed25519` or the `ecdsa-sha2-nistp256` host key algorithm. When you connect by SSH to an instance, you must use a client that supports a compatible algorithm, such as `ssh-ed25519` or `ecdsa-sha2-nistp256`. If you need to use other key types, override the list of generated keys with a `cloud-config` fragment in user-data.

In the following example, `cloud-config` generates a `rsa` host key with the `ecdsa` and `ed25519` keys.

```
#cloud-config
 ssh_genkeytypes:
 - ed25519
 - ecdsa
 - rsa
```

If you use an RSA key pair for public key authentication, your SSH client must support a `rsa-sha2-256` or `rsa-sha2-512` signature. If you're using an incompatible client and can't upgrade, re-enable `ssh-rsa` support on your instance. To re-enable `ssh-rsa` support, activate the `LEGACY` system crypto policy using the following commands.

```
$ sudo dnf install crypto-policies-scripts
$ sudo update-crypto-policies --set LEGACY
```

DSA keys are **disabled by default**. If you have existing DSA host keys or client keys, you must migrate to a supported key type (RSA, ECDSA, or Ed25519) before upgrading to AL2027.

For more information about managing host keys, see Module reference: SSH on the cloud-init website.

Deprecated functionality in AL2027

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

This chapter covers functionality that was present in AL2023 and is removed in AL2027, and functionality that remains in AL2027 but is deprecated. Understanding both helps you migrate to AL2027 and prepare for future versions of Amazon Linux.

Functionality deprecated in AL1 or AL2 and removed in earlier versions remains unavailable in AL2027. For that history, see Deprecated functionality in AL2023.

Topics

- compat- library packages
- Deprecated in AL2023
- Deprecated in AL2027

`compat-` library packages

Packages with the `compat-` prefix provide binary compatibility with binaries built against older library versions. Each major version of Amazon Linux does not carry forward the `compat-` library packages of prior releases, and any it introduces are deprecated from the start. Rebuild software against the current library versions rather than relying on `compat-` packages.

Deprecated in AL2023

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

This section describes functionality that was deprecated in AL2023. Most deprecated functionality is removed in AL2027 as of the Preview release. Some functionality will be present in AL2027 Preview but will be removed prior to GA. For information on the current AL2027 Preview release, see the AL2027 User Guide and AL2027 Release Notes.

> **Note**
>
> This section will be updated over time as the Linux ecosystem evolves and future major versions of Amazon Linux are closer to release.
> This section will be updated during the AL2027 Preview.

Topics

- DNF version 4 and its Python API
- 32bit x86 (i686) runtime support
- aspell
- Berkeley DB (libdb)
- cron
- IMDSv1
- pcre version 1
- System V init (sysvinit)
- Control groups version 1 (cgroup v1)
- ksh
- opensmtpd
- ftp client
- EOL Packages are deprecated

DNF version 4 and its Python API

AL2027 replaces DNF (version 4) with DNF5. The `dnf` and `yum` commands run DNF5, and compatibility aliases keep common DNF (version 4) command spellings working. The DNF (version 4) Python API (`python3-dnf`) is not available; port Python scripts to the `python3-libdnf5` bindings. For more information, see Package management tool.

32bit x86 (i686) runtime support

AL2023 retains the ability to run 32bit x86 (i686) binaries on x86_64, but does not ship any userspace packages to support doing so. AL2027 will have support for running 32bit x86 (i686) binaries disabled.

`aspell`

While AL2023 ships with the `aspell` package, it is deprecated and is removed in AL2027. While still present in the initial AL2027 Preview release, it will be removed before general availability. We recommend that you migrate to modern replacements such as `hunspell` or `enchant2`.

The deprecation of `aspell` in AL2023 follows the broader community shift, for example `aspell` deprecation in Fedora.

Berkeley DB (`libdb`)

AL2023 ships with version 5.3.28 of the Berkeley DB (`libdb`) library. This is the last version of Berkeley DB before the license changed to the GNU Affero GPLv3 (AGPL) license, from the less restrictive Sleepycat license.

There are few packages in AL2023 that remain reliant on Berkeley DB (`libdb`). The library is deprecated in AL2023 and is removed in AL2027.

> **Note**
>
> The `dnf` package manager in AL2023 retains read-only support for a Berkeley DB (BDB) format `rpm` database. This support is removed in AL2027, which uses the SQLite `rpm` database format.

The deprecation of `libdb` follows the broader community shift away from it, for example the `libdb` deprecation in Fedora.

`cron`

The `cronie` package was installed by default on the AL2 AMI, providing support for the traditional `crontab` way of scheduling periodic tasks. In AL2023, `cronie` is not included by default. Therefore, support for `crontab` is no longer provided by default.

You can optionally install the `cronie` package to use classic `cron` jobs. We recommend that you migrate to `systemd` timers due to the added functionality provided by `systemd`.

It is possible that a future version of Amazon Linux will no longer include support for classic `cron` jobs and complete the transition to `systemd` timers. We recommend that you migrate away from using `cron`.

IMDSv1

By default, AL2023 AMIs are configured to launch in IMDSv2-only mode, disabling the use of IMDSv1. There is still the option to use AL2023 with IMDSv1 enabled. A future version of Amazon Linux is likely to enforce IMDSv2-only.

For more information on IMDS configuration for AMIs, see Configure the AMI in the *Amazon EC2 User Guide*.

`pcre` version 1

The legacy `pcre` package is deprecated and is removed in AL2027. The `pcre2` package is the successor. Although the first versions of AL2023 shipped with a limited number of packages building against `pcre`, these packages will be migrated to `pcre2` within AL2023. The deprecated `pcre` library will remain available in AL2023.

> **Note**
>
> The deprecated version of `pcre` will not receive security updates for the full lifetime of AL2023. For more information about the `pcre` support lifecycle and the amount of time that the package will receive security updates, see the package support statements on the `pcre` package.

The deprecation of `pcre` in favor of `pcre2` follows the broader community shift in this direction, for example `pcre` deprecation in Fedora.

System V init (`sysvinit`)

Although AL2023 retains backwards compatibility with System V service (init) scripts, the upstream `systemd` project, as part of its v254 release, announced the deprecation of support for System V service scripts, and indicated that support will be removed in a future version of `systemd`. For more information, see systemd.

AL2023 retains backwards compatibility with System V service (init) scripts, but the support is deprecated. AL2027, which ships a newer `systemd`, removes support for System V service (init) scripts, so migrate to native `systemd` unit files.

Control groups version 1 (cgroup v1)

cgroup v1 was deprecated in AL2 and AL2023 ran cgroup v2 by default. The `systemd` project removed cgroup v1 support in v258, and AL2027 ships a `systemd` newer than v258, so AL2027 supports cgroup v2 only. Move any remaining tooling that mounts or reads the legacy cgroup v1 hierarchy to the cgroup v2 interfaces. For more information, see Limiting process resource usage in Amazon Linux using cgroups.

ksh

The `ksh` shell is deprecated in AL2023 and is removed in AL2027. Port shell scripts to `bash`, which is included by default.

opensmtpd

The `opensmtpd` mail server is deprecated in AL2023 and is removed in AL2027. Use `postfix` or `sendmail`, both available in the repositories.

ftp client

The legacy `ftp` package from AL2 was removed in AL2023 and remains unavailable in AL2027. The package was not actively maintained upstream, and the FTP protocol transmits data, including credentials, in plaintext. For file transfers, `lftp` supports FTP, FTPS, HTTP, HTTPS, and SFTP. For downloads, `curl` is installed by default. Where you control both endpoints, we recommend `sftp` from the `openssh-clients` package, because transfers and credentials are encrypted.

EOL Packages are deprecated

Each package available in AL2023 has an associated support statement which covers Amazon Linux specific information. These statements cover the core of the OS and its lifetime, as well as packages such as php and `python`, where AL2023 ships multiple versions and each are supported for the duration that the upstream Open Source project does.

In AL2023 you can get package support information using the `dnf` package manager. For more information, see Getting package support information.

Where a package is no longer supported before the end of the major version of Amazon Linux, it should be assumed that this package is deprecated and will not be present in the next major version of Amazon Linux.

For packages such as `php` and `python`, where each major Amazon Linux version has shipped multiple versions, each with a different support lifecycle, it is likely that they will continue to be present in new major versions of Amazon Linux, albeit with little or no overlap of major versions of the packages. It is recommended to keep the Amazon Linux package support timelines in mind when selecting dependencies.

Deprecated in AL2027

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

Functionality deprecated in AL2027 is likely to be removed in a future version of Amazon Linux. Moving off deprecated functionality now reduces the work of migrating to the next version. This section is updated over the lifetime of AL2027 as the Linux ecosystem evolves.

Topics

- EOL packages are deprecated

EOL packages are deprecated

Each package in AL2027 has an associated support timeline, which you can query with the `dnf supportinfo` command. For more information, see Getting package support information. Where a package's support ends before the end of the major version of Amazon Linux, assume that the package is deprecated and will not be present in the next major version of Amazon Linux.

> **Note**
>
> During the preview period, the support timelines reflect the preview terms. Support statements for GA will be published before the public release.

Comparing AL2023 and AL2027

AL2027 Preview

AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

The following table summarizes the major component changes between AL2023 and AL2027.

Component changes from AL2023 to AL2027

Component	AL2023	AL2027	Notes
SELinux	Permissive	Enforcing	SELinux is now set to Enforcing mode by default.
Kernel	6.1 (default); 6.12 and 6.18 available	7.1	Kernel packages use versioned names, such as `kernel7.1` . The current preview release ships kernel version 7.1. Amazon Linux adds newer kernel versions over time, as with AL2023.
GCC	11.5	16.1	Major toolchain upgrade from GCC 11 to 16.
glibc	2.34	2.44	
binutils	2.41	2.46	
LLVM/Clang	15, 18, 19 (separate SRPMs)	22 (unified SRPM)	clang, lld, lldb, and libomp are built from a single source package.
systemd	252	260	Support for System V service scripts has been removed.

Component	AL2023	AL2027	Notes
RPM	4.16	6.0	OpenPGP signature verification uses the `rpm-sequoia` implementation.
DNF	DNF 4.14	DNF5 5.4	DNF5 replaces DNF. The `dnf` and `yum` commands run DNF5. The DNF (version 4) Python API (`python3-dnf`) is not available. Use `python3-libdnf5` instead. See Package management tool.
Python	3.9	3.14	The default `python3` is Python 3.14.
zlib	1.2.11	zlib-ng 2.3	Drop-in replacement through the `zlib-ng-compat` package.
PCRE	PCRE 8.x + PCRE2	PCRE2 only	PCRE1 has been removed.

Toolchain considerations

The GCC 11 to 16 jump is the largest toolchain change. Code compiled on AL2023 generally runs on AL2027, but recompiling might expose new warnings or errors because of stricter defaults in GCC 16.

RPM 6.0 replaces 4.16. If you maintain custom spec files, test them against the new RPM version.

LLVM 22 is built from a single unified SRPM. The separate `clang`, `lld`, and `lldb` source packages no longer exist.

AMI package changes

The set of packages installed by default differs between the AL2023 and AL2027 AMIs. The following tables list the RPMs present on the AL2023 and AL2027 standard and Minimal AMIs, along with their versions. A blank cell means the package is not installed by default on that AMI.

For the complete list of packages added, removed, and upgraded in the AL2027 repositories compared to AL2023, see Package changes in Amazon Linux 2027.

AL2027 installs fewer packages by default than AL2023, which reduces the disk space used on the root volume, as shown in the following table.

Comparing Amazon Linux 2023 and Amazon Linux 2027 AMI disk usage

Root-filesystem disk usage of the AL2023 and AL2027 standard and Minimal AMIs, measured with `df -m` on the `aarch64` images. Lower usage leaves more free space on the root volume.

AMI type	AL2023	AL2027	Reduction in AL2027
Standard AMI	1715 MiB	1142 MiB	33.4%
Minimal AMI	1246 MiB	893 MiB	28.3%

Comparing packages installed on Amazon Linux 2023 and Amazon Linux 2027 AMIs

A comparison of the RPMs present on the AL2023 and AL2027 standard AMIs.

Package	AL2023 AMI	AL2027 AMI
`acl`	2.4.0	2.4.0
`acpid`	2.0.32	2.0.34
`alternatives`	1.15	1.33
`amazon-chrony-config`	4.3	4.8
`amazon-ec2-net-utils`	2.7.6	2.7.6
`amazon-linux-repo-s3`	2023.12.20260831	2027.0.20260903
`amazon-linux-sb-keys`	2023.1	2027.1
`amazon-rpm-config`	228	

Package	AL2023 AMI	AL2027 AMI
amazon-ssm-agent	3.3.4624.0	3.3.5226.0
at	3.1.23	
attr	2.5.1	2.5.2
audit	3.1.5	4.1.3
audit-libs	3.1.5	4.1.3
audit-rules		4.1.3
authselect		1.7.1
authselect-libs		1.7.1
aws-cfn-bootstrap	2.0	2.0
aws-lc-libs		5.2.0
awscli-2	2.33.15	2.36.2
basesystem	11	
bash	5.2.15	5.3.0
bash-completion	2.11	2.17
bc	1.07.1	
bind-libs	9.18.50	
bind-license	9.18.50	
bind-utils	9.18.50	
binutils	2.41	
boost-filesystem	1.75.0	

Package	AL2023 AMI	AL2027 AMI
boost-system	1.75.0	
boost-thread	1.75.0	
bzip2	1.0.8	1.0.8
bzip2-libs	1.0.8	1.0.8
c-ares	1.19.1	1.34.6
ca-certificates	2025.2.76	2025.2.80_v9.0.305
checkpolicy	3.4	3.10
chkconfig	1.15	
chrony	4.3	4.8
cloud-init	22.2.2	26.1
cloud-init-cfg-ec2	22.2.2	26.1
cloud-utils-growpart	0.31	0.33
compact-dnf-utils		5.4.2.1
coreutils	8.32	9.10
coreutils-common	8.32	9.10
cpio	2.13	2.15
cracklib	2.9.6	2.10.3
cracklib-dicts	2.9.6	
crontabs	1.11	
crypto-policies	20260224	20260525

Package	AL2023 AMI	AL2027 AMI
crypto-policies-scripts	20260224	
cryptsetup	2.6.1	
cryptsetup-libs	2.6.1	2.8.4
curl		8.18.0
curl-minimal	8.17.0	
cyrus-sasl-gssapi		2.1.28
cyrus-sasl-lib	2.1.27	2.1.28
cyrus-sasl-plain	2.1.27	
dbus	1.12.28	1.16.0
dbus-broker	32	37
dbus-common	1.12.28	1.16.0
dbus-libs	1.12.28	1.16.0
device-mapper	1.02.185	1.02.212
device-mapper-libs	1.02.185	1.02.212
diffutils	3.8	3.12
dnf	4.14.0	
dnf-data	4.14.0	
dnf-plugin-release-notification	1.4	2.0.0

Package	AL2023 AMI	AL2027 AMI
dnf-plugin-support-info	2.0.0	3.0.0
dnf-plugins-core	4.3.0	
dnf-utils	4.3.0	
dnf5		5.4.2.1
dnf5-plugins		5.4.2.1
dosfstools	4.2	
dracut	102	108
dracut-config-ec2	3.1	3.1
dracut-config-generic	102	108
dwz	0.16	
dyninst	10.2.1	
e2fsprogs	1.46.5	1.47.3
e2fsprogs-libs	1.46.5	1.47.3
ec2-hibinit-agent	1.0.12	
ec2-instance-connect	1.1	1.1
ec2-instance-connect-selinux	1.1	1.1
ec2-utils	2.3.0	2.2.0
ed	1.14.2	

Package	AL2023 AMI	AL2027 AMI
efi-filesystem	5	6
efi-srpm-macros	5	
efivar	38	39
efivar-libs	38	39
elfutils-debuginfod-client	0.188	
elfutils-default-yama-scope	0.188	0.195
elfutils-libelf	0.188	0.195
elfutils-libs	0.188	0.195
ethtool	5.15	
expat	2.6.3	2.8.1
file	5.39	5.46
file-libs	5.39	5.46
filesystem	3.14	3.18
findutils	4.8.0	4.10.0
fmt		11.2.0
fonts-srpm-macros	2.0.5	
fstrm	0.6.1	
fuse-libs	2.9.9	
fuse3-libs		3.18.2

Package	AL2023 AMI	AL2027 AMI
gawk	5.1.0	5.3.2
gdbm		1.23
gdbm-libs	1.19	1.23
gdisk	1.0.8	1.0.10
gettext	0.21	1.0
gettext-envsubst		1.0
gettext-libs	0.21	1.0
gettext-runtime		1.0
ghc-srpm-macros	1.5.0	
glib2	2.82.2	2.88.1
glibc	2.34	2.44
glibc-all-langpacks	2.34	
glibc-common	2.34	2.44
glibc-gconv-extra	2.34	
glibc-locale-source	2.34	
glibc-minimal-lang pack		2.44
gmp	6.2.1	6.3.0
gnulib-l10n		20241231
gnupg2		2.4.9
gnupg2-dirmngr		2.4.9

Package	AL2023 AMI	AL2027 AMI
gnupg2-gpg-agent		2.4.9
gnupg2-gpgconf		2.4.9
gnupg2-keyboxd		2.4.9
gnupg2-minimal	2.3.7	
gnupg2-verify		2.4.9
gnutls	3.8.10	3.8.10
go-srpm-macros	3.8.0	
gpgme	1.23.2	
gpm-libs	1.20.7	
grep	3.8	3.12
groff-base	1.22.4	1.23.0
grub2-common	2.06	2.12
grub2-efi-aa64-ec2	2.06	2.12
grub2-pc-modules	2.06	
grub2-tools	2.06	2.12
grub2-tools-minimal	2.06	2.12
grubby	8.40	8.40
gssproxy	0.9.2	
gzip	1.12	1.14
hostname	3.23	3.25

Package	AL2023 AMI	AL2027 AMI
hunspell	1.7.0	
hunspell-en	0.20201207	
hunspell-en-GB	0.20201207	
hunspell-en-US	0.20201207	
hunspell-filesystem	1.7.0	
hwdata	0.384	0.409
ima-evm-utils-libs		1.6.2
info	6.7	
inih	58	62
initscripts	10.09	
iproute	6.10.0	6.17.0
iputils	20210202	20250605
irqbalance	1.9.0	1.9.5
jansson	2.14	2.14
jemalloc	5.2.1	
jitterentropy	3.4.1	
jq	1.8.1	1.8.1
json-c	0.14	0.18
kbd	2.4.0	2.10.0
kbd-legacy		2.10.0

Package	AL2023 AMI	AL2027 AMI
kbd-misc	2.4.0	2.10.0
kernel	6.1.182	
kernel-livepatch-repo-s3	2023.12.20260831	
kernel-srpm-macros	1.0	
kernel-tools	6.1.182	
kernel7.1		7.1.0
kernel7.1-tools		7.1.0
keyutils	1.6.3	
keyutils-libs	1.6.3	1.6.3
kmod	29	34.2
kmod-libs	29	34.2
kpatch-runtime	0.9.10	
krb5-libs	1.21.3	1.22.2
less	608	702
libacl	2.4.0	2.4.0
libaio	0.3.111	
libarchive	3.7.4	3.8.8
libargon2	20171227	
libassuan	2.5.5	2.5.7
libattr	2.5.1	2.5.2

Package	AL2023 AMI	AL2027 AMI
libbasicobjects	0.1.1	0.1.1
libblkid	2.37.4	2.41.5
libbpf	1.6.1	1.6.3
libcap	2.73	2.78
libcap-ng	0.8.2	0.9.3
libcbor	0.7.0	0.13.0
libcollection	0.7.0	0.7.0
libcom_err	1.46.5	1.47.3
libcomps	0.1.20	
libconfig	1.7.2	
libcurl-minimal	8.17.0	8.18.0
libdb	5.3.28	
libdhash	0.5.0	0.5.0
libdnf	0.69.0	
libdnf5		5.4.2.1
libdnf5-cli		5.4.2.1
libeconf	0.7.9	0.7.9
libedit	3.1	3.1
libev	4.33	
libevent	2.1.12	2.1.12

Package	AL2023 AMI	AL2027 AMI
libfdisk	2.37.4	2.41.5
libffi	3.4.4	3.5.2
libfido2	1.10.0	1.16.0
libgcc	14.2.1	16.1.1
libgcrypt	1.10.2	1.11.1
libgomp	14.2.1	16.1.1
libgpg-error	1.42	1.50
libibverbs	48.0	61.0
libicu		77.1
libidn2	2.3.2	2.3.8
libini_config	1.3.1	1.3.1
libkcapi	1.4.0	1.5.0
libkcapi-hasher		1.5.0
libkcapi-hmaccalc	1.4.0	1.5.0
libksba		1.6.7
liblastlog2		2.41.5
libldb	2.6.2	4.24.6
libmaxminddb	1.5.2	
libmetalink	0.1.3	
libmnl	1.0.4	1.0.5

Package	AL2023 AMI	AL2027 AMI
libmodulemd	2.13.0	
libmount	2.37.4	2.41.5
libnfsidmap	2.5.4	
libnghttp2	1.59.0	1.68.0
libnl3	3.5.0	3.12.0
libpath_utils	0.2.1	0.2.1
libpcap	1.10.1	1.10.6
libpipeline	1.5.3	1.5.8
libpkgconf	1.8.0	2.5.1
libpsl	0.21.5	0.21.5
libpwquality	1.4.4	1.4.5
libref_array	0.1.5	0.1.5
librepo	1.14.5	1.20.0
libreport-filesystem	2.15.2	
libseccomp	2.5.3	2.6.1
libselinux	3.4	3.10
libselinux-utils	3.4	3.10
libsemanage	3.4	3.10
libsepol	3.4	3.10
libsigsegv	2.13	

Package	AL2023 AMI	AL2027 AMI
libsmartcols	2.37.4	2.41.5
libsolv	0.7.22	0.7.39
libss	1.46.5	1.47.3
libsss_certmap	2.9.4	2.13.1
libsss_idmap	2.9.4	2.13.1
libsss_nss_idmap	2.9.4	2.13.1
libsss_sudo	2.9.4	2.13.1
libstdc++	14.2.1	16.1.1
libstoragemgmt	1.9.4	
libsupportinfo		2.0.0
libtalloc	2.3.4	2.4.4
libtasn1	4.19.0	4.20.0
libtdb	1.4.7	1.4.15
libtevent	0.13.0	0.17.1
libtextstyle	0.21	1.0
libtirpc	1.3.3	1.3.7
libtool-ltdl		2.5.4
libunistring	0.9.10	1.1
libusb1		1.0.30
libuser	0.63	

Package	AL2023 AMI	AL2027 AMI
libutempter	1.2.1	
libuuid	2.37.4	2.41.5
libuv	1.51.0	
libverto	0.3.2	0.3.2
libverto-libev	0.3.2	
libxcrypt	4.4.33	4.5.2
libxkbcommon		1.13.1
libxml2	2.10.4	2.12.10
libxslt	1.1.43	
libyaml	0.2.5	0.2.5
libzstd	1.5.5	1.5.7
lm_sensors-libs	3.6.0	
lmdb-libs	0.9.29	0.9.34
logrotate	3.20.1	3.22.0
lsof	4.94.0	4.98.0
lua-libs	5.4.4	5.4.8
lua-srpm-macros	1	
lz4-libs	1.9.4	1.10.0
man-db	2.9.3	2.13.1
man-pages	6.04	6.13

Package	AL2023 AMI	AL2027 AMI
mpdecimal		4.0.1
mpfr	4.1.0	4.2.2
nano	8.3	8.7.1
ncurses	6.6	
ncurses-base	6.6	6.6
ncurses-libs	6.6	6.6
net-tools	2.0	
nettle	3.10.1	3.10.1
newt	0.52.21	
nfs-utils	2.5.4	
nmap-ncat		7.93
npth	1.6	1.8
nspr	4.36.0	
nss	3.112.0	
nss-softokn	3.112.0	
nss-softokn-freebl	3.112.0	
nss-sysinit	3.112.0	
nss-util	3.112.0	
ntsysv	1.15	
numactl-libs	2.0.14	2.0.19

Package	AL2023 AMI	AL2027 AMI
ocaml-srpm-macros	6	
oniguruma	6.9.7.1	6.9.10
openblas-srpm-macros	2	
openldap	2.4.57	2.6.13
openssh	9.9p1	9.9p1
openssh-clients	9.9p1	9.9p1
openssh-server	9.9p1	9.9p1
openssl	3.5.7	3.5.7
openssl-fips-provider-latest	3.5.7	3.5.7
openssl-libs	3.5.7	3.5.7
openssl-pkcs11	0.4.12	
os-prober	1.77	1.81
p11-kit	0.24.1	0.26.2
p11-kit-trust	0.24.1	0.26.2
package-notes-srpm-macros	0.4	
pam	1.5.1	1.7.1
pam-libs		1.7.1
parted	3.4	
passwd	0.80	

Package	AL2023 AMI	AL2027 AMI
pciutils	3.7.0	3.15.0
pciutils-libs	3.7.0	3.15.0
pcre2	10.40	10.47
pcre2-syntax	10.40	10.47
perl-AutoLoader	5.74	
perl-B	1.80	
perl-base	2.27	
perl-Carp	1.50	
perl-Class-Struct	0.66	
perl-constant	1.33	
perl-Data-Dumper	2.191	
perl-Digest	1.20	
perl-Digest-MD5	2.59	
perl-DynaLoader	1.47	
perl-Encode	3.21	
perl-Errno	1.30	
perl-Exporter	5.79	
perl-Fcntl	1.13	
perl-File-Basename	2.85	
perl-File-Path	2.18	

Package	AL2023 AMI	AL2027 AMI
perl-File-stat	1.09	
perl-File-Temp	0.231.200	
perl-FileHandle	2.03	
perl-Getopt-Long	2.58	
perl-Getopt-Std	1.12	
perl-HTTP-Tiny	0.092	
perl-if	0.60.800	
perl-interpreter	5.32.1	
perl-IO	1.43	
perl-IO-Socket-IP	0.43	
perl-IO-Socket-SSL	2.075	
perl-IPC-Open3	1.21	
perl-libnet	3.13	
perl-libs	5.32.1	
perl-MIME-Base64	3.16	
perl-mro	1.23	
perl-Net-SSLeay	1.94	
perl-overload	1.31	
perl-overloading	0.02	
perl-parent	0.238	

Package	AL2023 AMI	AL2027 AMI
perl-PathTools	3.78	
perl-Pod-Escapes	1.07	
perl-Pod-Perldoc	3.28.01	
perl-Pod-Simple	3.42	
perl-Pod-Usage	2.01	
perl-podlators	4.14	
perl-POSIX	1.94	
perl-Scalar-List-Utils	1.56	
perl-SelectSaver	1.02	
perl-Socket	2.032	
perl-srpm-macros	1	
perl-Storable	3.37	
perl-subs	1.03	
perl-Symbol	1.08	
perl-Term-ANSIColor	5.01	
perl-Term-Cap	1.17	
perl-Text-ParseWords	3.30	
perl-Text-Tabs+Wrap	2021.0726	
perl-Time-HiRes	1.9764	
perl-Time-Local	1.300	

Package	AL2023 AMI	AL2027 AMI
perl-URI	5.18	
perl-vars	1.05	
pkgconf	1.8.0	2.5.1
pkgconf-m4	1.8.0	2.5.1
pkgconf-pkg-config	1.8.0	2.5.1
policycoreutils	3.4	3.10
policycoreutils-python-utils	3.4	3.10
popt	1.18	1.19
procps-ng	3.3.17	4.0.6
protobuf-c	1.5.0	
psacct	6.6.4	
psmisc	23.4	23.7
publicsuffix-list-dafsa	20260116	20260116
python-chevron	0.13.1	0.14.0
python-srpm-macros	3.9	
python3	3.9.25	3.14.7
python3-attrs	20.3.0	25.4.0
python3-audit	3.1.5	4.1.3
python3-awscrt	0.31.1	0.36.0

Package	AL2023 AMI	AL2027 AMI
python3-babel	2.9.1	
python3-cffi	1.14.5	
python3-chardet	4.0.0	
python3-charset-normalizer		3.4.4
python3-colorama	0.4.4	0.4.6
python3-configobj	5.0.6	5.0.9
python3-cryptography	36.0.1	
python3-daemon	2.3.0	3.1.0
python3-dateutil	2.8.1	2.9.0.post0
python3-dbus	1.2.18	
python3-distro	1.5.0	1.9.0
python3-dnf	4.14.0	
python3-dnf-plugins-core	4.3.0	
python3-docutils	0.16	0.21.2
python3-elementpath	2.3.2	
python3-gpg	1.23.2	
python3-hawkey	0.69.0	
python3-idna	2.10	3.11
python3-jinja2	2.11.3	3.1.6

Package	AL2023 AMI	AL2027 AMI
python3-jmespath	0.10.0	1.0.1
python3-jsonpatch	1.21	1.33
python3-jsonpointer	2.0	2.4
python3-jsonschema	3.2.0	4.23.0
python3-jsonschema-specifications		2024.10.1
python3-libcomps	0.1.20	
python3-libdnf	0.69.0	
python3-libs	3.9.25	3.14.7
python3-libselinux	3.4	3.10
python3-libsemanage	3.4	3.10
python3-libstorage mgmt	1.9.4	
python3-lockfile	0.12.2	0.12.2
python3-lxml	4.7.1	
python3-markupsafe	1.1.1	3.0.2
python3-netifaces	0.10.6	
python3-oauthlib	3.0.2	3.3.1
python3-pip-wheel	21.3.1	26.2.1
python3-ply	3.11	

Package	AL2023 AMI	AL2027 AMI
python3-policycoreutils	3.4	3.10
python3-prettytable	0.7.2	
python3-prompt-toolkit	3.0.24	3.0.41
python3-pycparser	2.20	
python3-pyrsistent	0.17.3	
python3-pyserial	3.4	
python3-pysocks	1.7.1	
python3-pytz	2022.7.1	
python3-pyyaml	5.4.1	6.0.3
python3-referencing		0.36.2
python3-requests	2.25.1	2.33.1
python3-rpds-py		0.27.0
python3-rpm	4.16.1.3	
python3-ruamel-yaml	0.16.6	0.19.1
python3-ruamel-yaml+oldlibyaml		0.19.1
python3-ruamel-yaml-clib	0.1.2	0.2.15
python3-setools	4.4.1	4.6.0
python3-setuptools	59.6.0	

Package	AL2023 AMI	AL2027 AMI
python3-setuptools-wheel	59.6.0	
python3-six	1.15.0	1.17.0
python3-supportinfo	1.0.0	
python3-systemd	235	
python3-urllib3	1.25.10	2.6.3
python3-wcwidth	0.2.5	0.6.0
python3-xmlschema	1.4.2	
quota	4.06	
quota-nls	4.06	
rdma-core-common		61.0
readline	8.1	8.3
rng-tools	6.17	
rootfiles	8.1	9.0
rpcbind	1.2.6	
rpm	4.16.1.3	6.0.0
rpm-build-libs	4.16.1.3	6.0.0
rpm-libs	4.16.1.3	6.0.0
rpm-plugin-selinux	4.16.1.3	6.0.0
rpm-plugin-systemd-inhibit	4.16.1.3	6.0.0

Package	AL2023 AMI	AL2027 AMI
rpm-sequoia		1.10.2.1
rpm-sign-libs	4.16.1.3	6.0.0
rsync	3.4.0	
rust-toolset-srpm-macros	1.97.0	
samba-common		4.24.6
samba-core-libs		4.24.6
samba-ndr-libs		4.24.6
sbsigntools	0.9.4	0.9.5
screen	4.8.0	
sdbus-cpp		2.2.1
sed	4.8	4.9
selinux-policy	38.1.76	44.5
selinux-policy-targeted	38.1.76	44.5
setup	2.13.7	2.15.1
shadow-utils	4.9	4.19.0
slang	2.3.2	
sqlite-libs	3.40.0	3.51.2
sssd-client	2.9.4	2.13.1
sssd-common	2.9.4	2.13.1

Package	AL2023 AMI	AL2027 AMI
sssd-kcm	2.9.4	2.13.1
sssd-krb5-common		2.13.1
sssd-nfs-idmap	2.9.4	
strace	6.12	
sudo	1.9.15	1.9.17
sysctl-defaults	1.0	1.0
sysstat	12.5.6	
system-release	2023.12.20260831	2027.0.20260903
systemd	252.23	260.1
systemd-libs	252.23	260.1
systemd-networkd	252.23	260.1
systemd-pam	252.23	260.1
systemd-resolved	252.23	260.1
systemd-shared		260.1
systemd-sysusers		260.1
systemd-udev	252.23	260.1
systemtap-runtime	5.4	
tar	1.34	1.35
tbb	2020.3	
tcpdump	4.99.1	

Package	AL2023 AMI	AL2027 AMI
tcsh	6.24.14	
time	1.9	
tpm2-tss		4.1.3
traceroute	2.1.3	
tzdata	2026c	2026c
unzip	6.0	6.0
update-motd	2.3	2.3
userspace-rcu	0.12.1	0.15.6
util-linux	2.37.4	2.41.5
util-linux-core	2.37.4	2.41.5
vim-common	9.2.920	
vim-data	9.2.920	9.2.920
vim-enhanced	9.2.920	
vim-filesystem	9.2.920	
vim-minimal	9.2.920	9.2.920
wget	1.21.3	
which	2.21	2.25
words	3.0	
xfsdump	3.1.11	
xfsprogs	6.12.0	7.1.1

Package	AL2023 AMI	AL2027 AMI
xkeyboard-config		2.47
xxd	9.2.920	
xxhash-libs	0.8.0	
xz	5.2.5	5.8.1
xz-libs	5.2.5	5.8.1
yum	4.14.0	
zip	3.0	3.0
zlib	1.2.11	
zlib-ng-compat		2.3.3
zram-generator	1.1.2	1.2.1
zram-generator-defaults	1.1.2	1.2.1
zstd	1.5.5	1.5.7

Comparing packages installed on Amazon Linux 2023 and Amazon Linux 2027 Minimal AMIs

A comparison of the RPMs present on the AL2023 and AL2027 Minimal AMIs.

Package	AL2023 Minimal	AL2027 Minimal
alternatives	1.15	1.33
amazon-chrony-config	4.3	4.8
amazon-ec2-net-utils	2.7.6	2.7.6

Package	AL2023 Minimal	AL2027 Minimal
amazon-linux-repo-s3	2023.12.20260831	2027.0.20260903
amazon-linux-sb-keys	2023.1	2027.1
audit	3.1.5	4.1.3
audit-libs	3.1.5	4.1.3
audit-rules		4.1.3
authselect		1.7.1
authselect-libs		1.7.1
aws-lc-libs		5.2.0
awscli-2	2.33.15	2.36.2
basesystem	11	
bash	5.2.15	5.3.0
bzip2-libs	1.0.8	1.0.8
ca-certificates	2025.2.76	2025.2.80_v9.0.305
checkpolicy	3.4	3.10
chrony	4.3	4.8
cloud-init	22.2.2	26.1
cloud-init-cfg-ec2	22.2.2	26.1
cloud-utils-growpart	0.31	0.33
coreutils	8.32	9.10
coreutils-common	8.32	9.10

Package	AL2023 Minimal	AL2027 Minimal
cpio	2.13	2.15
cracklib	2.9.6	2.10.3
cracklib-dicts	2.9.6	
crypto-policies	20260224	20260525
cryptsetup-libs	2.6.1	2.8.4
curl		8.18.0
curl-minimal	8.17.0	
cyrus-sasl-lib	2.1.27	2.1.28
dbus	1.12.28	1.16.0
dbus-broker	32	37
dbus-common	1.12.28	1.16.0
dbus-libs	1.12.28	1.16.0
device-mapper	1.02.185	1.02.212
device-mapper-libs	1.02.185	1.02.212
diffutils	3.8	3.12
dnf	4.14.0	
dnf-data	4.14.0	
dnf-plugin-release-notification	1.4	2.0.0
dnf-plugin-support-info	2.0.0	3.0.0

Package	AL2023 Minimal	AL2027 Minimal
dnf-plugins-core	4.3.0	
dnf5		5.4.2.1
dracut	102	108
dracut-config-ec2	3.1	3.1
dracut-config-generic	102	108
e2fsprogs	1.46.5	1.47.3
e2fsprogs-libs	1.46.5	1.47.3
ec2-instance-connect		1.1
ec2-instance-connect-selinux		1.1
ec2-utils	2.3.0	2.2.0
efi-filesystem	5	6
efivar	38	39
efivar-libs	38	39
elfutils-default-yama-scope	0.188	
elfutils-libelf	0.188	0.195
elfutils-libs	0.188	
expat	2.6.3	2.8.1
file	5.39	5.46

Package	AL2023 Minimal	AL2027 Minimal
file-libs	5.39	5.46
filesystem	3.14	3.18
findutils	4.8.0	4.10.0
fmt		11.2.0
fuse-libs	2.9.9	
fuse3-libs		3.18.2
gawk	5.1.0	5.3.2
gdbm		1.23
gdbm-libs	1.19	1.23
gdisk	1.0.8	1.0.10
gettext	0.21	1.0
gettext-envsubst		1.0
gettext-libs	0.21	1.0
gettext-runtime		1.0
glib2	2.82.2	2.88.1
glibc	2.34	2.44
glibc-all-langpacks	2.34	
glibc-common	2.34	2.44
glibc-locale-source	2.34	
glibc-minimal-langpack		2.44

Package	AL2023 Minimal	AL2027 Minimal
gmp	6.2.1	6.3.0
gnulib-l10n		20241231
gnupg2-minimal	2.3.7	
gnutls	3.8.10	3.8.10
gpgme	1.23.2	
grep	3.8	3.12
groff-base	1.22.4	
grub2-common	2.06	2.12
grub2-efi-aa64-ec2	2.06	2.12
grub2-pc-modules	2.06	
grub2-tools	2.06	2.12
grub2-tools-minimal	2.06	2.12
grubby	8.40	8.40
gzip	1.12	1.14
hostname	3.23	3.25
hwdata	0.384	0.409
inih	58	62
initscripts	10.09	
iproute	6.10.0	6.17.0
iputils	20210202	20250605

Package	AL2023 Minimal	AL2027 Minimal
irqbalance	1.9.0	1.9.5
jansson	2.14	
jitterentropy	3.4.1	
jq	1.8.1	1.8.1
json-c	0.14	0.18
kbd	2.4.0	2.10.0
kbd-legacy		2.10.0
kbd-misc	2.4.0	2.10.0
kernel	6.1.182	
kernel-livepatch-repo-s3	2023.12.20260831	
kernel7.1		7.1.0
keyutils-libs	1.6.3	1.6.3
kmod	29	34.2
kmod-libs	29	34.2
krb5-libs	1.21.3	1.22.2
less	608	702
libacl	2.4.0	2.4.0
libarchive	3.7.4	3.8.8
libargon2	20171227	
libassuan	2.5.5	

Package	AL2023 Minimal	AL2027 Minimal
libattr	2.5.1	2.5.2
libblkid	2.37.4	2.41.5
libbpf	1.6.1	1.6.3
libcap	2.73	2.78
libcap-ng	0.8.2	0.9.3
libcbor	0.7.0	0.13.0
libcom_err	1.46.5	1.47.3
libcomps	0.1.20	
libcurl-minimal	8.17.0	8.18.0
libdb	5.3.28	
libdnf	0.69.0	
libdnf5		5.4.2.1
libdnf5-cli		5.4.2.1
libeconf	0.7.9	0.7.9
libedit	3.1	3.1
libevent		2.1.12
libfdisk	2.37.4	2.41.5
libffi	3.4.4	3.5.2
libfido2	1.10.0	1.16.0
libgcc	14.2.1	16.1.1

Package	AL2023 Minimal	AL2027 Minimal
libgcrypt	1.10.2	
libgomp	14.2.1	16.1.1
libgpg-error	1.42	
libibverbs		61.0
libidn2	2.3.2	2.3.8
libkcapi	1.4.0	1.5.0
libkcapi-hasher		1.5.0
libkcapi-hmaccalc	1.4.0	1.5.0
liblastlog2		2.41.5
libmnl	1.0.4	1.0.5
libmodulemd	2.13.0	
libmount	2.37.4	2.41.5
libnghttp2	1.59.0	1.68.0
libnl3		3.12.0
libpcap		1.10.6
libpipeline	1.5.3	
libpkgconf		2.5.1
libpsl	0.21.5	0.21.5
libpwquality	1.4.4	1.4.5
librepo	1.14.5	1.20.0

Package	AL2023 Minimal	AL2027 Minimal
libreport-filesystem	2.15.2	
libseccomp	2.5.3	2.6.1
libselinux	3.4	3.10
libselinux-utils	3.4	3.10
libsemanage	3.4	3.10
libsepol	3.4	3.10
libsigsegv	2.13	
libsmartcols	2.37.4	2.41.5
libsolv	0.7.22	0.7.39
libss	1.46.5	1.47.3
libstdc++	14.2.1	16.1.1
libsupportinfo		2.0.0
libtasn1	4.19.0	4.20.0
libtextstyle	0.21	1.0
libtool-ltdl		2.5.4
libunistring	0.9.10	1.1
libuser	0.63	
libutempter	1.2.1	
libuuid	2.37.4	2.41.5
libverto	0.3.2	0.3.2

Package	AL2023 Minimal	AL2027 Minimal
libxcrypt	4.4.33	4.5.2
libxkbcommon		1.13.1
libxml2	2.10.4	2.12.10
libxslt	1.1.43	
libyaml	0.2.5	0.2.5
libzstd	1.5.5	1.5.7
logrotate	3.20.1	3.22.0
lua-libs	5.4.4	5.4.8
lz4-libs	1.9.4	1.10.0
man-db	2.9.3	
mpdecimal		4.0.1
mpfr	4.1.0	4.2.2
ncurses	6.6	6.6
ncurses-base	6.6	6.6
ncurses-libs	6.6	6.6
net-tools	2.0	
nettle	3.10.1	3.10.1
nmap-ncat		7.93
npth	1.6	
numactl-libs	2.0.14	2.0.19

Package	AL2023 Minimal	AL2027 Minimal
oniguruma	6.9.7.1	6.9.10
openldap	2.4.57	2.6.13
openssh	9.9p1	9.9p1
openssh-clients	9.9p1	9.9p1
openssh-server	9.9p1	9.9p1
openssl	3.5.7	3.5.7
openssl-fips-provider-latest	3.5.7	3.5.7
openssl-libs	3.5.7	3.5.7
openssl-pkcs11	0.4.12	
os-prober	1.77	1.81
p11-kit	0.24.1	0.26.2
p11-kit-trust	0.24.1	0.26.2
pam	1.5.1	1.7.1
pam-libs		1.7.1
passwd	0.80	
pciutils	3.7.0	3.15.0
pciutils-libs	3.7.0	3.15.0
pcre2	10.40	10.47
pcre2-syntax	10.40	10.47
pkgconf		2.5.1

Package	AL2023 Minimal	AL2027 Minimal
pkgconf-m4		2.5.1
pkgconf-pkg-config		2.5.1
policycoreutils	3.4	3.10
policycoreutils-python-utils		3.10
popt	1.18	1.19
procps-ng	3.3.17	4.0.6
psmisc	23.4	23.7
publicsuffix-list-dafsa	20260116	20260116
python3	3.9.25	3.14.7
python3-attrs	20.3.0	25.4.0
python3-audit	3.1.5	4.1.3
python3-awscrt	0.31.1	0.36.0
python3-babel	2.9.1	
python3-cffi	1.14.5	2.0.0
python3-chardet	4.0.0	
python3-charset-normalizer		3.4.4
python3-colorama	0.4.4	0.4.6
python3-configobj	5.0.6	5.0.9

Package	AL2023 Minimal	AL2027 Minimal
python3-cryptography	36.0.1	49.0.0
python3-dateutil	2.8.1	2.9.0.post0
python3-dbus	1.2.18	
python3-distro	1.5.0	1.9.0
python3-dnf	4.14.0	
python3-dnf-plugins-core	4.3.0	
python3-docutils	0.16	0.21.2
python3-elementpath	2.3.2	
python3-gpg	1.23.2	
python3-hawkey	0.69.0	
python3-idna	2.10	3.11
python3-jinja2	2.11.3	3.1.6
python3-jmespath	0.10.0	1.0.1
python3-jsonpatch	1.21	1.33
python3-jsonpointer	2.0	2.4
python3-jsonschema	3.2.0	4.23.0
python3-jsonschema-specifications		2024.10.1
python3-libcomps	0.1.20	
python3-libdnf	0.69.0	

Package	AL2023 Minimal	AL2027 Minimal
python3-libs	3.9.25	3.14.7
python3-libselinux	3.4	3.10
python3-libsemanage	3.4	3.10
python3-lxml	4.7.1	
python3-markupsafe	1.1.1	3.0.2
python3-netifaces	0.10.6	
python3-oauthlib	3.0.2	3.3.1
python3-pip-wheel	21.3.1	26.2.1
python3-ply	3.11	3.11
python3-policycoreutils	3.4	3.10
python3-prettytable	0.7.2	
python3-prompt-toolkit	3.0.24	3.0.41
python3-pycparser	2.20	2.22
python3-pyrsistent	0.17.3	
python3-pyserial	3.4	
python3-pysocks	1.7.1	
python3-pytz	2022.7.1	
python3-pyyaml	5.4.1	6.0.3
python3-referencing		0.36.2

Package	AL2023 Minimal	AL2027 Minimal
python3-requests	2.25.1	2.33.1
python3-rpds-py		0.27.0
python3-rpm	4.16.1.3	
python3-ruamel-yaml	0.16.6	0.19.1
python3-ruamel-yaml +oldlibyaml		0.19.1
python3-ruamel-yaml-clib	0.1.2	0.2.15
python3-setools	4.4.1	4.6.0
python3-setuptools	59.6.0	
python3-setuptools-wheel	59.6.0	
python3-six	1.15.0	1.17.0
python3-supportinfo	1.0.0	
python3-systemd	235	
python3-urllib3	1.25.10	2.6.3
python3-wcwidth	0.2.5	0.6.0
python3-xmlschema	1.4.2	
rdma-core-common		61.0
readline	8.1	8.3
rng-tools	6.17	

Package	AL2023 Minimal	AL2027 Minimal
rootfiles	8.1	9.0
rpm	4.16.1.3	6.0.0
rpm-build-libs	4.16.1.3	
rpm-libs	4.16.1.3	6.0.0
rpm-plugin-selinux	4.16.1.3	6.0.0
rpm-plugin-systemd-inhibit	4.16.1.3	6.0.0
rpm-sequoia		1.10.2.1
rpm-sign-libs	4.16.1.3	
sbsigntools	0.9.4	0.9.5
sdbus-cpp		2.2.1
sed	4.8	4.9
selinux-policy	38.1.76	44.5
selinux-policy-targeted	38.1.76	44.5
setup	2.13.7	2.15.1
shadow-utils	4.9	4.19.0
sqlite-libs	3.40.0	3.51.2
sudo	1.9.15	1.9.17
sysctl-defaults	1.0	1.0
system-release	2023.12.20260831	2027.0.20260903

Package	AL2023 Minimal	AL2027 Minimal
systemd	252.23	260.1
systemd-libs	252.23	260.1
systemd-networkd	252.23	260.1
systemd-pam	252.23	
systemd-resolved	252.23	260.1
systemd-shared		260.1
systemd-sysusers		260.1
systemd-udev	252.23	260.1
tar	1.34	1.35
tzdata	2026c	2026c
update-motd	2.3	2.3
userspace-rcu	0.12.1	0.15.6
util-linux	2.37.4	2.41.5
util-linux-core	2.37.4	2.41.5
vim-data	9.2.920	9.2.920
vim-minimal	9.2.920	9.2.920
which	2.21	2.25
xfsprogs	6.12.0	7.1.1
xkeyboard-config		2.47
xz	5.2.5	5.8.1

Package	AL2023 Minimal	AL2027 Minimal
xz-libs	5.2.5	5.8.1
yum	4.14.0	
zlib	1.2.11	
zlib-ng-compat		2.3.3
zram-generator	1.1.2	1.2.1
zram-generator-defaults	1.1.2	1.2.1
zstd	1.5.5	1.5.7

AL2027 system requirements

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

This section describes the system requirements for using AL2027.

Topics

- CPU requirements for running AL2027
- Memory (RAM) requirements for running AL2027

CPU requirements for running AL2027

To run any AL2027 code, the processor used needs to meet certain minimum requirements. Attempts to run AL2027 on CPUs that do not meet these requirements might result in illegal instruction errors very early in code execution.

The minimum requirements apply to AL2027 on Amazon EC2 and AL2027 in containers.

ARM CPU requirements for AL2027

All AL2027 `aarch64` (ARM) binaries are built for 64-bit. No 32-bit ARM binaries are available, so a 64-bit ARM CPU is required.

> **Note**
>
> For Arm-based instances, AL2027 only supports instance types that use Graviton2 or later processors. AL2027 doesn't support A1 instances.

AL2027 requires an ARMv8.2 compliant processor with the Cryptography Extension. All AL2027 packages for `aarch64` are built with the `-march=armv8.2-a+crypto -mtune=neoverse-n1` compiler flags. Although we attempt to print graceful error messages when AL2027 code is

attempted to be run on older ARM processors, it is possible that the first error message may be an illegal instruction error.

> **Note**
>
> Because of the AL2027 `aarch64` base CPU requirements, all Raspberry Pi systems prior to the Raspberry Pi 5 do not meet the minimum CPU requirements.

x86-64 CPU requirements for AL2027

All AL2027 `x86-64` binaries are built for the `x86-64-v3` revision of the `x86-64` architecture by passing `-march=x86-64-v3` to the compiler, with auto-vectorization enabled by default.

The `x86-64-v3` revision of the architecture requires the following CPU features on top of the `x86-64-v2` baseline:

- `AVX`
- `AVX2`
- `BMI1`
- `BMI2`
- `F16C`
- `FMA`
- `LZCNT`
- `MOVBE`
- `XSAVE`

This roughly maps to `x86-64` processors released in 2015 or later. Examples include the Intel Haswell and AMD Excavator microarchitectures and later.

Because of this requirement, AL2027 does not support several previous generation Amazon EC2 instance types that predate the `x86-64-v3` baseline. For the list of unsupported instance types, see AL2027 on Amazon EC2.

AL2027 does not support 32-bit x86 (i686) at all. No 32-bit AL2027 binaries are built, the repositories contain no i686 packages, and running 32-bit userspace binaries is not supported. For more information, see 32bit x86 (i686) runtime support.

Memory (RAM) requirements for running AL2027

The Amazon EC2 `.nano` family of instance types (`t3.nano`, `t3a.nano`, and `t4g.nano`) have 512 MB RAM, which is the minimum requirement for AL2027.

> **Note**
>
> Although 512 MB is the minimum requirement, these instance types are memory constrained and functionality and performance may be limited.

AL2027 images have not been tested on systems with less than 512 MB RAM. Running AL2027 based container images in less than 512 MB RAM will be dependent on the containerized workload.

For instances with less than 800 MB of RAM, AL2027 enables `zram` based swap by default. For containerized workloads, this means that some workloads might run fine on AL2027 instances with this amount of memory, but fail when run in a container restricted to this amount of memory usage.

Examples of Amazon EC2 instance types with less than 800 MB memory include `t4g.nano`, `t3a.nano`, and `t3.nano`. Enabling `zram` means fewer out of memory scenarios for these instance types, because AL2027 will on-demand compress and decompress memory pages. This enables workloads that would otherwise require an instance type with more memory, at the expense of CPU usage needed to do the compression.

Amazon Linux kernel

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

AL2027 ships with a single kernel major version at launch. Similar to AL2023, additional kernel versions will be released annually while AL2027 is under standard support.

Topics

- Using Multi-Gen LRU (MGLRU) on AL2027 kernels

Using Multi-Gen LRU (MGLRU) on AL2027 kernels

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

The Multi-Gen LRU is a modern page reclamation algorithm in the Linux kernel, designed to improve memory management performance under memory pressure. It replaces the traditional LRU (Least Recently Used) mechanism used for determining which memory pages to reclaim when the system runs low on memory.

The traditional LRU mechanism uses a two-list model (active and inactive) to track page usage, which can become inefficient in modern workloads with large working sets. MGLRU replaces this with multiple "generations" of pages, allowing the kernel to make smarter decisions based on finer-grained aging information.

Benefits of MGLRU include:

- **Better reclaim decisions:** More accurate identification of cold (unused) pages.
- **Lower latency and improved throughput:** Especially for workloads with large address spaces or many concurrent processes.

- **Improved cache retention:** Pages that are used recently are less likely to be evicted prematurely.
- **Scalable and lock-efficient design:** Performs better on machines with many CPUs.

Configuration and Tuning

The kernel configuration `CONFIG_LRU_GEN` is enabled on AL2027 kernels. This compiles in MGLRU but does not enable it by default.

MGLRU can be enabled and tuned using the `/sys/kernel/mm/lru_gen/enabled` file. The value is a bitmask. *It is recommended to enable all components unless some of them have undesirable side effects.*

Value	Components
`0x0001`	The main switch for the multi-gen LRU.
`0x0002`	Clearing the accessed bit in leaf page table entries in large batches, when MMU sets it (e.g., on x86). This behavior can theoretically worsen lock contention (`mmap_lock`). If it is disabled, the multi-gen LRU will suffer a minor performance degradation for workloads that contiguously map hot pages, whose accessed bits can be otherwise cleared by fewer larger batches.
`0x0004`	Clearing the accessed bit in non-leaf page table entries as well, when MMU sets it (e.g., on x86). This behavior was not verified on x86 varieties other than Intel and AMD. If it is disabled, the multi-gen LRU will suffer a negligible performance degradation.
`[yYnN]`	Apply to all the components above.

An example of how MGLRU can be enabled:

```
[ec2-user ~]$ echo y >/sys/kernel/mm/lru_gen/enabled
```

This enables all the components:

```
[ec2-user ~]$ cat /sys/kernel/mm/lru_gen/enabled
0x0007
```

AL2027 graphical desktop

AL2027 Preview

AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

AL2027 can run an optional graphical desktop environment. Use it for workloads that need a graphical interface instead of the command line, such as data visualization, computer-aided design (CAD), or scientific applications. This topic shows you how to install the GNOME desktop environment on an AL2027 instance along with some common workloads.

Topics

- Tutorial: Install the GNOME desktop environment
- Configure remote desktop using RDP on AL2027
- Tutorial: Install LibreOffice on AL2027

Tutorial: Install the GNOME desktop environment

The GNOME desktop environment is an optional graphical user interface for Amazon Linux 2027 (AL2027). The `Desktop` package group installs GNOME along with its supporting components.

In this tutorial, you install the GNOME desktop environment on your AL2027 instance so you can use a graphical interface instead of the command line. This is useful for workloads such as data visualization, computer-aided design (CAD), or scientific applications.

This tutorial takes approximately 15–20 minutes to complete.

Note

Completing this tutorial requires a running Amazon EC2 instance, which might result in charges to your AWS account for the instance, Amazon Elastic Block Store (Amazon EBS) storage, and associated resources. For pricing details, see Amazon EC2 pricing. To avoid ongoing charges, stop or terminate the instance when you are finished.

Topics

- Prerequisites
- Step 1: Install the GNOME desktop environment
- Step 2: Start the desktop
- Step 3: Clean up resources
- Related topics

Prerequisites

Before you install the desktop environment, make sure your instance meets the following requirements.

Memory (RAM)

The GNOME desktop environment requires **at least 2 GB of memory to start**, and **4 GB or more is recommended** for comfortable interactive use. Choose an instance type with sufficient memory, such as `t3.medium` (4 GB) or larger. Smaller types like `t3.nano`, `t3.micro`, and `t3.small` do not have enough memory for a responsive desktop.

Base system

This tutorial assumes that you have already launched an instance running a current release of AL2027 and that the system is up to date. For more information, see the AL2027 on Amazon EC2 and Updating AL2027 pages. Update your packages before installing the desktop so that dependencies resolve against the latest versions:

```
[ec2-user ~]$ sudo dnf update -y
```

Step 1: Install the GNOME desktop environment

1. Install the GNOME desktop environment and related packages.

   ```
   [ec2-user ~]$ sudo dnf groupinstall "Desktop" -y
   ```

   You can confirm what the group contains at any time with:

   ```
   [ec2-user ~]$ dnf group info "Desktop"
   ```

2. (Optional) Verify that the group is installed.

```
[ec2-user ~]$ dnf group list --installed
```

> **Note**
>
> To access the graphical desktop environment, you need to install and configure additional software such as Amazon DCV or VNC. These tools allow you to connect to and interact with the graphical user interface over the network.

Step 2: Start the desktop

The `Desktop` group installs the **GDM** display manager but does not start it automatically. Enable and start it so the graphical login is available now and on every boot:

```
[ec2-user ~]$ sudo systemctl enable --now gdm
```

To make the system boot into the graphical interface by default:

```
[ec2-user ~]$ sudo systemctl set-default graphical.target
```

You can verify the service and default target with:

```
[ec2-user ~]$ systemctl is-active gdm
[ec2-user ~]$ systemctl get-default
```

Step 3: Clean up resources

To avoid ongoing charges, clean up the resources you created for this tutorial when you are finished with them.

1. If you no longer need the instance, stop or terminate it. Stopping the instance ends compute charges but retains the Amazon EBS volume; terminating it removes the instance and its associated resources. For more information, see the AL2027 on Amazon EC2 page.
2. If you want to keep the instance but no longer need the graphical desktop, stop the graphical login and prevent it from starting on boot.

```
[ec2-user ~]$ sudo systemctl disable --now gdm
```

3. Restore the default boot target to the multiuser (non-graphical) target.

```
[ec2-user ~]$ sudo systemctl set-default multi-user.target
```

4. (Optional) Uninstall the desktop packages.

```
[ec2-user ~]$ sudo dnf groupremove "Desktop" -y
```

Related topics

For more information about the graphical desktop environment, see the following documentation:

- What Is Amazon DCV? in the *Amazon DCV Administrator Guide*

Configure remote desktop using RDP on AL2027

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

AL2027 does not include the TigerVNC server that was available in AL2023. To remotely access the graphical desktop, use the Remote Desktop Protocol (RDP) provided by GNOME Remote Desktop. GNOME Remote Desktop is included with the GNOME desktop environment. The following procedures configure the system-level RDP service. When you finish, you can connect to the graphical login screen and log in with your user account.

Contents

- Prerequisites
- Step 1: Install the desktop environment
- Step 2: Create a TLS certificate
- Step 3: Configure and enable the RDP service

- Step 4: Connect using an RDP client
- (Optional) Start the service at boot
- (Optional) Disable RDP access
- (Optional) Use your own TLS certificate
- Related topics

Prerequisites

- You use SSH port forwarding to access the RDP server. For more information about setting up your key pair, see Connect to your Linux instance using SSH in the *Amazon EC2 User Guide*.
- The following procedure does not guide you through the process of installing an RDP client. You must have an RDP client installed on your local machine to be able to connect to and interact with the desktop environment.

Step 1: Install the desktop environment

1. Connect to your instance using SSH.
2. Install the GNOME desktop environment and the `systemd-pam` package. The `Desktop` group includes GNOME Remote Desktop, which provides the RDP service. The `systemd-pam` package is required for the graphical login session.

```
[ec2-user ~]$ sudo dnf group install "Desktop"
[ec2-user ~]$ sudo dnf install systemd-pam
```

Step 2: Create a TLS certificate

RDP connections are always encrypted with TLS, so GNOME Remote Desktop requires a server certificate before the service can be enabled. The following steps create a self-signed certificate owned by the `gnome-remote-desktop` service user. If you want to use a certificate issued by a certificate authority instead, see the optional section at the end of this topic.

1. Create the directory that holds the certificate and key.

```
[ec2-user ~]$ sudo -u gnome-remote-desktop mkdir -p ~gnome-remote-desktop/.local/
share/gnome-remote-desktop
```

2. Generate a self-signed certificate and private key.

```
[ec2-user ~]$ sudo -u gnome-remote-desktop openssl req -new -newkey rsa:4096 -days
 720 -nodes -x509 \
  -subj "/C=US/ST=NONE/L=NONE/O=GNOME/CN=gnome.org" \
  -out ~gnome-remote-desktop/.local/share/gnome-remote-desktop/tls.crt \
  -keyout ~gnome-remote-desktop/.local/share/gnome-remote-desktop/tls.key
```

> **Note**
>
> This example command generates a certificate that is valid for 720 days, after which you need to regenerate it. Consider changing the parameters to suit your environment.

> **Note**
>
> The certificate and key must be readable by the `gnome-remote-desktop` service user, which is why the files are created with `sudo -u gnome-remote-desktop`.

Step 3: Configure and enable the RDP service

1. Configure the RDP service to use the certificate and key.

```
[ec2-user ~]$ sudo grdctl --system rdp set-tls-key ~gnome-remote-desktop/.local/
share/gnome-remote-desktop/tls.key
[ec2-user ~]$ sudo grdctl --system rdp set-tls-cert ~gnome-remote-desktop/.local/
share/gnome-remote-desktop/tls.crt
```

2. Set the RDP credentials. You are prompted for a username and password.

```
[ec2-user ~]$ sudo grdctl --system rdp set-credentials
```

> **Note**
>
> These credentials are used only to authenticate the RDP connection. They are not connected to a system user account, and the username does not need to match one. After the RDP connection is established, you log in to the desktop with your system user account at the graphical login screen.

3. Make sure the system user account you use at the graphical login screen has a password you can enter. For a local account that does not have a password, set one as shown. If the account already has a password, skip this step and use its existing credentials.

```
[ec2-user ~]$ sudo passwd <username>
```

> **Note**
>
> Replace `<username>` with the system user account you log in with (for example, `ec2-user`).

4. Enable the RDP service.

```
[ec2-user ~]$ sudo grdctl --system rdp enable
```

> **Note**
>
> On instances without a TPM device, `grdctl` prints `Init TPM credentials failed because No TPM device found, using GKeyFile as fallback`. This message is expected. The credentials are stored in a file that is readable only by the service user.

5. Start the graphical login screen.

```
[ec2-user ~]$ sudo systemctl start gdm.service
```

After performing this step, you can create the SSH tunnel from your local machine and connect using your RDP client.

Step 4: Connect using an RDP client

The RDP server listens on TCP port 3389. You can expose this port directly through your security group. Instead, these steps use SSH tunneling, which is more secure. The SSH tunnel authenticates the instance and encrypts the connection between your local machine and the EC2 instance, so port 3389 is never exposed to the network. For more information about security groups, see Change the security groups for your Amazon EC2 instance in the *Amazon EC2 User Guide*.

1. Create an SSH tunnel from your local machine.

```
$ ssh -i <keypair> -L <local-port>:localhost:3389 ec2-user@<address>
```

> **Note**
>
> Replace `<keypair>` with the path to your SSH key and `<address>` with your instance's public IP or DNS name. Replace `<local-port>` with any available port on your local machine (for example, `5000`). Don't change the remote port `3389`; that is the port the RDP server listens on. On some local machines, port `3389` is already in use, so choosing a different local port avoids conflicts.

2. Use your RDP client to connect to `localhost:<local-port>` (for example, `localhost:5000`) with the RDP credentials you configured in the previous step.

> **Note**
>
> Your RDP client might warn that the server certificate can't be verified. You can safely accept the warning here, because the SSH tunnel already authenticates the instance and encrypts the connection. Don't ignore certificate warnings when connecting directly over the network.

3. Log in with your system user account at the graphical login screen to start the desktop session.

> **Important**
>
> Keep the SSH tunnel open while using RDP. If the SSH tunnel isn't open, your RDP client can't view or interact with the desktop environment.

(Optional) Start the service at boot

If you plan to use RDP regularly, configure the instance to boot into the graphical target so that the login screen and the RDP service start automatically when your instance boots.

- To start the login screen and RDP service automatically at boot, set the default boot target.

  ```
  [ec2-user ~]$ sudo systemctl set-default graphical.target
  ```

  After performing this step, you will no longer need to start `gdm.service` every time you reboot your instance.

(Optional) Disable RDP access

If you no longer need remote desktop access, disable the RDP service so the instance stops listening for connections.

1. Disable the RDP service.

   ```
   [ec2-user ~]$ sudo grdctl --system rdp disable
   ```

2. If you changed the default boot target earlier, restore it to the multi-user (non-graphical) target.

   ```
   [ec2-user ~]$ sudo systemctl set-default multi-user.target
   ```

(Optional) Use your own TLS certificate

Instead of the self-signed certificate from Step 2, you can use a certificate issued by a certificate authority that your clients trust. Register it with the same `grdctl --system rdp set-tls-key` and `set-tls-cert` commands from Step 3. With this, your RDP client can verify the server identity, so you can connect directly to port 3389 without an SSH tunnel. If you connect directly, restrict inbound access to port 3389 in your security group to known source IP addresses. For instructions, see Change the security groups for your Amazon EC2 instance in the *Amazon EC2 User Guide*.

Related topics

For more information about the graphical desktop environment, see the following documentation:

- AL2027 graphical desktop
- What Is Amazon DCV? in the *Amazon DCV Administrator Guide*

Tutorial: Install LibreOffice on AL2027

AL2027 Preview

AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

LibreOffice is a free, open-source office productivity suite that includes applications for word processing, spreadsheets, presentations, diagrams, databases, and formula editing. You can install LibreOffice on Amazon Linux 2027 (AL2027) instances to provide users with a full-featured desktop office suite.

This tutorial takes approximately 15–20 minutes to complete.

Note

Completing this tutorial requires a running Amazon EC2 instance, which might result in charges to your AWS account for the instance, Amazon Elastic Block Store (Amazon EBS) storage, and associated resources. For pricing details, see Amazon EC2 pricing. To avoid ongoing charges, stop or terminate the instance when you are finished.

Contents

- Prerequisites
- Step 1: Determine your system architecture
- Step 2: Download the LibreOffice RPM package
- Step 3: Extract the archive
- Step 4: Install the RPM packages

- Step 5: Verify the installation
- Launching LibreOffice
- Clean up
- Troubleshooting

Prerequisites

- An Amazon EC2 instance running AL2027 with a graphical desktop environment configured and accessible through a remote desktop protocol (for example, Amazon DCV). For more information, see the following:
  - the section called “Install GNOME”
  - Setting up the Amazon DCV server
- sudo or root access on the instance.
- At least 1.5 GB of free disk space for the installation.
- An active internet connection to download the LibreOffice package, or the package pre-downloaded and transferred to the instance.

Step 1: Determine your system architecture

Run the following command to identify your processor architecture. This determines which RPM package to download.

```
uname -m
```

The output is one of the following:

Output	Architecture	RPM package type
`x86_64`	64-bit Intel/AMD	`Linux_x86-64_rpm`
`aarch64`	64-bit ARM (Graviton)	`Linux_aarch64_rpm`

Step 2: Download the LibreOffice RPM package

To download the LibreOffice RPM package

1. Open the LibreOffice download page.
2. Select **Linux (64-bit) (rpm)** as the operating system.
3. Choose the version appropriate for your architecture (x86_64 or aarch64).
4. Download the `.tar.gz` archive to your instance. You can use `wget` or `curl` to download directly from the terminal.

```
cd ~/Downloads
curl -OL TAR_DOWNLOAD_URL
```

Step 3: Extract the archive

Navigate to the directory where you downloaded the archive and extract it.

```
cd ~/Downloads
tar -xf LibreOffice_VERSION_Linux_ARCHITECTURE_rpm.tar.gz
```

This creates a directory with a name similar to `LibreOffice_VERSION_Linux_ARCHITECTURE_rpm/`.

Step 4: Install the RPM packages

To install the RPM packages

1. Change to the `RPMS/` subdirectory inside the extracted folder.

```
cd LibreOffice_VERSION_Linux_ARCHITECTURE_rpm/RPMS/
```

2. Install all RPM packages using `sudo rpm`.

```
sudo rpm -i *.rpm
```

Alternatively, you can use `dnf` for dependency resolution:

```
sudo dnf install -y *.rpm
```

Step 5: Verify the installation

Confirm that LibreOffice was installed successfully by checking the version.

```
libreoffice --version
```

Expected output:

```
LibreOffice VERSION ...
```

Launching LibreOffice

From the terminal

Run the following command:

```
libreoffice
```

If the `libreoffice` command is not available, the binary may be versioned. You can launch it using the versioned name instead:

```
libreofficeMAJOR_VERSION
```

For example, `libreoffice26.8`.

To launch a specific application directly:

```
libreoffice --writer    # Word processor
libreoffice --calc      # Spreadsheet
libreoffice --impress   # Presentations
libreoffice --draw      # Diagrams
libreoffice --base      # Database
libreoffice --math      # Formula editor
```

From the graphical desktop

After installation, LibreOffice applications appear in the desktop applications menu. Open the applications menu, search for "LibreOffice", and select the desired application.

Clean up

To avoid incurring charges, uninstall LibreOffice by running the following command, or terminate the Amazon EC2 instance if it was created only for this tutorial.

```
sudo dnf remove libreoffice*
```

Troubleshooting

The following section can help you troubleshoot common issues when installing or running LibreOffice on AL2027.

Java Runtime Environment warning

Some LibreOffice features (such as Base and certain extensions) require a Java Runtime Environment (JRE). If you see a warning about missing Java, install it:

```
sudo dnf install -y java-25-amazon-corretto-headless
```

Then, in LibreOffice, go to **Tools** > **Options** > **LibreOffice** > **Advanced** and verify that the JRE is detected.

LibreOffice command not found

If the `libreoffice` command is not found after installation, the binary may be versioned. Try running:

```
libreoffice MAJOR_VERSION
```

For example, `libreoffice26.8`. You can also locate the binary:

```
find /opt -name "soffice" 2>/dev/null
```

Additional resources

If you encounter other issues, refer to the official LibreOffice installation documentation for Linux:

- LibreOffice Linux Installation Guide
- LibreOffice Download Page

For more information on AL2027 graphical desktop, refer to our guides:

- *Graphical desktop*

Using AL2027 on AWS

AL2027 Preview

AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

You can set up AL2027 for use with other AWS services. For example, you can choose an AL2027 AMI when you launch an Amazon Elastic Compute Cloud (Amazon EC2) instance.

For these setup procedures, you use the AWS Identity and Access Management (IAM) service. For complete information about IAM, see the following reference materials:

- AWS Identity and Access Management (IAM)
- IAM User Guide

Topics

- Getting started with AWS
- AL2027 on Amazon EC2
- Using AL2027 in containers
- Using Amazon Elastic File System on AL2027

Getting started with AWS

Sign up for an AWS account

To get started with AWS, you need an AWS account. For information about creating an AWS account, see Getting started with an AWS account in the *AWS Account Management Reference Guide*.

Granting programmatic access

Users need programmatic access if they want to interact with AWS outside of the AWS Management Console. The way to grant programmatic access depends on the type of user that's accessing AWS.

To grant users programmatic access, choose one of the following options.

Which user needs programmatic access?	To	By
IAM	(Recommended) Use console credentials as temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions for the interface that you want to use. • For the AWS CLI, see Login for AWS local development in the *AWS Command Line Interface User Guide*. • For AWS SDKs, see Login for AWS local development in the *AWS SDKs and Tools Reference Guide*.
Workforce identity (Users managed in IAM Identity Center)	Use temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions for the interface that you want to use. • For the AWS CLI, see Configuring the AWS CLI to use AWS IAM Identity Center in the *AWS Command Line Interface User Guide*. • For AWS SDKs, tools, and AWS APIs, see IAM Identity Center authentication in the *AWS SDKs and Tools Reference Guide*.
IAM	Use temporary credentials to sign programmatic requests	Following the instructions in Using temporary credentia

Which user needs programmatic access?	To	By
	to the AWS CLI, AWS SDKs, or AWS APIs.	ls with AWS resources in the *IAM User Guide*.
IAM	(Not recommended) Use long-term credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions for the interface that you want to use. • For the AWS CLI, see Authenticating using IAM user credentials in the *AWS Command Line Interface User Guide*. • For AWS SDKs and tools, see Authenticate using long-term credentials in the *AWS SDKs and Tools Reference Guide*. • For AWS APIs, see Managing access keys for IAM users in the *IAM User Guide*.

AL2027 on Amazon EC2

> ⚠ **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

Use one of the following procedures to launch an Amazon EC2 instance with an AL2027 AMI. You can choose either the standard AMI or the minimal AMI.

Note

For x86-64 instances, AL2027 requires x86-64-v3 or later processors. The following instance types are not supported: `c1`, `c3`, `i2`, `m1`, `m2`, `m3`, `r3`, and `t1`.
For Arm-based instances, AL2027 requires Graviton2 or later processors. `a1` instances are not supported.

Launching AL2027 using the Amazon EC2 console

To launch an Amazon EC2 instance with an AL2027 AMI

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose **AMIs**.
3. From the filter drop-down, choose **Public images**.
4. In the search field, enter **`al2027-preview-ami`**.

   Note

   Make sure that **amazon** appears in the **Owner alias** column.

5. Select an image from the list. Under **Source**, you can determine whether the AMI is standard or minimal. An AL2027 AMI name uses this format:

   ```
   al2027-preview-[ami || ami-minimal]-2027.0.[release build date].[build number]-kernel-[version]-[arm64 || x86_64]
   ```

6. Choose **Launch instance from AMI** and configure the instance type, networking, storage, and key pair as needed.

For more information about launching Amazon EC2 instances, see Get started with Amazon EC2 Linux instances in the *Amazon EC2 User Guide*.

Launching AL2027 using the SSM parameter and AWS CLI

In the AWS CLI, you can use an AMI SSM parameter value to launch a new instance of AL2027. Use one of the following dynamic SSM parameter values with `/aws/service/ami-amazon-linux-latest/` as the prefix:

- `al2027-preview-ami-kernel-default-arm64` for arm64 architecture
- `al2027-preview-ami-minimal-kernel-default-arm64` for arm64 architecture (minimal AMI)
- `al2027-preview-ami-kernel-default-x86_64` for x86_64 architecture
- `al2027-preview-ami-minimal-kernel-default-x86_64` for x86_64 architecture (minimal AMI)

Note

Each of the following items in the command is a placeholder. Replace them with your own values: *m5.xlarge*, *us-east-1*, *my-key-pair*, and *sg-004a7650*.

```
$ aws ec2 run-instances \
  --image-id \
    resolve:ssm:/aws/service/ami-amazon-linux-latest/al2027-preview-ami-kernel-default-
x86_64 \
  --instance-type m5.xlarge \
  --region us-east-1 \
  --key-name my-key-pair \
  --security-group-ids sg-004a7650
```

The `--image-id` flag specifies the SSM parameter value. The `resolve:ssm:` prefix tells the AWS CLI to resolve the parameter to the latest AMI ID at launch time.

The `--instance-type` flag specifies the type and size of the instance. This must be compatible with the AMI architecture you selected.

The `--region` flag specifies the AWS Region where you create your instance.

The `--key-name` flag specifies the key pair used to connect to the instance using SSH.

The `--security-group-ids` flag specifies the security group that determines access permissions for inbound and outbound network traffic.

> **Important**
>
> The AWS CLI requires that you specify an existing security group that allows access to the instance from your remote machine over port TCP:22. Without a specified security group, your new instance is placed in a default security group.

For more information, see Launching, listing, and terminating Amazon EC2 instances in the *AWS Command Line Interface User Guide*.

Launching AL2027 using a specific AMI ID

You can launch a specific AL2027 AMI using the AMI ID. You can determine which AL2027 AMI ID is needed by looking at the AMI list in the Amazon EC2 console. Or, you can use AWS Systems Manager. For more information, see Query for the latest Amazon Linux AMI IDs using AWS Systems Manager Parameter Store.

Connect to your instance

After your instance is running, connect to it using SSH:

```
ssh -i <your-key.pem> ec2-user@<instance-public-ip>
```

The default user for AL2027 instances is `ec2-user`.

Connecting to AL2027 instances

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

Use SSH or AWS Systems Manager to connect to your AL2027 instance.

Connect to your instance using SSH

For instructions on how to use SSH to connect to an instance, see Connect to your Linux instance using SSH in the *Amazon EC2 User Guide*.

Connect to your instance using AWS Systems Manager

For instructions on how to use AWS Systems Manager to connect to an AL2027 instance, see Connect to your Linux instance using Session Manager in the *Amazon EC2 User Guide*.

Using Amazon EC2 Instance Connect

To use EC2 Instance Connect with an AL2027 instance, you must install the `ec2-instance-connect` package. For instructions on using EC2 Instance Connect, see Connect to your Linux instance with EC2 Instance Connect in the *Amazon EC2 User Guide*.

Comparing AL2027 standard and minimal AMIs

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

You can launch an Amazon EC2 instance with either a standard (default) or minimal AL2027 AMI. For instructions on how to launch an Amazon EC2 instance with the standard or minimal AMI type, see AL2027 on Amazon EC2.

The standard AL2027 AMI comes with all of the most commonly used applications and tools installed. We recommend the standard AMI if you want to get started quickly and aren't interested in customizing the AMI.

The minimal AL2027 AMI is a streamlined version that contains only essential tools and utilities necessary to run the operating system. We recommend it when you want the smallest OS footprint possible, with slightly reduced disk space utilization and better long-term cost efficiency. Use the minimal AMI when you are comfortable manually installing additional tools and applications as needed.

The container image is closer to the AL2027 minimal AMI in package set.

The set of packages installed by default differs across the AL2027 images. Use the following table to compare the packages present on the AL2027 standard AMI, minimal AMI, and container image, along with their versions. A blank cell means the package is not installed by default in that image.

Comparing packages installed on Amazon Linux 2027 Images

A comparison of the RPMs present on the AL2027 AMI, Minimal AMI, and Container images.

Package	AMI	Minimal AMI	Container
acl	2.4.0		
acpid	2.0.34		
alternatives	1.33	1.33	1.33
amazon-chrony-config	4.8	4.8	
amazon-ec2-net-utils	2.7.6	2.7.6	
amazon-linux-repo-cdn			2027.0.20260903
amazon-linux-repo-s3	2027.0.20260903	2027.0.20260903	
amazon-linux-sb-keys	2027.1	2027.1	
amazon-ssm-agent	3.3.5226.0		
attr	2.5.2		
audit	4.1.3	4.1.3	
audit-libs	4.1.3	4.1.3	
audit-rules	4.1.3	4.1.3	
authselect	1.7.1	1.7.1	
authselect-libs	1.7.1	1.7.1	

Package	AMI	Minimal AMI	Container
aws-cfn-bootstrap	2.0		
aws-lc-libs	5.2.0	5.2.0	
awscli-2	2.36.2	2.36.2	
bash	5.3.0	5.3.0	5.3.0
bash-completion	2.17		
bzip2	1.0.8		
bzip2-libs	1.0.8	1.0.8	1.0.8
c-ares	1.34.6		
ca-certificates	2025.2.80_v9.0.305	2025.2.80_v9.0.305	2025.2.80_v9.0.305
checkpolicy	3.10	3.10	
chrony	4.8	4.8	
cloud-init	26.1	26.1	
cloud-init-cfg-ec2	26.1	26.1	
cloud-utils-growpart	0.33	0.33	
compact-dnf-utils	5.4.2.1		
coreutils	9.10	9.10	
coreutils-common	9.10	9.10	

Package	AMI	Minimal AMI	Container
coreutils-single			9.10
cpio	2.15	2.15	
cracklib	2.10.3	2.10.3	
crypto-policies	20260525	20260525	20260525
cryptsetup-libs	2.8.4	2.8.4	
curl	8.18.0	8.18.0	8.18.0
cyrus-sasl-gssapi	2.1.28		
cyrus-sasl-lib	2.1.28	2.1.28	
dbus	1.16.0	1.16.0	
dbus-broker	37	37	
dbus-common	1.16.0	1.16.0	
dbus-libs	1.16.0	1.16.0	
device-mapper	1.02.212	1.02.212	
device-mapper-libs	1.02.212	1.02.212	
diffutils	3.12	3.12	
dnf-plugin-release-notification	2.0.0	2.0.0	
dnf-plugin-support-info	3.0.0	3.0.0	

Package	AMI	Minimal AMI	Container
dnf5	5.4.2.1	5.4.2.1	5.4.2.1
dnf5-plugins	5.4.2.1		
dracut	108	108	
dracut-config-ec2	3.1	3.1	
dracut-config-generic	108	108	
e2fsprogs	1.47.3	1.47.3	
e2fsprogs-libs	1.47.3	1.47.3	
ec2-instance-connect	1.1	1.1	
ec2-instance-connect-selinux	1.1	1.1	
ec2-utils	2.2.0	2.2.0	
efi-filesystem	6	6	
efivar	39	39	
efivar-libs	39	39	
elfutils-default-yama-scope	0.195		
elfutils-libelf	0.195	0.195	
elfutils-libs	0.195		

Package	AMI	Minimal AMI	Container
expat	2.8.1	2.8.1	
file	5.46	5.46	
file-libs	5.46	5.46	
filesystem	3.18	3.18	3.18
findutils	4.10.0	4.10.0	4.10.0
fmt	11.2.0	11.2.0	11.2.0
fuse3-libs	3.18.2	3.18.2	
gawk	5.3.2	5.3.2	5.3.2
gdbm	1.23	1.23	
gdbm-libs	1.23	1.23	
gdisk	1.0.10	1.0.10	
gettext	1.0	1.0	
gettext-envsubst	1.0	1.0	
gettext-libs	1.0	1.0	
gettext-runtime	1.0	1.0	
glib2	2.88.1	2.88.1	2.88.1
glibc	2.44	2.44	2.44
glibc-common	2.44	2.44	2.44
glibc-minimal-langpack	2.44	2.44	2.44

Package	AMI	Minimal AMI	Container
gmp	6.3.0	6.3.0	6.3.0
gnulib-l10n	20241231	20241231	
gnupg2	2.4.9		
gnupg2-dirmngr	2.4.9		
gnupg2-gpg-agent	2.4.9		
gnupg2-gpgconf	2.4.9		
gnupg2-keyboxd	2.4.9		
gnupg2-verify	2.4.9		
gnutls	3.8.10	3.8.10	
grep	3.12	3.12	3.12
groff-base	1.23.0		
grub2-common	2.12	2.12	
grub2-efi-aa64-ec2	2.12	2.12	
grub2-tools	2.12	2.12	
grub2-tools-minimal	2.12	2.12	
grubby	8.40	8.40	
gzip	1.14	1.14	
hostname	3.25	3.25	

Package	AMI	Minimal AMI	Container
hwdata	0.409	0.409	
ima-evm-utils-libs	1.6.2		
inih	62	62	
iproute	6.17.0	6.17.0	
iputils	20250605	20250605	
irqbalance	1.9.5	1.9.5	
jansson	2.14		
jq	1.8.1	1.8.1	
json-c	0.18	0.18	0.18
kbd	2.10.0	2.10.0	
kbd-legacy	2.10.0	2.10.0	
kbd-misc	2.10.0	2.10.0	
kernel7.1	7.1.0	7.1.0	
kernel7.1-tools	7.1.0		
keyutils-libs	1.6.3	1.6.3	1.6.3
kmod	34.2	34.2	
kmod-libs	34.2	34.2	
krb5-libs	1.22.2	1.22.2	1.22.2
less	702	702	
libacl	2.4.0	2.4.0	2.4.0

Package	AMI	Minimal AMI	Container
libarchive	3.8.8	3.8.8	3.8.8
libassuan	2.5.7		
libattr	2.5.2	2.5.2	2.5.2
libbasicobjects	0.1.1		
libblkid	2.41.5	2.41.5	2.41.5
libbpf	1.6.3	1.6.3	
libcap	2.78	2.78	2.78
libcap-ng	0.9.3	0.9.3	
libcbor	0.13.0	0.13.0	
libcollection	0.7.0		
libcom_err	1.47.3	1.47.3	1.47.3
libcurl-minimal	8.18.0	8.18.0	8.18.0
libdhash	0.5.0		
libdnf5	5.4.2.1	5.4.2.1	5.4.2.1
libdnf5-cli	5.4.2.1	5.4.2.1	5.4.2.1
libeconf	0.7.9	0.7.9	
libedit	3.1	3.1	
libevent	2.1.12	2.1.12	
libfdisk	2.41.5	2.41.5	
libffi	3.5.2	3.5.2	3.5.2

Package	AMI	Minimal AMI	Container
libfido2	1.16.0	1.16.0	
libgcc	16.1.1	16.1.1	16.1.1
libgcrypt	1.11.1		
libgomp	16.1.1	16.1.1	
libgpg-error	1.50		
libibverbs	61.0	61.0	
libicu	77.1		
libidn2	2.3.8	2.3.8	2.3.8
libini_config	1.3.1		
libkcapi	1.5.0	1.5.0	
libkcapi-hasher	1.5.0	1.5.0	
libkcapi-hmaccalc	1.5.0	1.5.0	
libksba	1.6.7		
liblastlog2	2.41.5	2.41.5	
libldb	4.24.6		
libmnl	1.0.5	1.0.5	
libmount	2.41.5	2.41.5	2.41.5
libnghttp2	1.68.0	1.68.0	1.68.0
libnl3	3.12.0	3.12.0	
libpath_utils	0.2.1		

Package	AMI	Minimal AMI	Container
libpcap	1.10.6	1.10.6	
libpipeline	1.5.8		
libpkgconf	2.5.1	2.5.1	
libpsl	0.21.5	0.21.5	0.21.5
libpwquality	1.4.5	1.4.5	
libref_array	0.1.5		
librepo	1.20.0	1.20.0	1.20.0
libseccomp	2.6.1	2.6.1	
libselinux	3.10	3.10	3.10
libselinux-utils	3.10	3.10	
libsemanage	3.10	3.10	
libsepol	3.10	3.10	3.10
libsmartcols	2.41.5	2.41.5	2.41.5
libsolv	0.7.39	0.7.39	0.7.39
libss	1.47.3	1.47.3	
libsss_certmap	2.13.1		
libsss_idmap	2.13.1		
libsss_nss_idmap	2.13.1		
libsss_sudo	2.13.1		

Package	AMI	Minimal AMI	Container
libstdc++	16.1.1	16.1.1	16.1.1
libsupportinfo	2.0.0	2.0.0	
libtalloc	2.4.4		
libtasn1	4.20.0	4.20.0	4.20.0
libtdb	1.4.15		
libtevent	0.17.1		
libtextstyle	1.0	1.0	
libtirpc	1.3.7		
libtool-ltdl	2.5.4	2.5.4	
libunistring	1.1	1.1	1.1
libusb1	1.0.30		
libuuid	2.41.5	2.41.5	2.41.5
libverto	0.3.2	0.3.2	0.3.2
libxcrypt	4.5.2	4.5.2	
libxkbcommon	1.13.1	1.13.1	
libxml2	2.12.10	2.12.10	2.12.10
libyaml	0.2.5	0.2.5	
libzstd	1.5.7	1.5.7	1.5.7
lmdb-libs	0.9.34		
logrotate	3.22.0	3.22.0	

Package	AMI	Minimal AMI	Container
lsof	4.98.0		
lua-libs	5.4.8	5.4.8	5.4.8
lz4-libs	1.10.0	1.10.0	1.10.0
man-db	2.13.1		
man-pages	6.13		
mpdecimal	4.0.1	4.0.1	
mpfr	4.2.2	4.2.2	4.2.2
nano	8.7.1		
ncurses		6.6	
ncurses-base	6.6	6.6	6.6
ncurses-libs	6.6	6.6	6.6
nettle	3.10.1	3.10.1	
nmap-ncat	7.93	7.93	
npth	1.8		
numactl-libs	2.0.19	2.0.19	
oniguruma	6.9.10	6.9.10	
openldap	2.6.13	2.6.13	
openssh	9.9p1	9.9p1	
openssh-clients	9.9p1	9.9p1	
openssh-server	9.9p1	9.9p1	

Package	AMI	Minimal AMI	Container
openssl	3.5.7	3.5.7	
openssl-fips-provider-latest	3.5.7	3.5.7	3.5.7
openssl-libs	3.5.7	3.5.7	3.5.7
os-prober	1.81	1.81	
p11-kit	0.26.2	0.26.2	0.26.2
p11-kit-trust	0.26.2	0.26.2	0.26.2
pam	1.7.1	1.7.1	
pam-libs	1.7.1	1.7.1	
pciutils	3.15.0	3.15.0	
pciutils-libs	3.15.0	3.15.0	
pcre2	10.47	10.47	10.47
pcre2-syntax	10.47	10.47	10.47
pkgconf	2.5.1	2.5.1	
pkgconf-m4	2.5.1	2.5.1	
pkgconf-pkg-config	2.5.1	2.5.1	
policycoreutils	3.10	3.10	
policycoreutils-python-utils	3.10	3.10	

Package	AMI	Minimal AMI	Container
popt	1.19	1.19	1.19
procps-ng	4.0.6	4.0.6	
psmisc	23.7	23.7	
publicsuffix-list-dafsa	20260116	20260116	20260116
python-chevron	0.14.0		
python3	3.14.7	3.14.7	
python3-attrs	25.4.0	25.4.0	
python3-audit	4.1.3	4.1.3	
python3-awscrt	0.36.0	0.36.0	
python3-cffi		2.0.0	
python3-charset-normalizer	3.4.4	3.4.4	
python3-colorama	0.4.6	0.4.6	
python3-configobj	5.0.9	5.0.9	
python3-cryptography		49.0.0	
python3-daemon	3.1.0		
python3-dateutil	2.9.0.post0	2.9.0.post0	

Package	AMI	Minimal AMI	Container
python3-distro	1.9.0	1.9.0	
python3-docutils	0.21.2	0.21.2	
python3-idna	3.11	3.11	
python3-jinja2	3.1.6	3.1.6	
python3-jmespath	1.0.1	1.0.1	
python3-jsonpatch	1.33	1.33	
python3-jsonpointer	2.4	2.4	
python3-jsonschema	4.23.0	4.23.0	
python3-jsonschema-specifications	2024.10.1	2024.10.1	
python3-libs	3.14.7	3.14.7	
python3-libselinux	3.10	3.10	
python3-libsemanage	3.10	3.10	
python3-lockfile	0.12.2		
python3-markupsafe	3.0.2	3.0.2	

Package	AMI	Minimal AMI	Container
python3-oauthlib	3.3.1	3.3.1	
python3-pip-wheel	26.2.1	26.2.1	
python3-ply		3.11	
python3-policycoreutils	3.10	3.10	
python3-prompt-toolkit	3.0.41	3.0.41	
python3-pycparser		2.22	
python3-pyyaml	6.0.3	6.0.3	
python3-referencing	0.36.2	0.36.2	
python3-requests	2.33.1	2.33.1	
python3-rpds-py	0.27.0	0.27.0	
python3-ruamel-yaml	0.19.1	0.19.1	
python3-ruamel-yaml+oldlibyaml	0.19.1	0.19.1	
python3-ruamel-yaml-clib	0.2.15	0.2.15	
python3-setools	4.6.0	4.6.0	

Package	AMI	Minimal AMI	Container
python3-six	1.17.0	1.17.0	
python3-urllib3	2.6.3	2.6.3	
python3-wcwidth	0.6.0	0.6.0	
rdma-core-common	61.0	61.0	
readline	8.3	8.3	8.3
rootfiles	9.0	9.0	
rpm	6.0.0	6.0.0	6.0.0
rpm-build-libs	6.0.0		
rpm-libs	6.0.0	6.0.0	6.0.0
rpm-plugin-selinux	6.0.0	6.0.0	
rpm-plugin-systemd-inhibit	6.0.0	6.0.0	
rpm-sequoia	1.10.2.1	1.10.2.1	1.10.2.1
rpm-sign-libs	6.0.0		
samba-common	4.24.6		
samba-core-libs	4.24.6		
samba-ndr-libs	4.24.6		
sbsigntools	0.9.5	0.9.5	
sdbus-cpp	2.2.1	2.2.1	2.2.1

Package	AMI	Minimal AMI	Container
sed	4.9	4.9	4.9
selinux-policy	44.5	44.5	
selinux-policy-targeted	44.5	44.5	
setup	2.15.1	2.15.1	2.15.1
shadow-utils	4.19.0	4.19.0	
sqlite-libs	3.51.2	3.51.2	3.51.2
sssd-client	2.13.1		
sssd-common	2.13.1		
sssd-kcm	2.13.1		
sssd-krb5-common	2.13.1		
sudo	1.9.17	1.9.17	
sysctl-defaults	1.0	1.0	
system-release	2027.0.20260903	2027.0.20260903	2027.0.20260903
systemd	260.1	260.1	
systemd-libs	260.1	260.1	260.1
systemd-networkd	260.1	260.1	
systemd-pam	260.1		
systemd-resolved	260.1	260.1	

Package	AMI	Minimal AMI	Container
systemd-shared	260.1	260.1	
systemd-standalone-sysusers			260.1
systemd-sysusers	260.1	260.1	
systemd-udev	260.1	260.1	
tar	1.35	1.35	1.35
tpm2-tss	4.1.3		
tzdata	2026c	2026c	
unzip	6.0		
update-motd	2.3	2.3	
userspace-rcu	0.15.6	0.15.6	
util-linux	2.41.5	2.41.5	
util-linux-core	2.41.5	2.41.5	
vim-data	9.2.920	9.2.920	
vim-minimal	9.2.920	9.2.920	
which	2.25	2.25	
xfsprogs	7.1.1	7.1.1	
xkeyboard-config	2.47	2.47	
xz	5.8.1	5.8.1	

Package	AMI	Minimal AMI	Container
xz-libs	5.8.1	5.8.1	5.8.1
zip	3.0		
zlib-ng-compat	2.3.3	2.3.3	2.3.3
zram-generator	1.2.1	1.2.1	
zram-generator-defaults	1.2.1	1.2.1	
zstd	1.5.7	1.5.7	

Using AL2027 in containers

AL2027 Preview

AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

Containers provide a consistent AL2027 environment for developing and running applications. Choose an option based on how you want to create the container.

- Use the AL2027 base container image for a ready-to-run AL2027 environment or as the base for your own container image.
- Use the AL2027 minimal container image as a separate image variant. Add the packages that your application requires.
- Use the Bare-bones AL2027 container images workflow to build an application-specific root filesystem. DNF5 resolves the package dependencies.

Topics

- Using the AL2027 base container image
- AL2027 minimal container image

- Building bare-bones AL2027 container images
- Comparing packages installed on Amazon Linux 2027 Container Images
- Comparing packages installed on Amazon Linux 2027 Minimal AMI and Container Images

Using the AL2027 base container image

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

The AL2027 base container image provides an AL2027 environment for local development and containerized workloads. You can use the image directly or as the base for your own container images.

The base container image is available in the Amazon ECR Public Gallery. Public repositories support unauthenticated pulls. You can optionally authenticate your Docker client before pulling the image.

Authenticate to Amazon ECR Public (optional)

Run the following command to authenticate Docker to Amazon ECR Public. Always use the `us-east-1` Region when you retrieve an Amazon ECR Public authentication token. The token is valid for 12 hours.

```
$ aws ecr-public get-login-password --region us-east-1 | docker login --username AWS --password-stdin public.ecr.aws
```

For more information, see Registry authentication in Amazon ECR Public in the *Amazon ECR Public User Guide*.

Pull and run the base container image

To pull and run the AL2027 base container image

1. Pull the image from Amazon ECR Public.

   ```
   $ docker pull public.ecr.aws/amazonlinux/amazonlinux:2027
   ```

2. Run an interactive shell in the container.

```
$ docker run --rm -it public.ecr.aws/amazonlinux/amazonlinux:2027 /bin/bash
```

Manage software packages

The AL2027 base container image uses DNF5. Use the **dnf** command to manage software packages in the container. The **dnf** command invokes DNF5.

AL2027 minimal container image

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

The AL2027 minimal container image is a separate container image variant. You can use it as a starting point and install the packages that your application requires.

The minimal container image is available in the Amazon ECR Public Gallery.

Minimal container image size

Because the AL2027 minimal container image contains fewer packages than the AL2027 base container image, it is also smaller. The following table compares the AL2027 container image options with the equivalent AL2023 container images.

> **Note**
>
> Image Size is as-shown on Amazon Linux on Amazon ECR Public Gallery.

Image	Version	Image Size
Amazon Linux 2023 base container image	2023.12.20260831.0	54.6MB

Image	Version	Image Size
Amazon Linux 2023 minimal container image	2023.12.20260831.0-minimal	37.4MB
Amazon Linux 2027 base container image	2027.0.20260903.0	44.8MB
Amazon Linux 2027 minimal container image	2027.0.20260903.0-minimal	38.9MB

Pull the minimal container image

Public repositories support unauthenticated pulls. For optional authentication, see Authenticate to Amazon ECR Public (optional).

```
$ docker pull public.ecr.aws/amazonlinux/amazonlinux:2027-minimal
```

Use the minimal image in a Dockerfile

The minimal container image uses DNF5. Use the **dnf** command to install packages. The following example installs GCC and removes cached repository data.

```
FROM public.ecr.aws/amazonlinux/amazonlinux:2027-minimal
RUN dnf install -y gcc && dnf clean all
```

Building bare-bones AL2027 container images

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

A bare-bones container image starts with an empty root filesystem. You add the packages that your application requires. Use a multi-stage build to resolve and install those packages with DNF5. The final stage copies the root filesystem into a `FROM scratch` image.

Runtime dependencies

The final image contains only the packages and files that the build stage adds to `/sysroot`. The examples on this page don't include the `dnf` or `rpm` commands, common diagnostic utilities such as `ls` and `du`, or the CA certificate bundle. Add every tool and runtime dependency that your application requires.
Rebuild the image from an updated AL2027 base image to include package updates.

Prerequisites

Install and start Docker on the system that you use to build the image. On AL2027, run the following commands.

```
$ sudo dnf install -y docker
$ sudo systemctl start docker
```

The examples assume that your user can run `docker` commands without `sudo`. For information about configuring access to the Docker daemon, see Linux post-installation steps for Docker Engine in the Docker documentation.

Build an image with Bash

The following procedure creates a container image that contains Bash and the package dependencies that DNF5 resolves for it.

To build a bare-bones image with Bash

1. Create and open a directory for the example.

   ```
   $ mkdir al2027-barebones-bash-example
   $ cd al2027-barebones-bash-example
   ```

2. Create a file named `Dockerfile` with the following content.

   ```
   FROM public.ecr.aws/amazonlinux/amazonlinux:2027 AS build
   RUN mkdir /sysroot
   RUN dnf --releasever=$(rpm -q system-release --qf '%{VERSION}') \
     --installroot /sysroot \
   ```

```
  --use-host-config \
  -y \
  --setopt=install_weak_deps=False \
  install bash \
  && dnf --installroot /sysroot clean all

FROM scratch
COPY --from=build /sysroot /
WORKDIR /
ENTRYPOINT ["/bin/bash"]
```

In the build stage, DNF5 uses the repository configuration from the AL2027 base image to install packages into `/sysroot`. The following options configure the installation.

- `--releasever` uses the version of the `system-release` package in the build-stage image.
- `--installroot` installs packages into `/sysroot` instead of the build-stage root filesystem.
- `--use-host-config` uses the DNF5 configuration and repositories from the build-stage image. It applies this configuration to the empty install root.
- `--setopt=install_weak_deps=False` excludes suggested and recommended packages from dependency resolution.

After DNF5 removes its cached repository data, the final stage copies `/sysroot` into an empty image and sets Bash as the entry point.

3. Build the container image.

```
$ docker build -t al2027-barebones-bash-example .
```

4. Run Bash in the container.

```
$ docker run --rm -it al2027-barebones-bash-example
```

Build an image for a compiled application

The following procedure compiles a C application in the build stage and copies the application and its DNF5-resolved runtime package closure into the final image.

To build a bare-bones image for a C application

1. Create and open a directory for the example.

```
$ mkdir al2027-barebones-c-example
$ cd al2027-barebones-c-example
```

2. Create a file named `hello-world.c` with the following content.

```
#include <stdio.h>

int main(void)
{
    printf("Hello World!\n");
    return 0;
}
```

3. Create a file named `Dockerfile` with the following content.

```
FROM public.ecr.aws/amazonlinux/amazonlinux:2027 AS build
COPY hello-world.c /
RUN dnf -y install gcc
RUN gcc -o hello-world hello-world.c
RUN mkdir /sysroot
RUN mv hello-world /sysroot/
RUN dnf --releasever=$(rpm -q system-release --qf '%{VERSION}') \
  --installroot /sysroot \
  --use-host-config \
  -y \
  --setopt=install_weak_deps=False \
  install glibc \
  && dnf --installroot /sysroot clean all

FROM scratch
COPY --from=build /sysroot /
WORKDIR /
ENTRYPOINT ["/hello-world"]
```

   The compiler remains in the build stage. DNF5 installs `glibc` and its complete dependency closure into `/sysroot` for the final image. The dependency closure can contain packages in addition to the package that the `install` command names.

4. Build the container image.

```
$ docker build -t al2027-barebones-c-example .
```

5. Run the application.

```
$ docker run --rm al2027-barebones-c-example
Hello World!
```

Comparing packages installed on Amazon Linux 2027 Container Images

A comparison of the RPMs present on the AL2027 base container image compared with the RPMs present on the AL2027 minimal container image.

Package	Container	Minimal Container
alternatives	1.33	1.33
amazon-linux-repo-cdn	2027.0.20260903	2027.0.20260903
bash	5.3.0	5.3.0
bzip2-libs	1.0.8	1.0.8
ca-certificates	2025.2.80_v9.0.305	2025.2.80_v9.0.305
coreutils-single	9.10	9.10
crypto-policies	20260525	20260525
curl	8.18.0	8.18.0
dnf5	5.4.2.1	5.4.2.1
filesystem	3.18	3.18
findutils	4.10.0	4.10.0
fmt	11.2.0	11.2.0

Package	Container	Minimal Container
gawk	5.3.2	
glib2	2.88.1	2.88.1
glibc	2.44	2.44
glibc-common	2.44	2.44
glibc-minimal-langpack	2.44	2.44
gmp	6.3.0	
grep	3.12	3.12
json-c	0.18	0.18
keyutils-libs	1.6.3	1.6.3
krb5-libs	1.22.2	1.22.2
libacl	2.4.0	2.4.0
libarchive	3.8.8	3.8.8
libattr	2.5.2	2.5.2
libblkid	2.41.5	2.41.5
libcap	2.78	2.78
libcom_err	1.47.3	1.47.3
libcurl-minimal	8.18.0	8.18.0
libdnf5	5.4.2.1	5.4.2.1
libdnf5-cli	5.4.2.1	5.4.2.1
libffi	3.5.2	3.5.2

Package	Container	Minimal Container
libgcc	16.1.1	16.1.1
libidn2	2.3.8	2.3.8
libmount	2.41.5	2.41.5
libnghttp2	1.68.0	1.68.0
libpsl	0.21.5	0.21.5
librepo	1.20.0	1.20.0
libselinux	3.10	3.10
libsepol	3.10	3.10
libsmartcols	2.41.5	2.41.5
libsolv	0.7.39	0.7.39
libstdc++	16.1.1	16.1.1
libtasn1	4.20.0	4.20.0
libunistring	1.1	1.1
libuuid	2.41.5	2.41.5
libverto	0.3.2	0.3.2
libxml2	2.12.10	2.12.10
libzstd	1.5.7	1.5.7
lua-libs	5.4.8	5.4.8
lz4-libs	1.10.0	1.10.0
mpfr	4.2.2	

Package	Container	Minimal Container
ncurses-base	6.6	6.6
ncurses-libs	6.6	6.6
openssl-fips-provider-latest	3.5.7	3.5.7
openssl-libs	3.5.7	3.5.7
p11-kit	0.26.2	0.26.2
p11-kit-trust	0.26.2	0.26.2
pcre2	10.47	10.47
pcre2-syntax	10.47	10.47
popt	1.19	1.19
publicsuffix-list-dafsa	20260116	20260116
readline	8.3	
rpm	6.0.0	6.0.0
rpm-libs	6.0.0	6.0.0
rpm-sequoia	1.10.2.1	1.10.2.1
sdbus-cpp	2.2.1	2.2.1
sed	4.9	4.9
setup	2.15.1	2.15.1
sqlite-libs	3.51.2	3.51.2
system-release	2027.0.20260903	2027.0.20260903

Package	Container	Minimal Container
systemd-libs	260.1	260.1
systemd-standalone-sysusers	260.1	260.1
tar	1.35	
xz-libs	5.8.1	5.8.1
zlib-ng-compat	2.3.3	2.3.3

Comparing packages installed on Amazon Linux 2027 Minimal AMI and Container Images

A comparison of the RPMs present on the AL2027 Minimal AMI to the RPMs present on the AL2027 base and minimal container images.

Package	Minimal AMI	Container	Minimal Container
alternatives	1.33	1.33	1.33
amazon-chrony-config	4.8		
amazon-ec2-net-utils	2.7.6		
amazon-linux-repo-cdn		2027.0.20260903	2027.0.20260903
amazon-linux-repo-s3	2027.0.20260903		
amazon-linux-sb-keys	2027.1		

Package	Minimal AMI	Container	Minimal Container
audit	4.1.3		
audit-libs	4.1.3		
audit-rules	4.1.3		
authselect	1.7.1		
authselect-libs	1.7.1		
aws-lc-libs	5.2.0		
awscli-2	2.36.2		
bash	5.3.0	5.3.0	5.3.0
bzip2-libs	1.0.8	1.0.8	1.0.8
ca-certificates	2025.2.80_v9.0.305	2025.2.80_v9.0.305	2025.2.80_v9.0.305
checkpolicy	3.10		
chrony	4.8		
cloud-init	26.1		
cloud-init-cfg-ec2	26.1		
cloud-utils-growpart	0.33		
coreutils	9.10		
coreutils-common	9.10		
coreutils-single		9.10	9.10

Package	Minimal AMI	Container	Minimal Container
cpio	2.15		
cracklib	2.10.3		
crypto-policies	20260525	20260525	20260525
cryptsetup-libs	2.8.4		
curl	8.18.0	8.18.0	8.18.0
cyrus-sasl-lib	2.1.28		
dbus	1.16.0		
dbus-broker	37		
dbus-common	1.16.0		
dbus-libs	1.16.0		
device-mapper	1.02.212		
device-mapper-libs	1.02.212		
diffutils	3.12		
dnf-plugin-release-notification	2.0.0		
dnf-plugin-support-info	3.0.0		
dnf5	5.4.2.1	5.4.2.1	5.4.2.1
dracut	108		

Package	Minimal AMI	Container	Minimal Container
dracut-config-ec2	3.1		
dracut-config-generic	108		
e2fsprogs	1.47.3		
e2fsprogs-libs	1.47.3		
ec2-instance-connect	1.1		
ec2-instance-connect-selinux	1.1		
ec2-utils	2.2.0		
efi-filesystem	6		
efivar	39		
efivar-libs	39		
elfutils-libelf	0.195		
expat	2.8.1		
file	5.46		
file-libs	5.46		
filesystem	3.18	3.18	3.18
findutils	4.10.0	4.10.0	4.10.0
fmt	11.2.0	11.2.0	11.2.0

Package	Minimal AMI	Container	Minimal Container
fuse3-libs	3.18.2		
gawk	5.3.2	5.3.2	
gdbm	1.23		
gdbm-libs	1.23		
gdisk	1.0.10		
gettext	1.0		
gettext-envsubst	1.0		
gettext-libs	1.0		
gettext-runtime	1.0		
glib2	2.88.1	2.88.1	2.88.1
glibc	2.44	2.44	2.44
glibc-common	2.44	2.44	2.44
glibc-minimal-langpack	2.44	2.44	2.44
gmp	6.3.0	6.3.0	
gnulib-l10n	20241231		
gnutls	3.8.10		
grep	3.12	3.12	3.12
grub2-common	2.12		

Package	Minimal AMI	Container	Minimal Container
grub2-efi-aa64-ec2	2.12		
grub2-tools	2.12		
grub2-tools-minimal	2.12		
grubby	8.40		
gzip	1.14		
hostname	3.25		
hwdata	0.409		
inih	62		
iproute	6.17.0		
iputils	20250605		
irqbalance	1.9.5		
jq	1.8.1		
json-c	0.18	0.18	0.18
kbd	2.10.0		
kbd-legacy	2.10.0		
kbd-misc	2.10.0		
kernel7.1	7.1.0		
keyutils-libs	1.6.3	1.6.3	1.6.3
kmod	34.2		

Package	Minimal AMI	Container	Minimal Container
kmod-libs	34.2		
krb5-libs	1.22.2	1.22.2	1.22.2
less	702		
libacl	2.4.0	2.4.0	2.4.0
libarchive	3.8.8	3.8.8	3.8.8
libattr	2.5.2	2.5.2	2.5.2
libblkid	2.41.5	2.41.5	2.41.5
libbpf	1.6.3		
libcap	2.78	2.78	2.78
libcap-ng	0.9.3		
libcbor	0.13.0		
libcom_err	1.47.3	1.47.3	1.47.3
libcurl-minimal	8.18.0	8.18.0	8.18.0
libdnf5	5.4.2.1	5.4.2.1	5.4.2.1
libdnf5-cli	5.4.2.1	5.4.2.1	5.4.2.1
libeconf	0.7.9		
libedit	3.1		
libevent	2.1.12		
libfdisk	2.41.5		
libffi	3.5.2	3.5.2	3.5.2

Package	Minimal AMI	Container	Minimal Container
libfido2	1.16.0		
libgcc	16.1.1	16.1.1	16.1.1
libgomp	16.1.1		
libibverbs	61.0		
libidn2	2.3.8	2.3.8	2.3.8
libkcapi	1.5.0		
libkcapi-hasher	1.5.0		
libkcapi-hmaccalc	1.5.0		
liblastlog2	2.41.5		
libmnl	1.0.5		
libmount	2.41.5	2.41.5	2.41.5
libnghttp2	1.68.0	1.68.0	1.68.0
libnl3	3.12.0		
libpcap	1.10.6		
libpkgconf	2.5.1		
libpsl	0.21.5	0.21.5	0.21.5
libpwquality	1.4.5		
librepo	1.20.0	1.20.0	1.20.0
libseccomp	2.6.1		
libselinux	3.10	3.10	3.10

Package	Minimal AMI	Container	Minimal Container
libselinux-utils	3.10		
libsemanage	3.10		
libsepol	3.10	3.10	3.10
libsmartcols	2.41.5	2.41.5	2.41.5
libsolv	0.7.39	0.7.39	0.7.39
libss	1.47.3		
libstdc++	16.1.1	16.1.1	16.1.1
libsupportinfo	2.0.0		
libtasn1	4.20.0	4.20.0	4.20.0
libtextstyle	1.0		
libtool-ltdl	2.5.4		
libunistring	1.1	1.1	1.1
libuuid	2.41.5	2.41.5	2.41.5
libverto	0.3.2	0.3.2	0.3.2
libxcrypt	4.5.2		
libxkbcommon	1.13.1		
libxml2	2.12.10	2.12.10	2.12.10
libyaml	0.2.5		
libzstd	1.5.7	1.5.7	1.5.7
logrotate	3.22.0		

Package	Minimal AMI	Container	Minimal Container
lua-libs	5.4.8	5.4.8	5.4.8
lz4-libs	1.10.0	1.10.0	1.10.0
mpdecimal	4.0.1		
mpfr	4.2.2	4.2.2	
ncurses	6.6		
ncurses-base	6.6	6.6	6.6
ncurses-libs	6.6	6.6	6.6
nettle	3.10.1		
nmap-ncat	7.93		
numactl-libs	2.0.19		
oniguruma	6.9.10		
openldap	2.6.13		
openssh	9.9p1		
openssh-clients	9.9p1		
openssh-server	9.9p1		
openssl	3.5.7		
openssl-fips-provider-latest	3.5.7	3.5.7	3.5.7
openssl-libs	3.5.7	3.5.7	3.5.7
os-prober	1.81		

Package	Minimal AMI	Container	Minimal Container
p11-kit	0.26.2	0.26.2	0.26.2
p11-kit-trust	0.26.2	0.26.2	0.26.2
pam	1.7.1		
pam-libs	1.7.1		
pciutils	3.15.0		
pciutils-libs	3.15.0		
pcre2	10.47	10.47	10.47
pcre2-syntax	10.47	10.47	10.47
pkgconf	2.5.1		
pkgconf-m4	2.5.1		
pkgconf-pkg-config	2.5.1		
policycoreutils	3.10		
policycoreutils-python-utils	3.10		
popt	1.19	1.19	1.19
procps-ng	4.0.6		
psmisc	23.7		
publicsuffix-list-dafsa	20260116	20260116	20260116
python3	3.14.7		

Package	Minimal AMI	Container	Minimal Container
python3-attrs	25.4.0		
python3-audit	4.1.3		
python3-awscrt	0.36.0		
python3-cffi	2.0.0		
python3-charset-normalizer	3.4.4		
python3-colorama	0.4.6		
python3-configobj	5.0.9		
python3-cryptography	49.0.0		
python3-dateutil	2.9.0.post0		
python3-distro	1.9.0		
python3-docutils	0.21.2		
python3-idna	3.11		
python3-jinja2	3.1.6		
python3-jmespath	1.0.1		
python3-jsonpatch	1.33		

Package	Minimal AMI	Container	Minimal Container
python3-jsonpointer	2.4		
python3-jsonschema	4.23.0		
python3-jsonschema-specifications	2024.10.1		
python3-libs	3.14.7		
python3-libselinux	3.10		
python3-libsemanage	3.10		
python3-markupsafe	3.0.2		
python3-oauthlib	3.3.1		
python3-pip-wheel	26.2.1		
python3-ply	3.11		
python3-policycoreutils	3.10		
python3-prompt-toolkit	3.0.41		
python3-pycparser	2.22		

Package	Minimal AMI	Container	Minimal Container
python3-pyyaml	6.0.3		
python3-referencing	0.36.2		
python3-requests	2.33.1		
python3-rpds-py	0.27.0		
python3-ruamel-yaml	0.19.1		
python3-ruamel-yaml+oldlibyaml	0.19.1		
python3-ruamel-yaml-clib	0.2.15		
python3-setools	4.6.0		
python3-six	1.17.0		
python3-urllib3	2.6.3		
python3-wcwidth	0.6.0		
rdma-core-common	61.0		
readline	8.3	8.3	
rootfiles	9.0		
rpm	6.0.0	6.0.0	6.0.0
rpm-libs	6.0.0	6.0.0	6.0.0

Package	Minimal AMI	Container	Minimal Container
rpm-plugin-selinux	6.0.0		
rpm-plugin-systemd-inhibit	6.0.0		
rpm-sequoia	1.10.2.1	1.10.2.1	1.10.2.1
sbsigntools	0.9.5		
sdbus-cpp	2.2.1	2.2.1	2.2.1
sed	4.9	4.9	4.9
selinux-policy	44.5		
selinux-policy-targeted	44.5		
setup	2.15.1	2.15.1	2.15.1
shadow-utils	4.19.0		
sqlite-libs	3.51.2	3.51.2	3.51.2
sudo	1.9.17		
sysctl-defaults	1.0		
system-release	2027.0.20260903	2027.0.20260903	2027.0.20260903
systemd	260.1		
systemd-libs	260.1	260.1	260.1
systemd-networkd	260.1		

Package	Minimal AMI	Container	Minimal Container
systemd-resolved	260.1		
systemd-shared	260.1		
systemd-standalone-sysusers		260.1	260.1
systemd-sysusers	260.1		
systemd-udev	260.1		
tar	1.35	1.35	
tzdata	2026c		
update-motd	2.3		
userspace-rcu	0.15.6		
util-linux	2.41.5		
util-linux-core	2.41.5		
vim-data	9.2.920		
vim-minimal	9.2.920		
which	2.25		
xfsprogs	7.1.1		
xkeyboard-config	2.47		
xz	5.8.1		

Package	Minimal AMI	Container	Minimal Container
xz-libs	5.8.1	5.8.1	5.8.1
zlib-ng-compat	2.3.3	2.3.3	2.3.3
zram-generator	1.2.1		
zram-generator-defaults	1.2.1		
zstd	1.5.7		

Using Amazon Elastic File System on AL2027

AL2027 Preview

AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

With Amazon Elastic File System (Amazon EFS), you can share file data using serverless, fully elastic storage. You don't need to provision or manage storage capacity and performance. Amazon EFS scales on demand to petabytes without disrupting applications, growing and shrinking automatically as you add and remove files. Because Amazon EFS has a simple web services interface, you can create and configure file systems quickly and easily.

With Amazon EFS, you can use the Network File System version 4 (NFSv4.1 and NFSv4.0) protocol. The applications and tools that you already use work seamlessly with Amazon EFS. Multiple compute instances, including Amazon EC2, Amazon ECS, and AWS Lambda, can access an Amazon EFS file system at the same time. Therefore, an Amazon EFS file system can provide a common data source for workloads and applications. These workloads can run on more than one compute instance or server.

Installing `amazon-efs-utils` on AL2027

The `amazon-efs-utils` package is available in the AL2027 repositories.

Install the `amazon-efs-utils` package on AL2027

- Install `amazon-efs-utils` using the following command.

```
$ dnf -y install amazon-efs-utils
```

Mounting an Amazon EFS file system on AL2027

After you install `amazon-efs-utils`, you can mount an Amazon EFS file system on your AL2027 instance.

Mount an Amazon EFS file system on AL2027

- To mount using the file system ID, use the following command.

```
sudo mount -t efs file-system-id efs-mount-point/
```

You can also mount the file system with TLS to encrypt data in transit. Alternatively, use the DNS name or mount target IP instead of the file system ID. For more information, see Mounting on Amazon Linux instances using the EFS mount helper.

Identifying Amazon Linux instances and versions

AL2027 Preview

AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

It can be important to be able to determine what Linux distribution, and what version of that distribution an OS image or instance is. Amazon Linux provides mechanisms to identify Amazon Linux apart from other Linux distributions, as well as to identify what release of Amazon Linux the image is.

This section covers the different methods that can be used, their limitations, and provides examples of their use.

Topics

- Using the os-release standard
- Amazon Linux-specific identification
- Example code for OS detection

Using the `os-release` standard

Amazon Linux complies with the `os-release` standard on the freedesktop.org website for identifying Linux distributions. This file provides machine-readable information about the operating system identification and version.

Note

The standard dictates that `/etc/os-release` is parsed first, followed by `/usr/lib/os-release`. Take care to follow the standard around file names and paths.

Topics

- Key identification differences

- Field types: machine-readable and human-readable
- /etc/os-release examples
- Comparison with other distributions

Key identification differences

The `os-release` is found at `/etc/os-release`, and if that is not present, at `/usr/lib/os-release`. Consult the `os-release` standard for complete information.

The most reliable way to determine that an instance is running Amazon Linux is to check the `ID` field in `os-release`. On every version of Amazon Linux, the `ID` field is `amzn`.

The most reliable way to distinguish between Amazon Linux versions is to check the `VERSION_ID` field in `os-release`:

- AL2027: `VERSION_ID="2027"`
- AL2023: `VERSION_ID="2023"`
- AL2: `VERSION_ID="2"`

> **Note**
>
> Remember that `VERSION_ID` is a machine-readable field intended for programmatic use, while `PRETTY_NAME` is designed for display to users. See the section called “Field types” for more information about field types.

Field types: machine-readable and human-readable

The `/etc/os-release` file (or `/usr/lib/os-release` if `/etc/os-release` does not exist) contains two types of fields: machine-readable fields intended for programmatic use, and human-readable fields intended for presentation to users.

Machine-readable fields

These fields use standardized formats and are intended for processing by scripts, package managers, and other automated tools. They contain only lowercase letters, numbers, and limited punctuation (periods, underscores, and hyphens).

- `ID` – Operating system identifier. Amazon Linux uses `amzn` across all versions, distinguishing it from other distributions like Debian (`debian`), Ubuntu (`ubuntu`), or Fedora (`fedora`)
- `VERSION_ID` – Operating system version for programmatic use, such as `2027`
- `ID_LIKE` – Space-separated list of related distributions, such as `fedora`
- `VERSION_CODENAME` – Release codename for scripts
- `PLATFORM_ID` – Platform identifier, such as `platform:al2027`
- `VARIANT_ID` – Variant identifier for programmatic decisions
- `BUILD_ID` – Build identifier for system images
- `IMAGE_ID` – Image identifier for containerized environments

Human-readable fields

These fields are intended for display to users and can contain spaces, mixed case, and descriptive text. Use them when presenting operating system information in user interfaces.

- `NAME` – Operating system name for display, such as `Amazon Linux`
- `PRETTY_NAME` – Full operating system name with version for display, such as `Amazon Linux 2027.0.20260903`
- `VERSION` – Version information suitable for user presentation
- `VARIANT` – Variant or edition name for display, such as `Server Edition`

Other informational fields

These fields provide additional metadata about the operating system:

- `HOME_URL`, `DOCUMENTATION_URL`, `SUPPORT_URL`, `BUG_REPORT_URL` – Project, documentation, support, and bug reporting URLs
- `VENDOR_NAME`, `VENDOR_URL` – Vendor name and URL
- `SUPPORT_END` – End-of-support date in YYYY-MM-DD format
- `CPE_NAME` – Common Platform Enumeration identifier
- `ANSI_COLOR` – ANSI color code for terminal display

When writing scripts or applications that need to identify Amazon Linux programmatically, use the machine-readable fields like `ID` and `VERSION_ID`. When displaying operating system information to users, use the human-readable fields like `PRETTY_NAME`.

`/etc/os-release` examples

The `/etc/os-release` file content varies between Amazon Linux versions:

AL2027

```
[ec2-user ~]$ cat /etc/os-release
```

```
NAME="Amazon Linux"
VERSION="2027"
ID="amzn"
ID_LIKE="fedora"
VERSION_ID="2027"
VARIANT="Public Preview"
VARIANT_ID="preview"
PLATFORM_ID="platform:al2027"
PRETTY_NAME="Amazon Linux 2027.0.20260903"
ANSI_COLOR="0;33"
CPE_NAME="cpe:2.3:o:amazon:amazon_linux:2027:2027.0.20260903"
HOME_URL="https://aws.amazon.com/linux/"
DOCUMENTATION_URL="https://docs.aws.amazon.com/linux/"
SUPPORT_URL="https://aws.amazon.com/premiumsupport/"
BUG_REPORT_URL="https://github.com/amazonlinux/amazon-linux-2027"
VENDOR_NAME="AWS"
VENDOR_URL="https://aws.amazon.com/"
SUPPORT_END="2026-12-31"
```

AL2023

```
[ec2-user ~]$ cat /etc/os-release
```

```
NAME="Amazon Linux"
VERSION="2023"
ID="amzn"
ID_LIKE="fedora"
VERSION_ID="2023"
```

```
PLATFORM_ID="platform:al2023"
PRETTY_NAME="Amazon Linux 2023.12.20260831"
ANSI_COLOR="0;33"
CPE_NAME="cpe:2.3:o:amazon:amazon_linux:2023"
HOME_URL="https://aws.amazon.com/linux/amazon-linux-2023/"
DOCUMENTATION_URL="https://docs.aws.amazon.com/linux/"
SUPPORT_URL="https://aws.amazon.com/premiumsupport/"
BUG_REPORT_URL="https://github.com/amazonlinux/amazon-linux-2023"
VENDOR_NAME="AWS"
VENDOR_URL="https://aws.amazon.com/"
SUPPORT_END="2029-06-30"
```

AL2

```
[ec2-user ~]$ cat /etc/os-release
```

```
NAME="Amazon Linux"
VERSION="2"
ID="amzn"
ID_LIKE="centos rhel fedora"
VERSION_ID="2"
PRETTY_NAME="Amazon Linux 2"
ANSI_COLOR="0;33"
CPE_NAME="cpe:2.3:o:amazon:amazon_linux:2"
HOME_URL="https://amazonlinux.com/"
SUPPORT_END="2026-06-30"
```

Amazon Linux AMI

```
[ec2-user ~]$ cat /etc/os-release
```

```
NAME="Amazon Linux AMI"
VERSION="2018.03"
ID="amzn"
ID_LIKE="rhel fedora"
VERSION_ID="2018.03"
PRETTY_NAME="Amazon Linux AMI 2018.03"
ANSI_COLOR="0;33"
CPE_NAME="cpe:/o:amazon:linux:2018.03:ga"
HOME_URL="http://aws.amazon.com/amazon-linux-ami/"
```

Comparison with other distributions

To understand how Amazon Linux fits within the broader Linux ecosystem, compare its `/etc/os-release` format with other major distributions:

Fedora

```
[ec2-user ~]$ cat /etc/os-release
```

```
NAME="Fedora Linux"
VERSION="44 (Container Image)"
RELEASE_TYPE=stable
ID=fedora
VERSION_ID=44
VERSION_CODENAME=""
PRETTY_NAME="Fedora Linux 44 (Container Image)"
ANSI_COLOR="0;38;2;60;110;180"
LOGO=fedora-logo-icon
CPE_NAME="cpe:/o:fedoraproject:fedora:44"
DEFAULT_HOSTNAME="fedora"
HOME_URL="https://fedoraproject.org/"
DOCUMENTATION_URL="https://docs.fedoraproject.org/en-US/fedora/f44/"
SUPPORT_URL="https://ask.fedoraproject.org/"
BUG_REPORT_URL="https://bugzilla.redhat.com/"
REDHAT_BUGZILLA_PRODUCT="Fedora"
REDHAT_BUGZILLA_PRODUCT_VERSION=44
REDHAT_SUPPORT_PRODUCT="Fedora"
REDHAT_SUPPORT_PRODUCT_VERSION=44
SUPPORT_END=2027-05-19
VARIANT="Container Image"
VARIANT_ID=container
```

Debian

```
[ec2-user ~]$ cat /etc/os-release
```

```
PRETTY_NAME="Debian GNU/Linux 13 (trixie)"
NAME="Debian GNU/Linux"
VERSION_ID="13"
VERSION="13 (trixie)"
VERSION_CODENAME=trixie
```

```
DEBIAN_VERSION_FULL=13.6
ID=debian
HOME_URL="https://www.debian.org/"
SUPPORT_URL="https://www.debian.org/support"
BUG_REPORT_URL="https://bugs.debian.org/"
```

Ubuntu

```
[ec2-user ~]$ cat /etc/os-release
```

```
PRETTY_NAME="Ubuntu 26.04 LTS"
NAME="Ubuntu"
VERSION_ID="26.04"
VERSION="26.04 LTS (Resolute Raccoon)"
VERSION_CODENAME=resolute
ID=ubuntu
ID_LIKE=debian
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
UBUNTU_CODENAME=resolute
LOGO=ubuntu-logo
```

Notice how the machine-readable fields provide consistent identification across distributions:

- `ID` – Uniquely identifies the operating system: `amzn` for Amazon Linux, `fedora` for Fedora, `debian` for Debian, `ubuntu` for Ubuntu
- `ID_LIKE` – Shows distribution relationships: AL2027 and AL2023 use `fedora`, AL2 uses `centos rhel fedora`, and Ubuntu shows `debian` to indicate its Debian heritage
- `VERSION_ID` – Provides machine-parseable version information: `2027` for AL2027, `2023` for AL2023, `44` for Fedora, `13` for Debian, `26.04` for Ubuntu

In contrast, the human-readable fields are designed for display to users:

- `NAME` – User-friendly OS name: `Amazon Linux`, `Fedora Linux`, `Debian GNU/Linux`, `Ubuntu`

- PRETTY_NAME – Complete display name with version: `Amazon Linux 2027.0.20260903`, `Amazon Linux 2023.12.20260831`, `Fedora Linux 44 (Container Image)`, `Debian GNU/Linux 13 (trixie)`, `Ubuntu 26.04 LTS`
- VERSION – Human-readable version with additional context like codenames or release types

When writing cross-platform scripts, always use the machine-readable fields (ID, VERSION_ID, ID_LIKE) for logic and decisions, and use the human-readable fields (PRETTY_NAME, NAME) only for displaying information to users.

Amazon Linux-specific identification

Some files are specific to Amazon Linux and can identify Amazon Linux and its version. For new code, use the `/etc/os-release` standard for cross-distribution compatibility, and avoid the Amazon Linux-specific files.

Topics

- The /etc/system-release file
- Image identification file
- Examples of Amazon Linux-specific files

The `/etc/system-release` file

Amazon Linux contains an `/etc/system-release` file that specifies the current release that is installed. This file is updated through package managers, and on Amazon Linux it is part of the `system-release` package. Although some other distributions like Fedora also have this file, it is not present in Debian-based distributions like Ubuntu.

> **Note**
>
> The `/etc/system-release` file contains a human-readable string and should not be used programmatically to identify an OS or release. Use the machine-readable fields in `/etc/os-release` (or `/usr/lib/os-release` if `/etc/os-release` does not exist) instead.

Amazon Linux also contains a machine-readable version of `/etc/system-release` that follows the Common Platform Enumeration (CPE) specification in the `/etc/system-release-cpe` file.

On AL2027, the `/etc/system-release` and `/etc/system-release-cpe` files are symbolic links to files under `/usr/lib`. The `/etc/amazon-linux-release` and `/etc/amazon-linux-release-cpe` links point to the same files. The presence of either link indicates an Amazon Linux system.

Image identification file

Each Amazon Linux image contains a unique `/etc/image-id` file that provides additional information about the original image as generated by the Amazon Linux team. This file is specific to Amazon Linux and is not found in other Linux distributions such as Debian, Ubuntu, or Fedora. This file contains the following information about the image:

- `image_name`, `image_version`, `image_arch` – Values from the build recipe that was used to construct the image.
- `image_stamp` – A unique, random hex value generated during image creation.
- `image_date` – The UTC time of image creation, in *YYYYMMDDhhmmss* format.
- `recipe_name`, `recipe_id` – The name and ID of the build recipe used to construct the image.

Examples of Amazon Linux-specific files

The following sections provide examples of the Amazon Linux-specific identification files for each major version of Amazon Linux.

> **Note**
>
> In any real-world code, `/usr/lib/os-release` should be used if the `/etc/os-release` file does not exist.

AL2027

The following examples show the identification files for an AL2027 container image.

Example of `/etc/image-id` for AL2027:

```
[ec2-user ~]$ cat /etc/image-id
```

```
image_name="al2027-preview-container"
```

```
image_version="2027"
image_arch="aarch64"
image_file="al2027-preview-container-2027.0.20260903.0-arm64"
image_stamp="c6c3-7b9f"
image_date="20260901070223"
recipe_name="al2027-preview container"
recipe_id="ce25d616-1f24-28ca-9b6b-b436-69ee-3b04-9101499b"
```

Example of `/etc/system-release` for AL2027:

```
[ec2-user ~]$ cat /etc/system-release
```

```
Amazon Linux release 2027.0.20260903 (Amazon Linux)
```

AL2023

The following examples show the identification files for AL2023.

Example of `/etc/image-id` for AL2023:

```
[ec2-user ~]$ cat /etc/image-id
```

```
image_name="al2023-container"
image_version="2023"
image_arch="aarch64"
image_file="al2023-container-2023.12.20260831.0-arm64"
image_stamp="3edb-dbbb"
image_date="20260826031129"
recipe_name="al2023 container"
recipe_id="697759ad-93a1-fbd8-433c-218f-db48-0369-d107f5a1"
```

Example of `/etc/system-release` for AL2023:

```
[ec2-user ~]$ cat /etc/system-release
```

```
Amazon Linux release 2023.12.20260831 (Amazon Linux)
```

AL2

The following examples show the identification files for AL2.

Example of `/etc/image-id` for AL2:

```
[ec2-user ~]$ cat /etc/image-id
```

```
image_name="amzn2-container-raw"
image_version="2"
image_arch="aarch64"
image_file="amzn2-container-raw-2.0.20260826.0-arm64"
image_stamp="60a9-e332"
image_date="20260826163727"
recipe_name="amzn2 container"
recipe_id="33b97c11-a4f7-3b8c-b51e-583d-f0e1-34af-12f98d79"
```

Example of `/etc/system-release` for AL2:

```
[ec2-user ~]$ cat /etc/system-release
```

```
Amazon Linux release 2 (Karoo)
```

Amazon Linux AMI

The following examples show the identification files for Amazon Linux AMI.

Example of `/etc/image-id` for Amazon Linux AMI:

```
[ec2-user ~]$ cat /etc/image-id
```

```
image_name="amzn-container-minimal"
image_version="2018.03"
image_arch="x86_64"
image_file="amzn-container-minimal-2018.03.0.20231218.0-x86_64"
image_stamp="407d-5ef3"
image_date="20231218203210"
recipe_name="amzn container"
recipe_id="b1e7635e-14e3-dd57-b1ab-7351-edd0-d9e0-ca6852ea"
```

Example of `/etc/system-release` for Amazon Linux AMI:

```
[ec2-user ~]$ cat /etc/system-release
```

```
Amazon Linux AMI release 2018.03
```

Example code for OS detection

The following examples demonstrate how to programmatically detect the operating system and version using the `/etc/os-release` (or `/usr/lib/os-release` if `/etc/os-release` does not exist) file. These examples show how to distinguish between Amazon Linux and other distributions, as well as how to use the `ID_LIKE` field to determine distribution families.

The following script is implemented in several different programming languages, and each implementation produces the same output.

Shell

```
#!/bin/bash

# Function to get a specific field from os-release file
get_os_release_field() {
    local field="$1"
    local os_release_file

    # Find the os-release file
    if [ -f /etc/os-release ]; then
        os_release_file='/etc/os-release'
    elif [ -f /usr/lib/os-release ]; then
        os_release_file='/usr/lib/os-release'
    else
        echo "Error: os-release file not found" >&2
        return 1
    fi

    # Source the file in a subshell and return the requested field.
    #
    # A subshell means that variables from os-release are only available
    # within the subshell, and the main script environment remains clean.
    (
        . "$os_release_file"
        eval "echo \"\$$field\""
    )
}
```

```
is_amazon_linux() {
    [ "$(get_os_release_field ID)" = "amzn" ]
}

is_fedora() {
    [ "$(get_os_release_field ID)" = "fedora" ]
}

is_ubuntu() {
    [ "$(get_os_release_field ID)" = "ubuntu" ]
}

is_debian() {
    [ "$(get_os_release_field ID)" = "debian" ]
}

# Function to check if this is like Fedora (includes Amazon Linux, CentOS, RHEL,
 etc.)
is_like_fedora() {
    local id="$(get_os_release_field ID)"
    local id_like="$(get_os_release_field ID_LIKE)"
    [ "$id" = "fedora" ] || [[ "$id_like" == *"fedora"* ]]
}

# Function to check if this is like Debian (includes Ubuntu and derivatives)
is_like_debian() {
    local id="$(get_os_release_field ID)"
    local id_like="$(get_os_release_field ID_LIKE)"
    [ "$id" = "debian" ] || [[ "$id_like" == *"debian"* ]]
}

# Get the main fields we'll use multiple times
ID="$(get_os_release_field ID)"
VERSION_ID="$(get_os_release_field VERSION_ID)"
PRETTY_NAME="$(get_os_release_field PRETTY_NAME)"
ID_LIKE="$(get_os_release_field ID_LIKE)"

echo "Operating System Detection Results:"
echo "==================================="
echo "Is Amazon Linux: $(is_amazon_linux && echo YES || echo NO)"
echo "Is Fedora: $(is_fedora && echo YES || echo NO)"
echo "Is Ubuntu: $(is_ubuntu && echo YES || echo NO)"
echo "Is Debian: $(is_debian && echo YES || echo NO)"
echo "Is like Fedora: $(is_like_fedora && echo YES || echo NO)"
```

```
echo "Is like Debian: $(is_like_debian && echo YES || echo NO)"
echo
echo "Detailed OS Information:"
echo "======================="
echo "ID: $ID"
echo "VERSION_ID: $VERSION_ID"
echo "PRETTY_NAME: $PRETTY_NAME"
[ -n "$ID_LIKE" ] && echo "ID_LIKE: $ID_LIKE"

# Amazon Linux specific information
if is_amazon_linux; then
    echo ""
    echo "Amazon Linux Version Details:"
    echo "============================="
    case "$VERSION_ID" in
        2018.03)
            echo "Amazon Linux AMI (version 1)"
            ;;
        2)
            echo "Amazon Linux 2"
            ;;
        2023)
            echo "Amazon Linux 2023"
            ;;
        2027)
            echo "Amazon Linux 2027"
            ;;
        *)
            echo "Unknown Amazon Linux version: $VERSION_ID"
            ;;
    esac

    # Check for Amazon Linux specific files
    [ -f /etc/image-id ] && echo "Amazon Linux image-id file present"
fi
```

Python 3.7-3.9

```
#!/usr/bin/env python3

import os
import sys
```

```
def parse_os_release():
    """Parse the os-release file and return a dictionary of key-value pairs."""
    os_release_data = {}

    # Try /etc/os-release first, then /usr/lib/os-release
    for path in ['/etc/os-release', '/usr/lib/os-release']:
        if os.path.exists(path):
            try:
                with open(path, 'r') as f:
                    for line in f:
                        line = line.strip()
                        if line and not line.startswith('#') and '=' in line:
                            key, value = line.split('=', 1)
                            # Remove quotes if present
                            value = value.strip('"\'')
                            os_release_data[key] = value
                return os_release_data
            except IOError:
                continue

    print("Error: os-release file not found")
    sys.exit(1)

def is_amazon_linux(os_data):
    """Check if this is Amazon Linux."""
    return os_data.get('ID') == 'amzn'

def is_fedora(os_data):
    """Check if this is Fedora."""
    return os_data.get('ID') == 'fedora'

def is_ubuntu(os_data):
    """Check if this is Ubuntu."""
    return os_data.get('ID') == 'ubuntu'

def is_debian(os_data):
    """Check if this is Debian."""
    return os_data.get('ID') == 'debian'

def is_like_fedora(os_data):
    """Check if this is like Fedora (includes Amazon Linux, CentOS, RHEL, etc.)."""
    if os_data.get('ID') == 'fedora':
        return True
    id_like = os_data.get('ID_LIKE', '')
```

```
    return 'fedora' in id_like

def is_like_debian(os_data):
    """Check if this is like Debian (includes Ubuntu and derivatives)."""
    if os_data.get('ID') == 'debian':
        return True
    id_like = os_data.get('ID_LIKE', '')
    return 'debian' in id_like

def main():
    # Parse os-release file
    os_data = parse_os_release()

    # Display results
    print("Operating System Detection Results:")
    print("==================================")
    print(f"Is Amazon Linux: {'YES' if is_amazon_linux(os_data) else 'NO'}")
    print(f"Is Fedora: {'YES' if is_fedora(os_data) else 'NO'}")
    print(f"Is Ubuntu: {'YES' if is_ubuntu(os_data) else 'NO'}")
    print(f"Is Debian: {'YES' if is_debian(os_data) else 'NO'}")
    print(f"Is like Fedora: {'YES' if is_like_fedora(os_data) else 'NO'}")
    print(f"Is like Debian: {'YES' if is_like_debian(os_data) else 'NO'}")

    # Additional information
    print()
    print("Detailed OS Information:")
    print("========================")
    print(f"ID: {os_data.get('ID', '')}")
    print(f"VERSION_ID: {os_data.get('VERSION_ID', '')}")
    print(f"PRETTY_NAME: {os_data.get('PRETTY_NAME', '')}")
    if os_data.get('ID_LIKE'):
        print(f"ID_LIKE: {os_data.get('ID_LIKE')}")

    # Amazon Linux specific information
    if is_amazon_linux(os_data):
        print()
        print("Amazon Linux Version Details:")
        print("=============================")
        version_id = os_data.get('VERSION_ID', '')
        if version_id == '2018.03':
            print("Amazon Linux AMI (version 1)")
        elif version_id == '2':
            print("Amazon Linux 2")
        elif version_id == '2023':
```

```
            print("Amazon Linux 2023")
        elif version_id == '2027':
            print("Amazon Linux 2027")
        else:
            print(f"Unknown Amazon Linux version: {version_id}")

        # Check for Amazon Linux specific files
        if os.path.exists('/etc/image-id'):
            print("Amazon Linux image-id file present")

if __name__ == '__main__':
    main()
```

Python 3.10+

```
#!/usr/bin/env python3

import os
import sys
import platform

def is_amazon_linux(os_data):
    """Check if this is Amazon Linux."""
    return os_data.get('ID') == 'amzn'

def is_fedora(os_data):
    """Check if this is Fedora."""
    return os_data.get('ID') == 'fedora'

def is_ubuntu(os_data):
    """Check if this is Ubuntu."""
    return os_data.get('ID') == 'ubuntu'

def is_debian(os_data):
    """Check if this is Debian."""
    return os_data.get('ID') == 'debian'

def is_like_fedora(os_data):
    """Check if this is like Fedora (includes Amazon Linux, CentOS, RHEL, etc.)."""
    if os_data.get('ID') == 'fedora':
        return True
    id_like = os_data.get('ID_LIKE', '')
    return 'fedora' in id_like
```

```
def is_like_debian(os_data):
    """Check if this is like Debian (includes Ubuntu and derivatives)."""
    if os_data.get('ID') == 'debian':
        return True
    id_like = os_data.get('ID_LIKE', '')
    return 'debian' in id_like

def main():
    # Parse os-release file using the standard library function
    try:
        os_data = platform.freedesktop_os_release()
    except OSError:
        print("Error: os-release file not found")
        sys.exit(1)

    # Display results
    print("Operating System Detection Results:")
    print("==================================")
    print(f"Is Amazon Linux: {'YES' if is_amazon_linux(os_data) else 'NO'}")
    print(f"Is Fedora: {'YES' if is_fedora(os_data) else 'NO'}")
    print(f"Is Ubuntu: {'YES' if is_ubuntu(os_data) else 'NO'}")
    print(f"Is Debian: {'YES' if is_debian(os_data) else 'NO'}")
    print(f"Is like Fedora: {'YES' if is_like_fedora(os_data) else 'NO'}")
    print(f"Is like Debian: {'YES' if is_like_debian(os_data) else 'NO'}")

    # Additional information
    print()
    print("Detailed OS Information:")
    print("========================")
    print(f"ID: {os_data.get('ID', '')}")
    print(f"VERSION_ID: {os_data.get('VERSION_ID', '')}")
    print(f"PRETTY_NAME: {os_data.get('PRETTY_NAME', '')}")
    if os_data.get('ID_LIKE'):
        print(f"ID_LIKE: {os_data.get('ID_LIKE')}")

    # Amazon Linux specific information
    if is_amazon_linux(os_data):
        print()
        print("Amazon Linux Version Details:")
        print("=============================")
        version_id = os_data.get('VERSION_ID', '')
        if version_id == '2018.03':
            print("Amazon Linux AMI (version 1)")
```

```
        elif version_id == '2':
            print("Amazon Linux 2")
        elif version_id == '2023':
            print("Amazon Linux 2023")
        elif version_id == '2027':
            print("Amazon Linux 2027")
        else:
            print(f"Unknown Amazon Linux version: {version_id}")

        # Check for Amazon Linux specific files
        if os.path.exists('/etc/image-id'):
            print("Amazon Linux image-id file present")

if __name__ == '__main__':
    main()
```

Perl

```
#!/usr/bin/env perl

use strict;
use warnings;

# Function to parse the os-release file and return a hash of key-value pairs
sub parse_os_release {
    my %os_release_data;

    # Try /etc/os-release first, then /usr/lib/os-release
    my @paths = ('/etc/os-release', '/usr/lib/os-release');

    for my $path (@paths) {
        if (-f $path) {
            if (open(my $fh, '<', $path)) {
                while (my $line = <$fh>) {
                    chomp $line;
                    next if $line =~ /^\s*$/ || $line =~ /^\s*#/;

                    if ($line =~ /^([^=]+)=(.*)$/) {
                        my ($key, $value) = ($1, $2);
                        # Remove quotes if present
                        $value =~ s/^["']|["']$//g;
                        $os_release_data{$key} = $value;
                    }
```

```
                }
                close($fh);
                return %os_release_data;
            }
        }
    }

    die "Error: os-release file not found\n";
}

# Function to check if this is Amazon Linux
sub is_amazon_linux {
    my %os_data = @_;
    return ($os_data{ID} // '') eq 'amzn';
}

# Function to check if this is Fedora
sub is_fedora {
    my %os_data = @_;
    return ($os_data{ID} // '') eq 'fedora';
}

# Function to check if this is Ubuntu
sub is_ubuntu {
    my %os_data = @_;
    return ($os_data{ID} // '') eq 'ubuntu';
}

# Function to check if this is Debian
sub is_debian {
    my %os_data = @_;
    return ($os_data{ID} // '') eq 'debian';
}

# Function to check if this is like Fedora (includes Amazon Linux, CentOS, RHEL,
 etc.)
sub is_like_fedora {
    my %os_data = @_;
    return 1 if ($os_data{ID} // '') eq 'fedora';
    my $id_like = $os_data{ID_LIKE} // '';
    return $id_like =~ /fedora/;
}

# Function to check if this is like Debian (includes Ubuntu and derivatives)
```

```
sub is_like_debian {
    my %os_data = @_;
    return 1 if ($os_data{ID} // '') eq 'debian';
    my $id_like = $os_data{ID_LIKE} // '';
    return $id_like =~ /debian/;
}

# Main execution
my %os_data = parse_os_release();

# Display results
print "Operating System Detection Results:\n";
print "==================================\n";
print "Is Amazon Linux: " . (is_amazon_linux(%os_data) ? "YES" : "NO") . "\n";
print "Is Fedora: " . (is_fedora(%os_data) ? "YES" : "NO") . "\n";
print "Is Ubuntu: " . (is_ubuntu(%os_data) ? "YES" : "NO") . "\n";
print "Is Debian: " . (is_debian(%os_data) ? "YES" : "NO") . "\n";
print "Is like Fedora: " . (is_like_fedora(%os_data) ? "YES" : "NO") . "\n";
print "Is like Debian: " . (is_like_debian(%os_data) ? "YES" : "NO") . "\n";
print "\n";

# Additional information
print "Detailed OS Information:\n";
print "========================\n";
print "ID: " . ($os_data{ID} // '') . "\n";
print "VERSION_ID: " . ($os_data{VERSION_ID} // '') . "\n";
print "PRETTY_NAME: " . ($os_data{PRETTY_NAME} // '') . "\n";
print "ID_LIKE: " . ($os_data{ID_LIKE} // '') . "\n" if $os_data{ID_LIKE};

# Amazon Linux specific information
if (is_amazon_linux(%os_data)) {
    print "\n";
    print "Amazon Linux Version Details:\n";
    print "=============================\n";
    my $version_id = $os_data{VERSION_ID} // '';

    if ($version_id eq '2018.03') {
        print "Amazon Linux AMI (version 1)\n";
    } elsif ($version_id eq '2') {
        print "Amazon Linux 2\n";
    } elsif ($version_id eq '2023') {
        print "Amazon Linux 2023\n";
    } elsif ($version_id eq '2027') {
        print "Amazon Linux 2027\n";
```

```
    } else {
        print "Unknown Amazon Linux version: $version_id\n";
    }

    # Check for Amazon Linux specific files
    if (-f '/etc/image-id') {
        print "Amazon Linux image-id file present\n";
    }
}
```

When run on different systems, the script produces the following output:

AL2023

```
Operating System Detection Results:
==================================
Is Amazon Linux: YES
Is Fedora: NO
Is Ubuntu: NO
Is Debian: NO
Is like Fedora: YES
Is like Debian: NO

Detailed OS Information:
=======================
ID: amzn
VERSION_ID: 2023
PRETTY_NAME: Amazon Linux 2023.12.20260831
ID_LIKE: fedora

Amazon Linux Version Details:
============================
Amazon Linux 2023
Amazon Linux image-id file present
```

AL2

```
Operating System Detection Results:
==================================
Is Amazon Linux: YES
Is Fedora: NO
Is Ubuntu: NO
```

```
Is Debian: NO
Is like Fedora: YES
Is like Debian: NO

Detailed OS Information:
========================
ID: amzn
VERSION_ID: 2
PRETTY_NAME: Amazon Linux 2
ID_LIKE: centos rhel fedora

Amazon Linux Version Details:
=============================
Amazon Linux 2
Amazon Linux image-id file present
```

Amazon Linux AMI

```
Operating System Detection Results:
===================================
Is Amazon Linux: YES
Is Fedora: NO
Is Ubuntu: NO
Is Debian: NO
Is like Fedora: YES
Is like Debian: NO

Detailed OS Information:
========================
ID: amzn
VERSION_ID: 2018.03
PRETTY_NAME: Amazon Linux AMI 2018.03
ID_LIKE: rhel fedora

Amazon Linux Version Details:
=============================
Amazon Linux AMI (version 1)
Amazon Linux image-id file present
```

Ubuntu

```
Operating System Detection Results:
===================================
```

```
Is Amazon Linux: NO
Is Fedora: NO
Is Ubuntu: YES
Is Debian: NO
Is like Fedora: NO
Is like Debian: YES

Detailed OS Information:
========================
ID: ubuntu
VERSION_ID: 26.04
PRETTY_NAME: Ubuntu 26.04 LTS
ID_LIKE: debian
```

Debian

```
Operating System Detection Results:
===================================
Is Amazon Linux: NO
Is Fedora: NO
Is Ubuntu: NO
Is Debian: YES
Is like Fedora: NO
Is like Debian: YES

Detailed OS Information:
========================
ID: debian
VERSION_ID: 13
PRETTY_NAME: Debian GNU/Linux 13 (trixie)
```

Fedora

```
Operating System Detection Results:
===================================
Is Amazon Linux: NO
Is Fedora: YES
Is Ubuntu: NO
Is Debian: NO
Is like Fedora: YES
Is like Debian: NO

Detailed OS Information:
```

```
========================
ID: fedora
VERSION_ID: 44
PRETTY_NAME: Fedora Linux 44 (Container Image)
```

Filesystem layout

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

This section covers the filesystem layout of an AL2027 system, including details specific to instances and to AL2027 based containers. The layout follows the Filesystem Hierarchy Standard (FHS). For more information, see the `file-hierarchy(7)` man page.

AL2027 completes the merge of `/usr/sbin` into `/usr/bin`. This is the one structural change from AL2023. For details, see Unified `/usr/bin` and `/usr/sbin`.

Topics

- / (the root directory)
- /boot (kernel, initramfs, bootloader)
- /etc (system configuration)
- /home (user home directories)
- /root (root user home directory)
- /srv (server payload)
- /tmp (small temporary files)
- /run (runtime data)
- /usr (system resources)
- /var (persistent variable system data)

/ (the root directory)

By default, AL2027 images are configured with a writable /, allowing privileged users to create new files and directories.

The top-level `/bin`, `/sbin`, `/lib`, and `/lib64` paths are symbolic links to their counterparts under `/usr`, so commands and libraries resolve through either spelling.

Note

We recommend restricting what each `systemd` service can access. This can include the `ReadOnlyPaths=/` directive, which makes `/` read-only for that service. For more information, see the `systemd.exec(5)` man page.

`/boot` (kernel, initramfs, bootloader)

Bootable AL2027 images are configured with `/boot` on the root file system. It holds the files needed to boot, such as the Linux kernel and the initramfs. Change the content of this directory only through the tools provided with the OS. The `/boot` path is only relevant for bootable images; it is unused in AL2027 container images.

`/boot/efi` (EFI System Partition)

Bootable AL2027 images mount the EFI System partition at `/boot/efi`. AL2027 AMIs boot with UEFI on arm64 and UEFI-preferred on x86_64, so this file system contains code and configuration critical to booting the system. It is managed by the OS. This path is not relevant for container images.

`/etc` (system configuration)

The `/etc` directory contains system-specific configuration. It is on the root file system and writable by privileged users.

Note

Many applications, including `systemd` and DNF, keep their default configuration under `/usr` and let you override it by placing configuration in `/etc`. Edit the override in `/etc` rather than the defaults in `/usr`: package updates overwrite files under `/usr`.

`/home` (user home directories)

Normal users have their home directories under `/home`. Software should always use the per-user `$HOME` environment variable rather than a pattern such as `/home/$USER`.

By default, AL2027 images have /home on the root file system, but software should not rely on this. The OS can be configured with /home as a separate file system that is mounted later during boot or only after a user authenticates.

> **Note**
>
> For systemd services that need at most read access to home directories, set ProtectHome=read-only, which makes /home, /root, and /run/user read-only for the service. For services that need no access, set ProtectHome=tmpfs, which replaces those paths with empty read-only tmpfs file systems in the service's sandbox. For more information, see the systemd.exec(5) man page.

/root (root user home directory)

The home directory of the root user is /root. It is deliberately separate from /home so that it is present even when the /home file system is not available. The systemd service best practice for /home applies to /root as well.

/srv (server payload)

The /srv directory is managed by the administrator of the system. AL2027 places no restrictions on how it is organized and installs nothing into it. It can be a separate file system that becomes available later in boot.

/tmp (small temporary files)

The /tmp directory is for small, size-bounded temporary files. As in AL2023, it is a tmpfs file system by default, with a size limit of 50% of RAM and a maximum of one million inodes. For larger temporary files, use /var/tmp instead.

Applications should prefer the path in the $TMPDIR environment variable over a hardcoded /tmp, so users can override the location.

The content of /tmp is cleaned at boot, and unused files are removed by a cleanup job that runs shortly after boot and then daily. To configure the cleanup, see the tmpfiles.d(5) and systemd-tmpfiles(8) man pages.

Warning

Because /tmp is shared, use safe methods to create temporary files. For details, see the upstream systemd documentation on Using /tmp and /var/tmp safely on the systemd.io website.

Note

We recommend setting the PrivateTmp= directive to yes or disconnected for systemd services, which gives the service private /tmp and /var/tmp directories that are not shared with the host or other services. See the systemd.exec(5) man page.

Note

In a container, the container runtime configuration dictates whether /tmp is tmpfs or a path on disk, and whether any cleanup process runs.

/run (runtime data)

System components use the /run directory to store small amounts of runtime data, such as socket files. It is a tmpfs file system, writable only by privileged programs. The legacy /var/run path is a symbolic link to /run.

The /run/log directory can hold logs before they are written to /var/log, or before the /var/log file system is available.

The /run/user/ path contains per-user runtime directories: individual tmpfs file systems that systemd mounts when a user logs in and removes when the user logs out. Reference them through the $XDG_RUNTIME_DIR environment variable, as described in the XDG Base Directory Specification on the freedesktop.org website, not by literal path.

/usr (system resources)

The /usr hierarchy is for vendor-supplied operating system resources. Except for the /usr/local hierarchy, nothing should modify anything under /usr except the OS package manager.

Software must assume that /usr can be read-only, and must not use it for volatile data. Software installed outside of the OS package manager belongs in /usr/local, not elsewhere in /usr, so that it cannot interfere with package operations.

Unified /usr/bin and /usr/sbin

In AL2027, /usr/sbin is a symbolic link to /usr/bin. The executables that AL2023 shipped in /usr/sbin, such as useradd, sysctl, and ip, now live in /usr/bin. This follows the same change in Unify bin and sbin on the Fedora Project wiki.

Existing paths keep working: because /sbin and /usr/sbin both resolve to /usr/bin, a script that calls, for example, /usr/sbin/ip or /sbin/ldconfig runs unchanged. Commands formerly restricted to administrator search paths now also appear in every user's $PATH; the commands themselves still require the same privileges to do their work.

/usr/bin (executables)

Executable files that appear in the standard search $PATH and are useful to invoke from a shell. Daemons and executables not useful from a shell live in /usr/lib or /usr/libexec.

/usr/include (C/C++ headers)

C and C++ header files, usually installed by packages with the -devel suffix.

/usr/lib and /usr/lib64 (shared libraries)

The /usr/lib64 path holds 64-bit shared libraries and architecture-dependent package data. AL2027 does not ship 32-bit (i686) packages, so all shared libraries are 64-bit. The /usr/lib path is for static, architecture-independent package data, and might include executables not usually invoked from a shell, which might also be found in /usr/libexec.

/usr/local (administrator-installed software)

The /usr/local hierarchy is reserved for the system administrator to install software that is not owned by the OS. The OS does not touch it. Its layout mirrors the / hierarchy.

/usr/share (shared resources)

Architecture-independent shared resources such as documentation, fonts, and time zone data. Package documentation is installed under /usr/share/doc.

/var (persistent variable system data)

The /var directory stores persistent variable system data.

/var/cache (cache)

Applications must be able to rebuild their /var/cache data from other sources, so erasing data here does not cause data loss. For example, DNF keeps its repository metadata cache under /var/cache/libdnf5 and rebuilds it on the next metadata refresh.

/var/lib (persistent system data)

System components store persistent, private data under /var/lib. In contrast to /var/cache, erasing data here causes data loss. For example, the RPM database lives in /var/lib/rpm, and the PostgreSQL database server stores database data in /var/lib/pgsql.

/var/log (persistent logs)

The /var/log directory holds persistent logs. Software should prefer the syslog(3) or sd_journal_print(3) API calls over writing log files directly. Read the systemd journal with journalctl; for more information, see the journalctl(1) man page. Applications that write their own log files under /var/log document their own rotation configuration.

/var/spool (queued data)

Persistent queued data, such as mail or printer queues.

/var/tmp (larger temporary files)

Although /tmp is a tmpfs file system, /var/tmp is a path on the root file system, and is the place for larger and longer-lived temporary files. A cleanup job removes files not recently accessed; to configure it, see the tmpfiles.d(5) and systemd-tmpfiles(8) man pages.

As with /tmp, prefer the $TMPDIR environment variable over the hardcoded path, use safe temporary file creation (see Using /tmp and /var/tmp safely on the systemd.io website), and set PrivateTmp= for systemd services.

Updating AL2027

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

Keep up to date with AL2027 releases to benefit from security updates and new features. AL2027 uses Deterministic upgrades through versioned repositories on AL2027, the same update model as AL2023. Each release locks to a specific set of package versions, which gives you consistent deployments across your fleet.

> **Warning**
>
> Running `sudo dnf upgrade --releasever=latest` is not best practice, and is likely to result in an OS update being first tested in production.
> Instead of `latest`, use a specific AL2027 release version. This ensures that you deploy the same changes across production instances that you previously tested. For example, `sudo dnf upgrade --releasever=2027.0.20260817` always updates to the 2027.0.20260817 release.

To check for and apply updates within the locked release version:

```
# Check for available updates
dnf check-update

# Apply all available updates
sudo dnf upgrade
```

To move to a newer AL2027 release, pass the target release version. For more information, see Manage package and operating system updates in AL2027.

```
sudo dnf upgrade --releasever=<version>
```

Topics

- Best practices for safely deploying updates
- Receive notifications on new updates
- Deterministic upgrades through versioned repositories on AL2027
- Manage package and operating system updates in AL2027
- Updating the Linux kernel on AL2027

Best practices for safely deploying updates

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

AL2027 has several features designed to help you deploy operating system updates safely, know what changed between updates, and, if necessary, revert to the previous version. This section reflects lessons we learned from more than a decade of internal and external use of Amazon Linux.

> **Warning**
>
> Running `sudo dnf upgrade --releasever=latest` is not best practice, and is likely to result in an OS update being first tested in production.
> Instead of `latest`, use a specific AL2027 release version. This ensures that you deploy the same changes across production instances that you previously tested. For example, `sudo dnf upgrade --releasever=2027.0.20260817` always updates to the 2027.0.20260817 release.

Plan for the deployment safety of OS updates. An unexpected negative interaction between your application and an OS update can cause impact up to a total outage. As with any software issue, the earlier an issue is detected, the less impact it has on end users.

Do not rely on two assumptions that are fundamentally not true:

1. The OS vendor will never make a mistake in an update to the OS.

2. The specific behavior of, or interface to, the OS that you rely on matches what the OS vendor considers something to be relied upon. In other words, both the OS vendor and you would agree that an update introduced a problem.

Treat the OS as another part of your deployment. Apply the same deployment safety mechanisms that you use for any other change to a production environment:

- Test all OS updates before deploying to production systems. Do not test new OS updates by deploying them to production.
- Use staged rollouts, such as wave or phase based deployments (for example 1%, 5%, 10%, 20%, 40%, 100% of a fleet), combined with monitoring. If a problem occurs, impact is restricted to a subset of the fleet and the rollout can be halted while you investigate.
- Know how to point your deployment systems at a previous known-good version of the OS. Mitigating impact is usually the first priority; reverting to a known-good version is a powerful tool. After the issue is resolved, move to a new known-good version rather than staying locked to the old one.

Deterministic upgrades through versioned repositories on AL2027 makes every change to the OS version repeatable. Each AL2027 release is a version you can lock to, with AMIs and container images that lock to that version. If a release causes a problem for your workload, you can immediately go back to the images of the prior release while you work out how to resolve it. Deterministic upgrades also protect in-progress deployments. A release that comes out mid-deployment does not affect the fleet: it keeps applying the exact release version the deployment started with.

The changes in each release are documented in the AL2027 Release Notes. Security issues addressed in package updates are covered by Amazon Linux Security advisories for AL2027.

Not taking OS updates promptly also causes issues. New releases contain bug and security fixes that are likely relevant to your environment. Configure your deployment systems so that taking a new release, testing it, and rolling it out is routine. For more information, see Security and Compliance in AL2027 and Manage package and operating system updates in AL2027.

Topics

- Preparing for minor updates
- Preparing for major updates

Preparing for minor updates

Preparing for smaller updates, such as a new point release of AL2027, is intended to require minimal effort. Read the AL2027 Release Notes for upcoming changes, and check package support timelines with the `dnf supportinfo` command. For more information, see Getting package support information.

Preparing for major updates

Updating to a new major version of an operating system requires planning, work to adapt to changed or removed functionality, and testing before deployment. You can prepare incrementally while still on the previous version by addressing deprecated functionality first.

For example, AL2023 documented System V init scripts as deprecated, and AL2027 removes support for them. Workloads that migrated to `systemd` unit files while still on AL2023 move to AL2027 without that change. The same applies to the other items in Deprecated functionality in AL2027: addressing them early turns a major version upgrade into a series of small, safe steps.

The list of deprecated functionality is updated over the lifetime of the OS. Check it regularly to prepare for the next major version of Amazon Linux.

Receive notifications on new updates

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

You can find out about new AL2027 releases in the following ways:

- Each release is documented in the AL2027 Release Notes, including the package changes it contains.
- On a running instance, the `dnf check-release-update` command reports when a newer release than the one the system is locked to is available. For more information, see Checking for newer repository versions with `dnf check-release-update`.
- An Amazon SNS topic sends a notification each time we release a new AL2027 AMI. This page includes the subscription steps.

Amazon SNS publishes a notification to the following topic each time we release a new AL2027 AMI.

```
arn:aws:sns:us-east-1:137112412989:amazon-linux-2027-ami-updates
```

The message includes the version of the new AMI.

You can receive these messages in several ways. We recommend the following method.

1. Open the Amazon SNS console.
2. In the navigation bar, change the AWS Region to **US East (N. Virginia)**, if necessary. You subscribe in the Region that hosts the topic.
3. In the navigation pane, choose **Subscriptions**, **Create subscription**.
4. For the **Create subscription** dialog box, do the following:
   a. For **Topic ARN**, copy and paste the following **Amazon Resource Name** (ARN): **`arn:aws:sns:us-east-1:137112412989:amazon-linux-2027-ami-updates`**.
   b. For **Protocol**, choose **Email**.
   c. For **Endpoint**, enter an email address that you can use to receive the notifications.
   d. Choose **Create subscription**.
5. You receive a confirmation email with the subject line "AWS Notification - Subscription Confirmation". Open the email and choose **Confirm subscription** to complete your subscription.

Deterministic upgrades through versioned repositories on AL2027

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

> **Note**
>
> By default, your AL2027 instance doesn't automatically receive additional security updates at launch. Your instance initially contains the updates that were available in the AL2027 release of the chosen AMI.

AL2027 uses deterministic upgrades through versioned repositories, the same update model as AL2023. The feature is turned on by default. To unlock a system so that every `dnf` operation uses the newest release, see Using persistent override with deterministic upgrade. Each AL2027 release is a version of the package repositories. Each AMI and container image locks to the release version it was built from. Whenever we release new package updates, there is a new version to lock to. New AMIs and container images lock to that new version.

With this model, you control consistency of package versions and updates across your environment:

- Multiple instances launched from the same AMI install identical package versions, even when you provision them at different times. Additional packages installed during provisioning come from the repository version of that AMI.
- AL2027 does not apply updates automatically. You apply updates on a schedule that meets your needs, to a release version you have tested.
- After we publish a release, it is immutable. Newer releases do not affect in-progress deployments.

Topics

- Differences between minor and major version upgrades
- Knowing when updates are available
- Control the package updates available from the AL2027 repositories
- Deterministic updates through instance replacement
- Using deterministic upgrades through versioned repositories

Differences between minor and major version upgrades

Major version releases of Amazon Linux (such as AL2023 to AL2027) include large-scale updates and might add, remove, or change packages. Upgrade to a new major version only after you test your application on that version.

Minor version releases of AL2027 include feature and security updates while keeping Linux features and the system library API stable. Testing your application before a minor update is not required, but staged rollouts remain best practice. For more information, see Best practices for safely deploying updates.

Knowing when updates are available

To apply an update, you need to know that one is available. The changes in each release are documented in the AL2027 Release Notes, and security issues addressed in package updates are covered by Amazon Linux Security advisories for AL2027. On a running system, `dnf check-release-update` reports newer releases. You can also Receive notifications on new updates.

For building derived AMIs when new AL2027 AMIs are released, EC2 Image Builder can automatically build, patch, and test AMIs. For patching in-place across a fleet, you can use tools such as AWS Systems Manager Patch Manager.

Other AMIs and container images based on AL2027 might have their own release schedule and notification methods. When you use derived images, check the publisher's documentation for when updates are released.

Control the package updates available from the AL2027 repositories

When we publish a new version of the AL2027 repositories, all previous versions remain available. By default, the system locks to the release version that was used to build the AMI or container image. To move to a newer version:

1. Discover available repository versions.

```
sudo dnf check-release-update
```

2. Upgrade to the version you selected.

```
sudo dnf upgrade --releasever=<version>
```

The command lists the package updates and asks for confirmation before applying them. After the upgrade completes, the new release version becomes the default release version for all future `dnf` operations. For more information, see Manage package and operating system updates in AL2027.

Deterministic updates through instance replacement

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

With Deterministic upgrades through versioned repositories on AL2027 in AL2027, you can use instance replacement as a deterministic and safe way to roll out updated versions of AL2027. As you progressively roll out a new version, you can revert to the previous AMI at any point while you determine the cause of an issue.

Using instance replacement rather than patching in-place makes updates more deterministic and predictable. Launching new capacity is a well-tested code path with clear A and B states. You can fully test both states in a CI/CD system before deployment starts. In-place patching passes through many intermediary states between before and after applying updates, which is harder to test for all combinations.

An OS update strategy of instance replacement with deterministic updates fits well into blue/green, wave, and phase based deployment models.

Using deterministic upgrades through versioned repositories

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

Topics

- Using a deterministic upgraded system
- Selective update of a deterministic upgraded system

- Using persistent override with deterministic upgrade

Using a deterministic upgraded system

Deterministic upgrades let you test every change before it reaches production. Each AL2027 AMI is locked to a specific release, so instances launched from it start with the same package versions. In-place updates target a specific release version, so the same tested set of changes rolls out across a fleet. Test each new AMI or release version in your CI/CD pipeline before deployment. This catches issues before they reach production.

When you run the `dnf upgrade` command, the system checks for upgrades in the repository version that the `releasever` variable specifies. A valid `releasever` is either `latest` or a date-stamped version such as `2027.0.20260817`. The following methods determine the value of `releasever`, in descending priority. Method 1 overrides methods 2 and 3, and method 2 overrides method 3.

1. The command line flag `--releasever=`*`version`*, if it's used.
2. The value in the override variable file `/etc/dnf/vars/releasever`, if it's set.
3. The currently installed version of the `system-release` package.

To see the release version your system is locked to, query the `system-release` package. In the following example, the system is locked to the 2027.0.20260803 release:

```
rpm -q system-release
```

```
system-release-2027.0.20260803-1.amzn2027.noarch
```

In a newly installed system, the override variable file `/etc/dnf/vars/releasever` does not exist, so the system is locked to the installed version of `system-release`. Because the locked release is immutable, running `sudo dnf upgrade` on an up-to-date system makes no changes:

```
sudo dnf upgrade
```

```
Updating and loading repositories:
Repositories loaded.
Nothing to do.
```

To upgrade to a specific release version, pass it with the `--releasever` option:

```
sudo dnf upgrade --releasever=2027.0.20260817
```

```
Updating and loading repositories:
 Amazon Linux 2027 repository           100% |  11.7 MiB/s |   2.4 MiB |  00m00s
Repositories loaded.
Package                                   Arch   Version
 Repository       Size
Upgrading:
 amazon-linux-repo-cdn                    noarch 0:2027.0.20260817-2.amzn2027
 amazonlinux   1.1 KiB
   replacing amazon-linux-repo-cdn        noarch 0:2027.0.20260803-1.amzn2027
 amazonlinux   1.1 KiB
 krb5-libs                                x86_64 0:1.22.2-6.amzn2027
 amazonlinux   2.4 MiB
   replacing krb5-libs                    x86_64 0:1.22.2-1.amzn2027
 amazonlinux   2.4 MiB

   ... [ list edited for clarity ]

 openssl                                  x86_64 1:3.5.7-2.amzn2027
 amazonlinux   1.8 MiB
   replacing openssl                      x86_64 1:3.5.5-3.amzn2027
 amazonlinux   1.8 MiB

   ... [ list edited for clarity ]

 system-release                           noarch 0:2027.0.20260817-2.amzn2027
 amazonlinux  15.6 KiB
   replacing system-release               noarch 0:2027.0.20260803-1.amzn2027
 amazonlinux  15.6 KiB
 systemd                                  x86_64 0:260.1-11.amzn2027.0.8
 amazonlinux  13.3 MiB
   replacing systemd                      x86_64 0:260.1-11.amzn2027.0.7
 amazonlinux  13.3 MiB

   ... [ list edited for clarity ]

Transaction Summary:
 Upgrading:         23 packages
 Replacing:         23 packages
```

```
Total size of inbound packages is 28 MiB. Need to download 28 MiB.
After this operation, 81 KiB will be freed (install 102 MiB, remove 102 MiB).
```

Because the `--releasever` option overrides both `system-release` and `/etc/dnf/vars/releasever`, this upgrade does the following:

1. Replaces all installed packages that changed between the previous and new versions.
2. When the target release contains a newer `system-release` package, the transaction includes it, which locks the system to the new release for all future `dnf` operations.

By always specifying which AL2027 release to update to, you get a deterministic set of changes across a fleet: you launched version A, updated to B, and then updated to C.

Selective update of a deterministic upgraded system

> **Note**
>
> We recommend installing all updates in a new release rather than selecting specific updates. Applying only part of an update should be the exception, not standard practice. For restricting updates to those from a specific security advisory, see Applying security updates in-place.

You might want to install selected packages from a newer release while leaving the system locked to the original release version. First, identify the packages that you want to upgrade:

```
sudo dnf check-update --releasever=latest
```

```
Updating and loading repositories:
 Amazon Linux 2027 repository           100% |   6.6 MiB/s |   2.4 MiB |  00m00s
Repositories loaded.
Upgrades
amazon-linux-repo-cdn.noarch        2027.0.20260817-2.amzn2027 amazonlinux
krb5-libs.x86_64                    1.22.2-6.amzn2027          amazonlinux
libssh.x86_64                       0.12.2-1.amzn2027          amazonlinux
libssh-config.noarch                0.12.2-1.amzn2027          amazonlinux
openldap.x86_64                     2.6.13-3.amzn2027          amazonlinux
openssh.x86_64                      9.9p1-28.amzn2027          amazonlinux
```

```
openssh-clients.x86_64              9.9p1-28.amzn2027          amazonlinux
openssl.x86_64                      1:3.5.7-2.amzn2027         amazonlinux
openssl-fips-provider-latest.x86_64 1:3.5.7-2.amzn2027         amazonlinux
openssl-libs.x86_64                 1:3.5.7-2.amzn2027         amazonlinux
python-unversioned-command.noarch   3.14.5-2.amzn2027.0.4      amazonlinux
python3.x86_64                      3.14.5-2.amzn2027.0.4      amazonlinux
python3-libs.x86_64                 3.14.5-2.amzn2027.0.4      amazonlinux
qrencode-libs.x86_64                4.1.1-3.amzn2027.0.2       amazonlinux
system-release.noarch               2027.0.20260817-2.amzn2027 amazonlinux
systemd.x86_64                      260.1-11.amzn2027.0.8      amazonlinux
systemd-libs.x86_64                 260.1-11.amzn2027.0.8      amazonlinux
systemd-networkd.x86_64             260.1-11.amzn2027.0.8      amazonlinux
systemd-pam.x86_64                  260.1-11.amzn2027.0.8      amazonlinux
systemd-resolved.x86_64             260.1-11.amzn2027.0.8      amazonlinux
systemd-shared.x86_64               260.1-11.amzn2027.0.8      amazonlinux
systemd-sysusers.x86_64             260.1-11.amzn2027.0.8      amazonlinux
systemd-udev.x86_64                 260.1-11.amzn2027.0.8      amazonlinux
```

Then upgrade only those packages by naming them. Naming specific packages keeps the `system-release` package, and therefore the version lock, unchanged. The transaction includes related packages that the named packages require at the same version, such as `openssl-fips-provider-latest` here:

```
sudo dnf upgrade --releasever=latest openssl openssl-libs
```

```
Updating and loading repositories:
Repositories loaded.
Package                                 Arch   Version            Repository
 Size
Upgrading:
 openssl                                x86_64 1:3.5.7-2.amzn2027 amazonlinux   1.8
 MiB
   replacing openssl                    x86_64 1:3.5.5-3.amzn2027 amazonlinux   1.8
 MiB
 openssl-fips-provider-latest           x86_64 1:3.5.7-2.amzn2027 amazonlinux   2.5
 MiB
   replacing openssl-fips-provider-latest x86_64 1:3.5.5-3.amzn2027 amazonlinux   2.5
 MiB
 openssl-libs                           x86_64 1:3.5.7-2.amzn2027 amazonlinux   6.9
 MiB
   replacing openssl-libs               x86_64 1:3.5.5-3.amzn2027 amazonlinux   6.9
 MiB
```

```
Transaction Summary:
 Upgrading:          3 packages
 Replacing:          3 packages

Total size of inbound packages is 4 MiB. Need to download 4 MiB.
After this operation, 14 KiB will be freed (install 11 MiB, remove 11 MiB).
```

Note

Running `sudo dnf upgrade --releasever=latest` without naming packages updates all packages, including `system-release`. The system then remains locked to the new `system-release` version unless you set the persistent override.

Using persistent override with deterministic upgrade

Instead of adding `--releasever=latest` to each command, you can *unlock* the system by setting the variable file to `latest`. This reverts AL2027 to the AL2 update model, where every package manager invocation uses the latest release and does not lock to any specific version of the OS:

```
echo latest | sudo tee /etc/dnf/vars/releasever
```

```
latest
```

Warning

Unlocking the package manager with a persistent override removes the ability to test an OS update before it reaches production. A new AL2027 release can happen at any point in time, so all uses of `latest` in production carry the risk of discovering an incompatibility between your application and an OS update in production.

To restore the default locking behavior, remove the override file:

```
sudo rm /etc/dnf/vars/releasever
```

Rather than disabling deterministic upgrades, we recommend replacing instances with ones launched from a new AMI. If instance replacement is not an option, use tools such as AWS Systems Manager Patch Manager to orchestrate applying updates across a fleet, or EC2 Image Builder to automatically build, patch, and test your own AMIs derived from AL2027 base images.

Manage package and operating system updates in AL2027

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

AL2027 AMIs and container images lock to a specific version of the AL2027 repositories. To apply security and bug fixes to an existing system, update to a newer release version. Alternatively, launch a new instance from a newer AMI. This section describes how to manage packages and repositories on a running system. For the complete command reference, see the DNF5 documentation on the dnf5.readthedocs.io website.

We recommend applying *all* updates available in a new AL2027 release. Picking only security updates, or only specific updates, should be the exception rather than the rule. To list which security advisories are relevant to an instance, see Listing applicable advisories. To install only the updates from a specific advisory, see Applying security updates in-place.

> **Important**
>
> If you want to report a vulnerability or have a security concern regarding AWS cloud services or open source projects, contact AWS Security using the Vulnerability Reporting page

Topics

- Checking for available package updates
- Applying updates using DNF and repository versions
- Checking for newer repository versions with dnf check-release-update
- Getting package support information
- Launching an instance with a specific repository version enabled

- Adding, enabling, or disabling new repositories
- Adding repositories with cloud-init
- Automatic service restart after updates
- When is a reboot required to apply updates?

Checking for available package updates

Use the `dnf check-update` command to check for updates. In AL2027 this is a compatibility alias for `dnf check-upgrade`; both spellings work. Without options, the command only checks the release version the system is locked to. Add `--releasever=`*`version`* to check what a newer release would update.

In the following example, a system on the 2027.0.20260803 release checks what the 2027.0.20260817 release would update:

```
dnf check-update --releasever=2027.0.20260817
```

```
Updating and loading repositories:
 Amazon Linux 2027 repository           100% |  15.1 MiB/s |   2.4 MiB |  00m00s
Repositories loaded.
Upgrades
amazon-linux-repo-cdn.noarch        2027.0.20260817-2.amzn2027 amazonlinux
krb5-libs.x86_64                    1.22.2-6.amzn2027          amazonlinux
libssh.x86_64                       0.12.2-1.amzn2027          amazonlinux
libssh-config.noarch                0.12.2-1.amzn2027          amazonlinux
openldap.x86_64                     2.6.13-3.amzn2027          amazonlinux
openssh.x86_64                      9.9p1-28.amzn2027          amazonlinux
openssh-clients.x86_64              9.9p1-28.amzn2027          amazonlinux
openssl.x86_64                      1:3.5.7-2.amzn2027         amazonlinux
openssl-fips-provider-latest.x86_64 1:3.5.7-2.amzn2027         amazonlinux
openssl-libs.x86_64                 1:3.5.7-2.amzn2027         amazonlinux
python-unversioned-command.noarch   3.14.5-2.amzn2027.0.4      amazonlinux
python3.x86_64                      3.14.5-2.amzn2027.0.4      amazonlinux
python3-libs.x86_64                 3.14.5-2.amzn2027.0.4      amazonlinux
qrencode-libs.x86_64                4.1.1-3.amzn2027.0.2       amazonlinux
system-release.noarch               2027.0.20260817-2.amzn2027 amazonlinux
systemd.x86_64                      260.1-11.amzn2027.0.8      amazonlinux
systemd-libs.x86_64                 260.1-11.amzn2027.0.8      amazonlinux
systemd-networkd.x86_64             260.1-11.amzn2027.0.8      amazonlinux
systemd-pam.x86_64                  260.1-11.amzn2027.0.8      amazonlinux
```

```
systemd-resolved.x86_64              260.1-11.amzn2027.0.8      amazonlinux
systemd-shared.x86_64                260.1-11.amzn2027.0.8      amazonlinux
systemd-sysusers.x86_64              260.1-11.amzn2027.0.8      amazonlinux
systemd-udev.x86_64                  260.1-11.amzn2027.0.8      amazonlinux
```

To always check against the newest release, use `--releasever=latest`. On this system it reports the same packages, because 2027.0.20260817 is the newest release:

```
dnf check-update --releasever=latest
```

The command exits with return code 100 if newer packages are available, and 0 if there aren't any. Checking for updates does not require elevated privileges.

Applying updates using DNF and repository versions

New package updates and security updates are made available in new repository versions only. For systems locked to an earlier release, pass the target release version to apply all updates available in it:

```
sudo dnf upgrade --releasever=<version>
```

DNF lists the package updates and asks for confirmation before applying them. Use the `-y` option to skip the confirmation prompt in scripts and other automation. After the upgrade, the system is locked to the new release version.

Applying updates is a privileged operation. On an Amazon EC2 instance, run the command as the `root` user, for example with `sudo`. In a container, elevated privileges are typically not required.

Applying all updates moves existing systems to the same package set as launching an updated AMI, which reduces variation of package versions across a fleet.

Checking for newer repository versions with `dnf check-release-update`

The `dnf check-release-update` command reports when a newer AL2027 release than the one the system is locked to is available, and prints the exact command to upgrade to it:

```
sudo dnf check-release-update
```

```
WARNING:
  A newer release of Amazon Linux is available.

  Available Versions:

  Version 2027.0.20260817:
    Run the following command to upgrade to 2027.0.20260817:

      dnf upgrade --releasever=2027.0.20260817

    Release notes:
     https://docs.aws.amazon.com/linux/al2027/release-notes/
relnotes-2027.0.20260817.html
```

If the system is already on the newest release, the command prints nothing and exits with return code 0.

The `dnf-plugin-release-notification` package provides the command, which is installed by default on AL2027 AMIs. AL2027 container images don't include it. To use it in a container, install the package first:

```
sudo dnf install dnf-plugin-release-notification
```

Getting package support information

Each package in AL2027 has an associated support timeline. Query it with the `dnf supportinfo` command:

```
dnf supportinfo --pkg bash
```

```
Name                         : bash
Version                      : 5.3.0-2.amzn2027
State                        : installed
Origin                       : Amazon Linux 2027 Core
Support Timeline             : from 2026-09-03     : supported (Low, Medium, Important,
 Critical)
                             : from 2027-03-31     : unsupported
Package Note                 : Amazon Linux will support this package until the end of
 the AL2027 public preview period
```

The `dnf-plugin-support-info` package provides the command. If it isn't present on your system, install it first:

```
sudo dnf install dnf-plugin-support-info
```

> **Note**
>
> During the preview period, the support timelines reflect the preview terms. Support statements for GA will be published before the public release.

Launching an instance with a specific repository version enabled

You can add DNF commands to a user data script to control which packages are installed when an AL2027 instance launches. In the following example, the user data script makes sure that every instance launched with it has the same package updates installed:

```
#!/bin/bash
dnf upgrade -y --releasever=2027.0.20260817
# Additional setup and install commands below
dnf install -y httpd mariadb1011-server
```

User data scripts run as the `root` user at launch. For more information, see User data and shell scripts in the *Amazon EC2 User Guide*.

> **Note**
>
> Instead of using a user data script, launch the latest AL2027 AMI, or a custom AMI based on it. The latest AMI has all current updates installed and is locked to the matching repository version.

Adding, enabling, or disabling new repositories

> **Warning**
>
> Only add repositories designed to be used with AL2027. Repositories designed for other distributions might appear to work, but there is no guarantee they keep working across package updates.

Repositories are defined in configuration files in `/etc/yum.repos.d/`. Many third-party repositories provide either the configuration file content or an installable package that includes it. To list the configured repositories:

```
dnf repolist --all
```

```
repo id                repo name                                          status
amazonlinux            Amazon Linux 2027 repository                      enabled
amazonlinux-debuginfo Amazon Linux 2027 repository - Debug           disabled
amazonlinux-source     Amazon Linux 2027 repository - Source packages disabled
```

Repository management commands are provided by the `dnf5-plugins` package. If `dnf config-manager` reports an unknown command, install it first:

```
sudo dnf install dnf5-plugins
```

To add a repository from a repository configuration file URL:

```
sudo dnf config-manager addrepo --from-repofile=<url>
```

To define a repository directly:

```
sudo dnf config-manager addrepo --id=<repo-id> --set=baseurl=<url>
```

To enable or disable a repository:

```
sudo dnf config-manager setopt <repo-id>.enabled=1
```

```
sudo dnf config-manager setopt <repo-id>.enabled=0
```

> **Note**
>
> The DNF (version 4) spelling `dnf config-manager --add-repo` changed in DNF5 to the `addrepo` subcommand shown earlier in this section. The `yum-config-manager` command is not available in AL2027. Update scripts that use either spelling.

Adding repositories with cloud-init

You can also add a repository at launch time with `cloud-init` user data. The following template writes a repository definition to `/etc/yum.repos.d/`:

```
#cloud-config
yum_repos:
  repository.repo:
    baseurl: https://www.example.com/
    enabled: true
    gpgcheck: true
    gpgkey: file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EXAMPLE
    name: Example Repository
```

You can add a `packages:` section to the same file to install packages from the default repositories or from the repository you added. For details of the file format, see Adding a YUM repository in the `cloud-init` documentation on the cloudinit.readthedocs.io website. Pass the file to the instance with the `--user-data` option of `aws ec2 run-instances`.

Automatic service restart after updates

After you apply updates, running services can still use the old, replaced versions of libraries until they restart. The `smart-restart` package restarts affected `systemd` services automatically after each DNF transaction. It uses the `dnf needs-restarting` command to find the affected services, and it decides whether a full reboot is advised. When a reboot is advised, it writes a hint marker file at `/run/smart-restart/reboot-hint-marker`.

```
sudo dnf install smart-restart
```

After installation, every subsequent transaction triggers the `smart-restart` logic.

To exclude services from automatic restart, add a file with the `-denylist` suffix in `/etc/smart-restart-conf.d/`. Excluded services also don't count toward the reboot decision. All `*-denylist` files in the directory are evaluated:

```
cat /etc/smart-restart-conf.d/custom-denylist
```

```
# Services that smart-restart must not restart
myservice.service
```

To run your own steps around a restart, place scripts with the `-pre-restart` or `-post-restart` suffix in `/etc/smart-restart-conf.d/`. When order matters, prefix the script names with a number.

When is a reboot required to apply updates?

In some situations, AL2027 requires a reboot to apply updates:

- Updates to the Linux kernel package require a reboot to activate the new kernel.
- On Amazon EC2 metal instances, CPU microcode updates (the `microcode_ctl` package for Intel and the `amd-ucode-firmware` package for AMD) activate on the next reboot. For virtualized instances, the underlying AWS Nitro System handles microcode updates for you.
- Some running `systemd` services only function correctly after a full system restart. The `smart-restart` mechanism informs you about these situations by leaving reboot hints. See Automatic service restart after updates.

Updating the Linux kernel on AL2027

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

AL2027 features a single kernel major version at launch. Additional major kernel versions will be released annually. Users are advised to check this section after a newer kernel major version is released in the future.

Query package support with the AL2027 support info plugin

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

Use the `dnf supportinfo` command, from the `dnf-plugin-support-info` package, to check the support status of a package: whether it is supported now, what severity levels receive patches, and when its support status changes. The following topics describe how to use the plugin and the structure of the support data behind it.

Topics

- Using the dnf supportinfo plugin
- SupportInfo XML structure

Using the `dnf supportinfo` plugin

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

Before you apply updates or plan a maintenance window, you can check whether a package stays supported for the period you need. Use the `dnf supportinfo` command, from the `dnf-plugin-support-info` package, to check the support status of an AL2027 package: whether it is supported now, what severity levels receive patches, and when its support status changes.

For more information about package support statements, see Package support information in the *AL2027 Release Notes*.

> **Note**
>
> During the preview, the reported support timeline for each package ends with the preview period. Support timelines for the AL2027 releases will be published with the public releases.

Topics

- Prerequisites
- Getting started
- Query packages
- Filter packages
- Structured output
- Manage the cache
- Non-root usage
- Configuration reference
- Output fields reference
- Troubleshooting
- Additional resources

Prerequisites

Before you begin, make sure that you have the following:

- An AL2027 instance
- The `dnf-plugin-support-info` package, available from the default AL2027 repositories

No root privileges are required to run queries.

Getting started

Install the plugin.

```
$ sudo dnf install dnf-plugin-support-info
```

Query a package to verify the installation.

```
$ dnf supportinfo --pkg glibc
```

The output shows the package's support status and timeline:

```
Name                         : glibc
Version                      : 2.44-2.amzn2027
State                        : installed
Origin                       : Amazon Linux 2027 Core
Support Timeline             : from 2026-09-03     : supported (Low, Medium, Important,
 Critical)
                             : from 2027-03-31     : unsupported
Package Note                 : Amazon Linux will support this package until the end of
 the AL2027 public preview period
```

Query packages

Pass `--pkg` with one or more package names to show their support details.

```
$ dnf supportinfo --pkg package [package ...]
```

To query multiple packages in one command:

```
$ dnf supportinfo --pkg glibc bash openssl-libs
```

Filter packages

The `--show` flag filters packages by state, support level, or both.

```
$ dnf supportinfo --show filter [filter ...]
```

There are two filter types:

Category	Values	Source
State filters (fixed)	`installed` , `available` , `unavailable` , `all`	Built into the plugin

Category	Values	Source
Support level filters (dynamic)	Varies. For example, `full_support` and `eos`	Loaded from the support data, so the values depend on what AL2027 defines

Multiple filters of the same type combine with OR. Filters of different types combine with AND. For example, `--show installed eos` matches packages that are installed and at the `eos` support level. If a filter name exists in both categories, use the prefix syntax: `state:`*`name`* or `support:`*`name`*.

Run `--list-filters` to see all valid filter values grouped by category:

```
$ dnf supportinfo --list-filters
```

```
State filters:
  all                  - All packages (installed, available, and unavailable)
  available            - Packages available in repositories but not installed
  installed            - Packages currently installed on the system
  unavailable          - Packages in support info but not in any repository

Support level filters:
  eos                  - End of support - no further updates
  full_support         - Full security and bug fix support for all severities
```

To list the support information for all installed packages:

```
$ dnf supportinfo --show installed
```

The output lists each package in a table:

```
Package                          Version                                State
 Support Level    Current Status         Until (Next Phase)           Note
alternatives                     1.33-3.amzn2027.0.2                   installed
                                                                      Amazon Linux will
 support this package until the end of the AL2027 public preview period
audit-libs                       4.1.3-1.amzn2027.0.1                  installed
                                                                      Amazon Linux will
 support this package until the end of the AL2027 public preview period
```

```
# Output truncated for brevity
```

To use the prefix syntax:

```
$ dnf supportinfo --show state:installed support:full_support
```

Structured output

For scripts and automation, `--showjson` and `--showxml` produce machine-readable output.

To get JSON output:

```
$ dnf supportinfo --pkg bash --showjson
```

The output contains the package's phases, origin, and timeline with support level details:

```
{
  "name": "bash",
  "version": "5.3.0-2.amzn2027",
  "state": "installed",
  "origin": "Amazon Linux 2027 Core",
  "phases": [
    {
      "name": "supported",
      "start_date": "2026-09-03",
      "patch_priority": "Full Support",
      "support_level_description": "Full security and bug fix support for all
 severities",
      "covered_severities": "Low, Medium, Important, Critical"
    },
    {
      "name": "unsupported",
      "start_date": "2027-03-31",
      "patch_priority": "End of Support",
      "support_level_description": "End of support - no further updates"
    }
  ],
  "note": "Amazon Linux will support this package until the end of the AL2027 public
 preview period"
}
```

To get XML output:

```
$ dnf supportinfo --pkg bash --showxml
```

The output is an XML document with a `package_support` root element containing the package's phases, start dates, and support level details. For the schema reference, see SupportInfo XML structure.

Manage the cache

Your queries use a local cache of the support data, which avoids unnecessary network requests. When the cache expires, the plugin checks for changes to the remote file. If the file is unchanged, the cache refreshes without downloading. If it changed, the plugin downloads the updated file. On network errors, the plugin uses the cached file if one is available.

To download and cache the latest support data, for example in automation that needs fresh data without a full query:

```
$ dnf supportinfo --sync
```

```
Support info sync complete
```

To clear the cache:

```
$ dnf supportinfo --clean-cache
```

```
Cleaned support info cache.
```

Non-root usage

All commands work without root privileges. When run as root, the cache is stored at `/var/cache/libdnf5/support-info/`. When run as a non-root user, the cache goes to `~/.cache/libdnf5/support-info/` in the user's home directory.

Configuration reference

The plugin reads configuration from `/etc/dnf/plugins/supportinfo.conf`. The following is the configuration shipped with AL2027:

```
[main]
baseurl = https://cdn-al2027.amazonlinux.com/core/AL2027-supportinfo-1.0.xml
templateurl = https://cdn-al2027.amazonlinux.com/core/supportinfo-1.0.xsd
```

Option	Description	Default
`baseurl`	URL to the support info XML file.	Required
`templateurl`	URL to the XSD schema for validation.	None
`metadata_expire`	Cache duration in seconds.	`1800`

Output fields reference

The `--pkg` output contains the following fields:

Field	Description
Name	Package name
Version	Installed or available version
State	`installed` , `available` , or `unavailable`
Origin	Repository providing the package
Support Timeline	Chronological list of lifecycle phases with dates and the severities covered in each phase
Package Note	Additional context, such as support period notes

Troubleshooting

If query results look stale, sync the cache manually:

```
$ dnf supportinfo --sync
```

If the issue persists, clear the cache and sync again:

```
$ dnf supportinfo --clean-cache
dnf supportinfo --sync
```

Additional resources

- DNF support info plugin on the GitHub website
- amazon-linux-supportinfo parsing library on the GitHub website

SupportInfo XML structure

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

Use this reference to understand the v1.0 XML schema that defines package support lifecycles, timelines, and classifications. The schema is independent of DNF. You can consume the data directly from any tool; the `dnf supportinfo` plugin is one such consumer. On AL2027, the schema definition (XSD) is published at `https://cdn-al2027.amazonlinux.com/core/supportinfo-1.0.xsd` and the support data (XML) at `https://cdn-al2027.amazonlinux.com/core/AL2027-supportinfo-1.0.xml`.

To parse this XML directly, use the open-source amazon-linux-supportinfo library on the GitHub website, which implements the schema described on this page, instead of writing a custom parser.

Topics

- How the plugin uses this file
- Example
- Element reference
- Referential integrity

How the plugin uses this file

From this file, the plugin reports which support level a package is at (these values populate the `--show` filter options), when lifecycle phases change (displayed in the `--pkg` timeline), the severity coverage for each support level, and where the package comes from (origin).

Example

The following example shows the structure with all sections, taken from the AL2027 support data published during the preview:

```
<?xml version='1.0' encoding='UTF-8'?>
<package_support current_as="2026-08-11T00:42:30.979109+00:00" schema_version="1.0">
  <lifecycles>
    <lifecycle name="eol_lc" note="al2027_public_preview"
               display_name="AL2027 Public Preview Lifecycle"
               description="Support timeline for Amazon Linux 2027 public preview
 packages">
      <phase name="supported" support_level="full_support"
             start_milestone="AL2027_PUBLIC_PREVIEW" display_name="Supported"/>
      <phase name="unsupported" support_level="eos" start_date="2027-03-31"
             display_name="End of Life"/>
    </lifecycle>
  </lifecycles>
  <support_milestones>
    <milestone name="AL2027_PUBLIC_PREVIEW" date="2026-09-03"
               display_name="AL2027 Public Preview"
               description="Amazon Linux 2027 public preview date"/>
  </support_milestones>
  <support_levels>
    <support_level name="full_support" severities="Low, Medium, Important, Critical"
                   description="Full security and bug fix support for all severities"
                   display_name="Full Support"/>
    <support_level name="eos" severities=""
                   description="End of support - no further updates"
                   display_name="End of Support"/>
  </support_levels>
  <packages>
    <package lifecycle="eol_lc" name="7zip" origin="al2027_core"/>
  </packages>
  <package_origins>
    <package_origin dist="amzn2027" name="al2027_core" repo_id="amazonlinux"
```

```
                    vendor="amazonlinux" display_name="Amazon Linux 2027 Core"
                    description="Core Amazon Linux 2027 repository containing base OS
 packages"/>
  </package_origins>
  <package_classes>
    <package_class name="default">
      <summary>Default class for Amazon Linux packages</summary>
      <text/>
    </package_class>
  </package_classes>
  <notes>
    <note name="al2027_public_preview">Amazon Linux will support this package until the
 end
of the AL2027 public preview period</note>
  </notes>
</package_support>
```

Element reference

The root element, `package_support`, carries the schema format version in `schema_version` (currently `1.0`) and the generation timestamp in `current_as`. It contains the following element families.

Element	Purpose	Key attributes
`lifecycle`	A sequence of phases that a package transitions through over time. The plugin walks the phases in order, finds the active one, and reports its support level.	`name` (referenced by `package@lifecycle`), `note`, `display_name`
`phase`	One stage within a lifecycle. Each phase is active from its start until the next phase begins. The last phase remains active indefinitely.	`name`, `support_level`, and exactly one of `start_date` (ISO date) or `start_milestone`
`milestone`	A reusable date marker that phases reference. When a milestone date changes, every phase referencing it updates automatically.	`name`, `date` (ISO date)

Element	Purpose	Key attributes
support_level	A support tier and the severities it covers. Each name becomes a valid --show filter value.	name, severities (comma-separated, empty means no patches), description
package	Maps an RPM package name to its lifecycle, origin, and optional class. The combination of name and origin must be unique.	name, lifecycle, origin, package_class
package_class	Categorizes packages by their role in the system, with a summary and text child element.	name (referenced by package@package_class)
package_origin	Identifies the repository source of a package. The display name appears as the Origin field in --pkg output.	name, repo_id, dist, vendor
note	Freeform text that a lifecycle references for additional context. Appears as the Package Note field in --pkg output.	name (referenced by lifecycle@note)

Referential integrity

The schema links elements together through name references. Each source attribute must match the name of an existing target element:

- package@lifecycle references lifecycle@name
- package@origin references package_origin@name
- package@package_class references package_class@name
- phase@support_level references support_level@name
- phase@start_milestone references milestone@name
- lifecycle@note references note@name

Getting started with programming runtimes on AL2027

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

The following table lists the language runtimes available in AL2027 and the versions removed from AL2023.

Topics

- Databases
- Python in AL2027
- Java in AL2027
- Node.js in AL2027
- TypeScript in AL2027
- Go in AL2027
- Rust in AL2027
- .NET in AL2027
- PHP in AL2027
- Ruby in AL2027
- Swift in AL2027
- C, C++, and Fortran in AL2027
- Perl in AL2027

Language runtimes in AL2027

Language	Available in AL2027	Removed from AL2023
Python	3.14 (system python), 3.15 (beta)	3.9, 3.11, 3.12, 3.13
PHP	8.5	8.1, 8.2, 8.3, 8.4

Language	Available in AL2027	Removed from AL2023
Node.js	24	18, 20, 22
Ruby	3.4	3.2
Java (Corretto)	8, 11, 17, 21, 25	22, 23, 24, 26
.NET	10.0	6.0, 8.0, 9.0
Rust	1.97 or later	—
Go	1.25	—
Swift	6.3 (new)	—
Perl	5.42	5.32

Important

If you are using Python 3.9, PHP 8.1–8.4, Node.js 18–22, or .NET 6, 8, or 9, you must upgrade to the versions available in AL2027 before migrating.

Databases

The following databases are available in the AL2027 preview:

Databases in AL2027

Database	Version	Notes
MariaDB	10.11.18	
MariaDB	11.4.12	
MariaDB	11.8.8	New in AL2027.
PostgreSQL	16.14	

Database	Version	Notes
PostgreSQL	17.10	
PostgreSQL	18.4	New in AL2027.
Memcached	1.6.42	New in AL2027.

Python in AL2027

AL2027 Preview

AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

With AL2027, you can use Python 3.14 at `/usr/bin/python3`. This is the system Python for AL2027. Python versions 3.9, 3.10, 3.11, 3.12, and 3.13 from AL2023 are not available in AL2027 at this time.

The system Python remains Python 3.14 for the life of AL2027. Later Python versions are provided in separate, name-spaced packages, the same way they are in AL2023.

Note

Do not change what the `/usr/bin/python3` symlink points to. Changing it might break core AL2027 functionality.

Migrating to Python 3.14

If you are moving an application from AL2023 that targets an earlier Python version, for more information about upgrading, see the Python 3.14 release notes on the Python documentation website.

Python modules in AL2027

With AL2027, you can install commonly used Python libraries as RPMs using `dnf`, without building from source. These RPMs target the system version of Python. To find available modules, run:

```
dnf search python3
```

Package Installation Security

AL2027 turns on a *dependency cooldown* for *pip* by default. The configuration in `/etc/pip.conf` sets `uploaded-prior-to = P1D`, so *pip* skips the package versions that were published less than one day ago.

To override the cooldown for a single command, use `--uploaded-prior-to=P0D`:

```
pip install --uploaded-prior-to=P0D package
```

To change the cooldown setting, edit `/etc/pip.conf`. For more information, see Secure your npm and pip package updates in Amazon Linux on the AWS Security Blog.

Java in AL2027

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

AL2027 provides several versions of Amazon Corretto to support Java-based workloads. All Java-based packages included in AL2027 are built with Amazon Corretto 25.

Amazon Corretto is a build of the Open Java Development Kit (OpenJDK) with long-term support from Amazon. Amazon Corretto is certified using the Java Technical Compatibility Kit (TCK) to ensure that it meets the Java SE standard. Amazon Corretto runs on Linux, Windows, and macOS.

There is an Amazon Corretto package available for each of Corretto 1.8.0, 11, 17, 21, and 25.

Amazon Linux supports each Amazon Corretto version in AL2027 for the same duration as the upstream Corretto version. Support ends when the Corretto version reaches end of life, or when

AL2027 reaches end of life, whichever comes first. For more information about Amazon Corretto support timelines, see the Amazon Corretto FAQs.

Node.js in AL2027

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

AL2027 includes Node.js 24. For more information about the runtime, see the Node.js website. Amazon Linux follows the upstream Node.js support schedule on the GitHub website. To check the support status of a specific Node.js version, see the Package support information page.

Amazon Linux namespaces all supported Node.js versions so that you can install them on the same system at the same time. Namespacing makes each Node.js installation unique within the file system. To do this, Amazon Linux renames key directories and files based on the runtime version. The resulting executable names look like *node-{MAJOR_VERSION}* or *npm-{MAJOR_VERSION}*.

Only one Node.js version can be active at a time. The active version provides the default directory and file names, such as *node*, *npm*, or */usr/lib/node_modules*, and points them to the currently active runtime. The *alternatives* tool provides this capability.

The default executable names are virtual and can change at any time to point to a different installed Node.js version. This flexibility lets software that uses *node* in the shebang select the desired version when invoked. When you need a specific version of Node.js, call the namespaced executable, for example *node-24*. A namespaced executable always uses the version of the runtime that its name specifies. The namespaced executables of the *npm* tool, such as *npm-24*, always use the corresponding Node.js version, regardless of the currently active runtime.

Amazon Linux distributes Node.js as several namespaced packages that begin with `nodejs{MAJOR_VERSION}`. These packages provide *node*, a compatible version of the *npm* tool, documentation, libraries, and more. For example, the `nodejs24` and `nodejs24-npm` packages provide *node* and *npm* for Node.js 24.

The *alternatives* tool provides a single command for switching between Node.js versions. By default, *alternatives* runs in auto mode and uses priorities to determine the active Node.js version.

However, you can activate any installed version at any time. All supported versions of Node.js currently have equal priority, so Amazon Linux activates the first version that you install.

Some useful examples of using *alternatives*

1. Check what *alternatives* is configured for

```
alternatives --list
```

2. Check *node*'s current configuration

```
alternatives --display node
```

3. Interactively change the Node.js version

```
alternatives --config node
```

4. Switch to manual mode and select a specific version

```
alternatives --set node /usr/bin/node-{MAJOR_VERSION}
```

5. Switch back to auto version selection mode

```
alternatives --auto node
```

Security best practices

When using Node.js and *npm* on AL2027, follow the upstream Node.js security best practices guide on the Node.js website. It covers application-level threats, including supply chain attacks. It also covers mitigations, such as using lockfiles and disabling lifecycle scripts.

TypeScript in AL2027

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

Note

This topic provides the essential information about TypeScript and its Node.js based execution environment. It also covers a typical development workflow. It explains how AL2027 packages TypeScript to deliver a consistent and reproducible development environment.

TypeScript (TS) is a programming language based on JavaScript (JS). It offers all JS features and extends them with a type system. For more information, see the TypeScript website and the TypeScript type system on the TypeScript website.

In a typical scenario, a program written in TS is first translated into JS code. Node.js then runs the result as any other regular JS program. TS terminology calls this translation process *compilation*, and calls the tool that performs it a *compiler*. The TS compiler is named *tsc*. For more information, see compiling TypeScript on the TypeScript website.

The *tsc* compiler is itself written in JS. To run, it needs a JS runtime environment, such as Node.js. Unlike some other JS runtime environments, Node.js currently provides only experimental and lightweight TS support. Full TS support, including type checking, still requires a third-party package, such as typescript on the npm website.

To get *tsc* for the Node.js runtime environment, install the `typescript` node module. You can install it with one of the package managers, typically *npm*, either globally or in a project. The officially recommended method on the TypeScript website is to install the compiler for each project. This approach ensures long-term consistency and reproducibility for your projects.

Installing the compiler globally can still be useful. A global installation provides the same version for the entire host and its JS runtime, including projects that don't have a compiler installed locally. Amazon Linux packages a compiler in exactly this way. Packages such as `nodejs24-typescript` install a compiler globally at the system level, separately for each supported Node.js version.

The *tsc* compiler doesn't directly depend on any Node.js version. Instead, it expects a certain level of runtime features. You define these features in a special file (*tsconfig.json*) through options such as target and lib on the TypeScript website. The values of these options represent a version of the ECMAScript (ES) standard, which the JS runtime environment might (or might not) support. For more information, see ECMAScript version history on Wikipedia.

Different versions of Node.js support different versions of the ES standard. The more recent the version of Node.js, the higher and more complete the supported ES standard version. If *tsconfig.json* does not exist in the root directory of a project, *tsc* uses the default set of configuration options. For a compatibility table of Node.js versions and the supported features of various ES standard versions, see node.green on the node.green website.

The *tsc* compiler has more than 100 different options that you can define in *tsconfig.json*. It also supports configuration chaining, where one file defines some configuration options and the main file then includes that file. With this approach, you can install a base configuration that is compatible with a certain version of Node.js, and then extend it with project-specific options. For more information, see Base TS Config on the GitHub website.

The base configurations for Node.js are available as node modules that you can install in a project folder using *npm*. For the source of the configuration for Node.js version 24, see node24.json on the GitHub website.

The Node.js based runtime design has a certain weakness. It supports only one version of the runtime on a host, and it requires reproducibility and consistency of all dependencies at the project level. This constraint led to a common approach for using TypeScript. You install the compiler, the TS base configuration for the current Node.js version, and all software dependencies locally, inside a project.

Globally installed node modules are expected to be CLI tools only, such as *npm*. Although *tsc* is also a CLI tool, users rarely install it globally. Global (system-wide) and local (within a project) installations of *tsc* can co-exist without issues. The two installations can also be different versions that you use independently. Run a locally installed *tsc* with the *npx* tool, which *npm* installs.

Even with a system TS compiler, you can choose the versions of the runtime components. You can switch the active Node.js version with the *alternatives* tool. You can also select a compiler version, either by installing it locally, or by installing it globally and then switching the active version with *alternatives*.

Amazon Linux packages a TS compiler in the same way as other globally installed node modules, such as *npm*, for each Node.js version. Amazon Linux namespaces packages and binaries, and includes the major version of Node.js in their names. The *alternatives* tool manages the default executable name of the compiler, *tsc*, at runtime. It points *tsc* to the Node.js version that the compiler was installed for and runs under. This selection doesn't depend on the current Node.js runtime version. You can also use the namespaced name of a compiler, for example *tsc-{MAJOR_VERSION}*, regardless of what the default *tsc* name points to.

Some useful commands for managing the active version of a TS compiler

1. Check what *alternatives* is configured for

```
alternatives --list
```

2. Check *tsc*'s current configuration

```
alternatives --display tsc
```

3. Interactively change the *tsc* version

```
alternatives --config tsc
```

4. Switch to manual mode and select a specific version

```
alternatives --set tsc /usr/bin/tsc-{MAJOR_VERSION}
```

5. Switch back to auto version selection mode

```
alternatives --auto tsc
```

Go in AL2027

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

Go is a relatively fast-moving language. Existing applications written in Go might need to adapt to new versions of the Go toolchain.

Although AL2027 will incorporate new versions of the Go toolchain during its lifetime, this will not be in lockstep with the upstream Go releases. If you want to build Go code that uses the latest features of the Go language or standard library, the Go toolchain provided in AL2027 might not be suitable.

The package name is golang. Amazon Linux does not remove previous package versions from the repositories. If you need a previous Go toolchain, you can install an earlier version from the repositories using the same mechanisms available for any RPM. Note that an earlier version does not include the bug fixes and security fixes of newer Go toolchains.

Rust in AL2027

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

AL2027 includes a Rust toolchain for building Rust applications. Because Rust releases far more often than AL2027 does, the toolchain is not always the newest release available upstream. For more information about the language, see Learn Rust on the Rust website.

The compiler package is named `rust`. The toolchain also includes `cargo`, `clippy`, `rustfmt`, and `rust-analyzer`. We do not namespace these packages. Because the package names do not include a version, you can install only one Rust toolchain at a time.

Rust toolchain versions

Rust is a fast-moving language. The Rust project publishes a new stable release about every six weeks. AL2027 publishes releases on a two-week cadence, and each new Rust release requires packaging and validation first.

The two cadences do not align. You can therefore expect the Rust toolchain in AL2027 to be either the current stable release or the release immediately before it.

The toolchain moves forward during the life of AL2027. We do not remove previous package versions from the repositories. If you need an earlier toolchain, you can install it using the same mechanisms available for any RPM. An earlier toolchain does not include the bug fixes and security fixes of newer Rust releases.

.NET in AL2027

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

With AL2027, you can use the .NET runtime and SDK, version 10.0. .NET 6.0, 8.0, and 9.0 from AL2023 are not available in AL2027 at this time.

Migrating to .NET 10

If you are moving an application from AL2023 that targets .NET 8.0 or 9.0, for more information about upgrading, see the .NET 10 migration guide on the Microsoft Learn website.

Installing .NET on AL2027

To install the .NET SDK, use the `dnf` command:

```
sudo dnf install dotnet-sdk-10.0
```

To install only the runtime without the SDK, use the following command:

```
sudo dnf install dotnet-runtime-10.0
```

PHP in AL2027

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

AL2027 currently provides version 8.5 of the PHP programming language on the PHP website. Each version is supported for the same period of time as upstream PHP.

AL2027 incorporates new versions of PHP during its lifetime. The package name is namespaced using a naming convention such as `php8.5`. Amazon Linux does not remove previous package

versions from the repositories. If you need a previous PHP version, you can install an earlier version from the repositories. Note that an earlier version does not include the bug fixes and security fixes of a newer PHP version.

Migrating from older PHP versions

Use the following migration documentation from the upstream PHP community:

- Migrating from PHP 8.4.x to PHP 8.5.x on the PHP website

PHP modules in AL2027

AL2027 includes the following packages from the PHP Extension Community Library (PECL) on the PECL website:

- `php-pecl-redis6`
- `php-pecl-apcu`
- `php-pecl-msgpack`
- `php-pecl-igbinary`
- `php-pecl-memcached`

Ruby in AL2027

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

AL2027 includes Ruby 3.4. Ruby 3.2 from AL2023 is not available. Amazon Linux follows the upstream support schedule and a support status of any Ruby version can always be checked on the Ruby website. Amazon Linux namespaces the supported Ruby version so that future versions can be installed on the same system at the same time. Namespacing ensures that each Ruby installation is unique within the file system. This is achieved by renaming key directories and files based on the runtime version. The actual executable name will look like *ruby{MAJOR.MINOR}* (e.g., `ruby3.4`). Ruby 3.4 also provides the MRI (Matz's Ruby Interpreter) namespaced binary `ruby3.4-`

`mri`, which refers to the standard C-based reference implementation of Ruby. However, only one Ruby version can be active at a time. This active version provides the default directories and file names, such as *ruby*, *gem*, or *bundle*, pointing them to the currently active runtime.

This is achieved using the capabilities of the *alternatives* tool. It is important to remember that the default executable names are virtual and can change at any time when pointing to a different installed Ruby version. This flexibility enables software that uses *ruby* in the shebang to select the desired version when invoked. However, when a specific version of Ruby is required, persistence of the version can be achieved by calling the namespaced executable (e.g., `ruby3.4`), which will always use the specified version of the runtime. Additionally, the namespaced executables of the *gem* and *bundler* tools, such as `ruby3.4-gem` or `ruby3.4-bundler`, are always associated with the corresponding Ruby version, regardless of the currently active runtime.

Ruby is distributed as several namespaced packages which begin with "`ruby{MAJOR.MINOR}`". These packages provide *ruby*, compatible versions of the *gem* and *bundler* tools, documentation, libraries, and more. For example, the core Ruby 3.4 runtime is provided by the `ruby3.4` package, which pulls in `ruby3.4-rubygems` (providing *gem*) and `ruby3.4-rubygem-bundler` (providing *bundle* and *bundler*) as dependencies.

After installing a Ruby version, the entries for companion tools may show as null in the *alternatives* configuration. This can be verified by running `alternatives --display ruby`. If entries appear as null, they must be registered manually using `alternatives --install`. For example, to register all companion tools for Ruby 3.4:

```
sudo alternatives --install /usr/bin/gem gem /usr/bin/ruby3.4-gem 34
sudo alternatives --install /usr/bin/bundle bundle /usr/bin/ruby3.4-bundle 34
sudo alternatives --install /usr/bin/bundler bundler /usr/bin/ruby3.4-bundler 34
sudo alternatives --install /usr/bin/erb erb /usr/bin/ruby3.4-erb 34
sudo alternatives --install /usr/bin/racc racc /usr/bin/ruby3.4-racc 34
sudo alternatives --install /usr/bin/rdoc rdoc /usr/bin/ruby3.4-rdoc 34
sudo alternatives --install /usr/bin/ri ri /usr/bin/ruby3.4-ri 34
```

The priority value (e.g., 34 for Ruby 3.4) should match the priority used in the main *ruby* alternative entry. Once registered, the companion tools will be managed automatically alongside the *ruby* alternative.

The *alternatives* tool provides a single command for switching between installed Ruby versions. By default, *alternatives* is configured to be in auto mode, which uses priorities to determine the currently active Ruby version. However, you can activate any installed version at any time.

Some useful examples of using *alternatives*

1. Check what *alternatives* is configured for

```
alternatives --list
```

2. Check *ruby*'s current configuration

```
alternatives --display ruby
```

3. Interactively change the Ruby version

```
alternatives --config ruby
```

4. Switch to manual mode and select a specific version

```
alternatives --set ruby /usr/bin/ruby{MAJOR.MINOR}
```

5. Switch back to auto version selection mode

```
alternatives --auto ruby
```

Swift in AL2027

AL2027 currently provides the Swift (website) runtime and SDK.

In AL2027, the Swift SDK and runtime are split into separate packages. The `swiftlang-lib` package contains the runtime libraries, and the `swiftlang` package contains the full SDK. This allows you to run binaries compiled with the Swift SDK on production hosts without requiring the full SDK to be installed. At this time, different versions of Swift are not ABI compatible. When updating Swift, applications will need to be rebuilt with the new version before they can be deployed.

> **Note**
>
> Like Rust, Swift versions will follow the latest stable version from the Swift project. Packages will not be namespaced, and only one version will be supported at a time.

Some notes on using Swift on AL2027

- **C++ interoperability** – The system's package manager `clang` may not work with Swift. To resolve this, add `swift-clang` to your PATH or use alternatives to set `clang` to `swift-clang`.
- **Debugging Swift code** – The system LLDB cannot debug Swift sources. Use the LLDB included with the Swift package instead by adding the Swift installation directory to your PATH or using alternatives to set `lldb` to `swift-lldb`.

C, C++, and Fortran in AL2027

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

AL2027 includes both the GNU Compiler Collection (GCC) and the Clang frontend for LLVM (Low Level Virtual Machine).

AL2027 includes GCC 16.1 as the default compiler with the C (`gcc`), C++ (`g++`), and Fortran (`gfortran`) frontends. This is a significant upgrade from GCC 11.5 in AL2023, which is not available in AL2027. AL2027 also includes LLVM/Clang 22.

GCC 16.1 is currently the only GCC version in AL2027. AL2027 does not currently provide a second GCC version.

The major version of GCC will remain constant throughout the life time of AL2027. Minor releases bring bug fixes and might be included in AL2027 releases. Other bug, performance, and security fixes might be backported to the major version of GCC that ships in AL2027.

AL2027 does not enable the Ada (`gnat`), Go (`gcc-go`), Objective-C, or Objective-C++ frontends.

The default compiler flags that AL2027 RPMs are built with include optimization and hardening flags. To build your own code with GCC, we recommend you include optimization and hardening flags.

Note

The GCC 11 to 16 upgrade is the largest toolchain change in AL2027. Code compiled on AL2023 will run on AL2027, but recompilation might expose new warnings or errors due to stricter defaults in GCC 16.

Note

When `gcc --version` is invoked, a version string such as `gcc (GCC) 16.1.1 ... (Red Hat 16.1.1-1)` is displayed. `Red Hat` refers to the GCC vendor branch that the Amazon Linux GCC package is based upon. According to the bug report URL shown by `gcc --help`, all bug reports and support requests should be directed to Amazon Linux.
For more information about some of the long-term changes in this vendor branch, such as the `__GNUC_RH_RELEASE__` macro, see Fedora package sources.

For more information on the core toolchain, see Core toolchain packages glibc, gcc, binutils.

For more information on AL2027 and its relationship to other Linux distributions, see Relationship to Fedora.

Topics

- LLVM/Clang 22
- Language standard versions comparison

LLVM/Clang 22

AL2027 includes LLVM/Clang 22, which provides the Clang C and C++ compilers along with the broader LLVM toolchain.

To install Clang and related tools, use the following command:

```
sudo dnf install clang lld lldb
```

The Clang compilers are invoked with the following commands:

- `clang` - C compiler

- `clang++` - C++ compiler

Example usage:

```
clang -o myprogram myprogram.c
clang++ -o mycppprogram mycppprogram.cpp
```

You can verify the installed version by running:

```
clang --version
```

> **Note**
>
> In AL2027, LLVM/Clang 22 is built from a single unified SRPM. The separate `clang`, `lld`, and `lldb` source packages no longer exist. This change affects only how the software is packaged at the source level; the installation commands and the names of the installed binary packages are unchanged.

Language standard versions comparison

The following table compares the default language standard versions across different Amazon Linux versions and GCC compiler versions:

Amazon Linux Version	C Standard (Default)	C++ Standard (Default)	Fortran Standard
AL2023 with GCC 11 (default)	C17/C18 (201710L)	C++17 (201703L)	Fortran 2008
AL2023 with GCC 14 (optional)	C17/C18 (201710L)	C++17 (201703L)	Fortran 2018
AL2027 with GCC 16.1 (default)	C23 (202311L)	C++20 (202002L)	Fortran 2018

Key improvements by GCC version:

- **GCC 16.1 vs GCC 11:** Upgraded the default C standard to C23 and the default C++ standard to C++20, added full C23 support and enhanced C++23 support, introduced initial C++26 features, and improved optimization, diagnostics, and standards compliance. Because the default C standard moved to C23 and the default C++ standard moved to C++20, recompiling AL2023 sources might surface new warnings or errors.

Supported language standards:

- **C Standards:** The GCC versions in the preceding table support C90, C99, C11, and C17/C18. GCC 16.1 defaults to C23 and provides full C23 support, along with initial support for later draft C standards.
- **C++ Standards:** The GCC versions in the preceding table support C++98, C++03, C++11, C++14, C++17, and C++20. GCC 16.1 defaults to C++20, provides enhanced C++23 support, and includes initial C++26 features.
- **Fortran Standards:** GCC 16.1 provides broad Fortran 2018 support, with ongoing work toward Fortran 2023 features.

> **Note**
>
> You can select a specific language standard by using the `-std=` flag (for example, `-std=c17`, `-std=c++20`, or `-std=f2008`) to override the compiler default and target a particular standard when building your code.

Perl in AL2027

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

The default Perl version in AL2027 is 5.42, and it stays at 5.42 for the life of the release. The package name is `perl`, and we do not namespace it. For more information about the language, see Learn Perl on the Perl website.

Perl 5.42 is the only Perl version that AL2027 retains. We do not move to a later feature release, such as 5.44. Updates cover 5.42 patch releases and security fixes only. You can therefore plan on 5.42 throughout the life of your deployment.

Perl modules in AL2027

You can install many Perl modules as RPMs from the AL2027 repositories. These package names begin with `perl-`. We do not aim to package every module available on CPAN.

For the modules that we do package, stability and reliability take priority over the newest available version. We do not track the latest release of each module. However, module versions are not locked. A module can move to a newer version after we evaluate the compatibility impact on the packages that depend on it.

Many packaged modules are dependencies of other operating system packages. We therefore prioritize security patches for those modules over feature updates.

AL2027 reserved users and groups

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

AL2027 pre-allocates certain users and groups, both when an image is provisioned and when specific packages are installed. This section lists the user and group names with their associated UIDs and GIDs. Refer to this list to avoid conflicts when you create users and groups.

AL2027 packages declare their users and groups with `systemd-sysusers` configuration. Some users and groups have a static ID that is the same on every AL2027 instance. Others get a dynamic ID. `systemd-sysusers` allocates that ID from the system range (201-999) when you install the owning package. A name can have a reserved ID in one table and a dynamic ID in the other. For example, the `systemtap-runtime` package reserves its three group IDs and creates its users with dynamic IDs. For a dynamic user or group, the name is reserved but the numeric ID varies between instances. Regular user accounts start at UID and GID 1000. For example, the `ec2-user` account created when an Amazon EC2 instance is provisioned uses UID and GID 1000.

Topics

- List of AL2027 reserved users
- List of AL2027 reserved groups

List of AL2027 reserved users

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

The following tables list the user names that AL2027 packages reserve, as of the AL2027 public preview. A numeric UID is listed when the user account is created with a static UID. A user account

marked *dynamic* is allocated a UID from the system range (201-999) when the owning package is installed, so the number varies between instances. The list grows as packages are added to AL2027.

Listed by UID

User name	UID
root	0
bin	1
daemon	2
adm	3
lp	4
sync	5
shutdown	6
halt	7
mail	8
operator	11
games	12
ftp	14
squid	23
named	25
postgres	26
mysql	27
rpcuser	29
rpc	32

User name	UID
gdm	42
mailnull	47
apache	48
smmsp	51
tomcat	53
ldap	55
tss	59
nslcd	65
avahi	70
tcpdump	72
sshd	74
radvd	75
dbus	81
postfix	89
radiusd	95
dovecot	97
polkitd	114
stap-server	155
stapunpriv	159
avahi-autoipd	170

User name	UID
sanlock	179
haproxy	188
hacluster	189
systemd-network	192
systemd-resolve	193
ec2-user	1000
nobody	65534
chrony	dynamic
clamilt	dynamic
clamscan	dynamic
clamupdate	dynamic
colord	dynamic
debuginfod	dynamic
dnsmasq	dynamic
dovenull	dynamic
ec2-instance-connect	dynamic
flatpak	dynamic
geoclue	dynamic
gnome-remote-desktop	dynamic
libstoragemgmt	dynamic

User name	UID
memcached	dynamic
munge	dynamic
networkflowmonitor	dynamic
nfsnobody	dynamic
nginx	dynamic
ods	dynamic
opendkim	dynamic
openvpn	dynamic
pcscd	dynamic
pesign	dynamic
pipewire	dynamic
rtkit	dynamic
saslauth	dynamic
sphinx	dynamic
sssd	dynamic
stapdev	dynamic
stapsys	dynamic
stapusr	dynamic
systemd-coredump	dynamic
systemd-journal-remote	dynamic

User name	UID
systemd-oom	dynamic
systemd-timesync	dynamic
unbound	dynamic
uuidd	dynamic
valkey	dynamic
wsdd	dynamic

Listed by name

User name	UID
adm	3
apache	48
avahi	70
avahi-autoipd	170
bin	1
chrony	dynamic
clamilt	dynamic
clamscan	dynamic
clamupdate	dynamic
colord	dynamic
daemon	2

User name	UID
dbus	81
debuginfod	dynamic
dnsmasq	dynamic
dovecot	97
dovenull	dynamic
ec2-instance-connect	dynamic
ec2-user	1000
flatpak	dynamic
ftp	14
games	12
gdm	42
geoclue	dynamic
gnome-remote-desktop	dynamic
hacluster	189
halt	7
haproxy	188
ldap	55
libstoragemgmt	dynamic
lp	4
mail	8

User name	UID
mailnull	47
memcached	dynamic
munge	dynamic
mysql	27
named	25
networkflowmonitor	dynamic
nfsnobody	dynamic
nginx	dynamic
nobody	65534
nslcd	65
ods	dynamic
opendkim	dynamic
openvpn	dynamic
operator	11
pcscd	dynamic
pesign	dynamic
pipewire	dynamic
polkitd	114
postfix	89
postgres	26

User name	UID
radiusd	95
radvd	75
root	0
rpc	32
rpcuser	29
rtkit	dynamic
sanlock	179
saslauth	dynamic
shutdown	6
smmsp	51
sphinx	dynamic
squid	23
sshd	74
sssd	dynamic
stap-server	155
stapdev	dynamic
stapsys	dynamic
stapunpriv	159
stapusr	dynamic
sync	5

User name	UID
systemd-coredump	dynamic
systemd-journal-remote	dynamic
systemd-network	192
systemd-oom	dynamic
systemd-resolve	193
systemd-timesync	dynamic
tcpdump	72
tomcat	53
tss	59
unbound	dynamic
uuidd	dynamic
valkey	dynamic
wsdd	dynamic

List of AL2027 reserved groups

AL2027 Preview

AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

The following tables list the group names that AL2027 packages reserve, as of the AL2027 public preview. A numeric GID is listed when the group is created with a static GID. A group marked *dynamic* is allocated a GID from the system range (201-999) when the owning package is installed, so the number varies between instances. The list grows as packages are added to AL2027.

Listed by GID

Group name	GID
root	0
bin	1
daemon	2
sys	3
adm	4
tty	5
disk	6
lp	7
mem	8
kmem	9
wheel	10
cdrom	11
mail	12
man	15
dialout	18
floppy	19
games	20
utmp	22
squid	23

Group name	GID
named	25
postgres	26
rpcuser	29
rpc	32
tape	33
utempter	35
kvm	36
video	39
gdm	42
mailnull	47
apache	48
ftp	50
smmsp	51
tomcat	53
lock	54
ldap	55
tss	59
audio	63
avahi	70
tcpdump	72

Group name	GID
sshd	74
radvd	75
saslauth	76
dbus	81
screen	84
wbpriv	88
postfix	89
postdrop	90
radiusd	95
dovecot	97
users	100
clock	103
input	104
render	105
sgx	106
polkitd	114
mock	135
stap-server	155
stapusr	156
stapsys	157

Group name	GID
stapdev	158
stapunpriv	159
avahi-autoipd	170
sanlock	179
haproxy	188
haclient	189
systemd-journal	190
systemd-network	192
systemd-resolve	193
ec2-user	1000
nobody	65534
chrony	dynamic
clamilt	dynamic
clamscan	dynamic
clamupdate	dynamic
colord	dynamic
debuginfod	dynamic
dnsmasq	dynamic
dovenull	dynamic
ec2-instance-connect	dynamic

Group name	GID
empower	dynamic
flatpak	dynamic
geoclue	dynamic
gnome-remote-desktop	dynamic
hacluster	dynamic
halt	dynamic
libstoragemgmt	dynamic
memcached	dynamic
munge	dynamic
mysql	dynamic
nagios	dynamic
networkflowmonitor-group	dynamic
nfsnobody	dynamic
nginx	dynamic
nslcd	dynamic
ods	dynamic
opendkim	dynamic
openvpn	dynamic
operator	dynamic
pcscd	dynamic

Group name	GID
pesign	dynamic
pipewire	dynamic
plocate	dynamic
printadmin	dynamic
rtkit	dynamic
shutdown	dynamic
sphinx	dynamic
sssd	dynamic
sync	dynamic
systemd-coredump	dynamic
systemd-journal-remote	dynamic
systemd-oom	dynamic
systemd-timesync	dynamic
tracing	dynamic
unbound	dynamic
usbmon	dynamic
usershares	dynamic
uuidd	dynamic
valkey	dynamic
virusgroup	dynamic

Group name	GID
wireshark	dynamic
wsdd	dynamic

Listed by name

Group name	GID
adm	4
apache	48
audio	63
avahi	70
avahi-autoipd	170
bin	1
cdrom	11
chrony	dynamic
clamilt	dynamic
clamscan	dynamic
clamupdate	dynamic
clock	103
colord	dynamic
daemon	2
dbus	81

Group name	GID
debuginfod	dynamic
dialout	18
disk	6
dnsmasq	dynamic
dovecot	97
dovenull	dynamic
ec2-instance-connect	dynamic
ec2-user	1000
empower	dynamic
flatpak	dynamic
floppy	19
ftp	50
games	20
gdm	42
geoclue	dynamic
gnome-remote-desktop	dynamic
haclient	189
hacluster	dynamic
halt	dynamic
haproxy	188

Group name	GID
input	104
kmem	9
kvm	36
ldap	55
libstoragemgmt	dynamic
lock	54
lp	7
mail	12
mailnull	47
man	15
mem	8
memcached	dynamic
mock	135
munge	dynamic
mysql	dynamic
nagios	dynamic
named	25
networkflowmonitor-group	dynamic
nfsnobody	dynamic
nginx	dynamic

Group name	GID
nobody	65534
nslcd	dynamic
ods	dynamic
opendkim	dynamic
openvpn	dynamic
operator	dynamic
pcscd	dynamic
pesign	dynamic
pipewire	dynamic
plocate	dynamic
polkitd	114
postdrop	90
postfix	89
postgres	26
printadmin	dynamic
radiusd	95
radvd	75
render	105
root	0
rpc	32

Group name	GID
rpcuser	29
rtkit	dynamic
sanlock	179
saslauth	76
screen	84
sgx	106
shutdown	dynamic
smmsp	51
sphinx	dynamic
squid	23
sshd	74
sssd	dynamic
stap-server	155
stapdev	158
stapsys	157
stapunpriv	159
stapusr	156
sync	dynamic
sys	3
systemd-coredump	dynamic

Group name	GID
systemd-journal	190
systemd-journal-remote	dynamic
systemd-network	192
systemd-oom	dynamic
systemd-resolve	193
systemd-timesync	dynamic
tape	33
tcpdump	72
tomcat	53
tracing	dynamic
tss	59
tty	5
unbound	dynamic
usbmon	dynamic
users	100
usershares	dynamic
utempter	35
utmp	22
uuidd	dynamic
valkey	dynamic

Group name	GID
video	39
virusgroup	dynamic
wbpriv	88
wheel	10
wireshark	dynamic
wsdd	dynamic

Running applications on AL2027

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

This section covers methods for running applications on Amazon Linux 2027 (AL2027), including managing when they start (and restart), and controlling resource usage.

Topics

- Limiting process resource usage in AL2027 using systemd
- Limiting process resource usage in AL2027 using cgroups

Limiting process resource usage in AL2027 using systemd

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

On Amazon Linux 2027 (AL2027), we recommend using `systemd` to control what resources can be used by processes, or groups of processes. Using `systemd` is a powerful and easy to use replacement for either manipulating `cgroups` manually, or using utilities such as `cpulimit`, which was previously only available for Amazon Linux in a third party repository.

For comprehensive information, see the upstream `systemd` documentation for systemd.resource-control on the freedesktop.org website, or the man page for `systemd.resource-control` on an AL2027 instance.

The following examples will use the `stress-ng` CPU stress test (from the `stress-ng` package) to simulate a CPU heavy application, and `memcached` to simulate a memory heavy application.

The following examples cover placing a CPU limit on a one-off command and a memory limit on a service. Most resource constraints that `systemd` offers can be used in any place that `systemd` will

run a process, and multiple can be used at the same time. The following examples are limited to a single constraint for illustrative purposes.

Resource control with `systemd-run` for running one-off commands

While commonly associated with system services, `systemd` can also be used by non-root users to run services, schedule timers, or run one-off processes. In the following example, we are going to use `stress-ng` as our example application. In the first example, we will run it using `systemd-run` in the `ec2-user` default account, and in the second example we will place limits on its CPU usage.

Example Use `systemd-run` on the command line to run a process, not limiting resource usage

1. Ensure the `stress-ng` package is installed, as we are going to use it for our example.

   ```
   [ec2-user ~]$ sudo dnf install -y stress-ng
   ```

2. Use `systemd-run` to execute a 10 second CPU stress test without limiting how much CPU it can use.

   ```
   [ec2-user ~]$ systemd-run --user --tty --wait stress-ng --cpu 1 --timeout 10
   Running as unit: run-p66001-i57336.service
   Press ^] three times within 1s to disconnect TTY.
   stress-ng: info:  [66004] setting to a 10 secs run per stressor
   stress-ng: info:  [66004] dispatching hogs: 1 cpu
   stress-ng: info:  [66004] successful run completed in 10.00 secs
   Finished with result: success
   Main processes terminated with: code=exited, status=0/SUCCESS
   Service runtime: 10.012s
   CPU time consumed: 9.987s
   Memory peak: 3M (swap: 0B)
   ```

   The `--user` option tells `systemd-run` to execute the command as the user we are logged in as, the `--tty` option means a TTY is attached, and `--wait` means to wait until the service is finished. The `--property` command line option can be used to pass `systemd-run` settings that could be configured in a `systemd.unit` configuration file.

When instructed to place load on the CPU, the `stress-ng` program will use all available CPU time to perform its test for the duration you ask it to run. For a real-world application, it may be desirable to place a limit on total run-time of a process. In the following example, we will ask

`stress-ng` to run for a longer time than the maximum duration restriction we place on it using `systemd-run`.

Example Use `systemd-run` on the command line to run a process, limiting CPU usage to 1 second

1. Ensure the `stress-ng` is installed to run this example.
2. The `LimitCPU` property is the equivalent of `ulimit -t` which will limit the maximum amount of time on the CPU this process will be allowed to use. In this case, since we are asking for a 10 second stress run, and we are limiting the CPU usage to 1 second, the command receives a `SIGXCPU` signal after roughly 1 second of CPU time and stops early.

```
[ec2-user ~]$ systemd-run --user --tty --wait --property=LimitCPU=1 stress-ng --cpu
 1 --timeout 10
Running as unit: run-p68466-i67823.service
Press ^] three times within 1s to disconnect TTY.
stress-ng: info:  [68469] setting to a 10 secs run per stressor
stress-ng: info:  [68469] dispatching hogs: 1 cpu
stress-ng: warn:  [68469] metrics-check: all bogo-op counters are zero, data may be
 incorrect
stress-ng: info:  [68469] successful run completed in 1.01 sec
Finished with result: success
Main processes terminated with: code=exited, status=0/SUCCESS
Service runtime: 1.020s
CPU time consumed: 1.017s
Memory peak: 3.1M (swap: 0B)
```

   The service runtime and CPU time consumed confirm that the `SIGXCPU` signal stopped the process at roughly 1 second, instead of allowing the full 10 second run to complete.

More commonly, you may want to restrict the percentage of CPU time that can be consumed by a particular process. In the following example, we will restrict the percentage of CPU time that can be consumed by `stress-ng`. For a real-world service, it may be desirable to limit the maximum percentage of CPU time a background process can consume in order to leave resources free for the process serving user requests.

Example Use `systemd-run` to limit a process to 10% of CPU time on one CPU

1. Ensure the `stress-ng` is installed to run this example.

2. We are going to use the `CPUQuota` property to tell `systemd-run` to constrain CPU usage for the command we are going to run. We are not limiting the amount of time the process can run for, just how much CPU it can use.

```
[ec2-user ~]$ systemd-run --user --tty --wait --property=CPUQuota=10% stress-ng --cpu 1 --timeout 10
Running as unit: run-p66058-i66947.service
Press ^] three times within 1s to disconnect TTY.
stress-ng: info:  [66059] setting to a 10 secs run per stressor
stress-ng: info:  [66059] dispatching hogs: 1 cpu
stress-ng: info:  [66059] successful run completed in 10.03 secs
Finished with result: success
Main processes terminated with: code=exited, status=0/SUCCESS
Service runtime: 10.069s
CPU time consumed: 1.019s
Memory peak: 3.1M (swap: 0B)
```

   Note how the CPU accounting tells us that while the service ran for 10 seconds, it only consumed 1 second of actual CPU time.

There are many ways to configure `systemd` to limit resource usage for CPU, memory, networking, and IO. See the upstream `systemd` documentation for systemd.resource-control on the freedesktop.org website, or the man page for `systemd.resource-control` on an AL2027 instance for comprehensive documentation.

Behind the scenes, `systemd` is using features of the Linux kernel such as `cgroups` to implement these limits while avoiding the need for you to configure them by hand. AL2027 uses cgroup v2. The Linux Kernel documentation for `cgroup-v2` on the kernel.org website contains extensive details about how `cgroups` work.

Resource control in a `systemd` service

There are several parameters that can be added to the `[Service]` section of `systemd` services to control system resource usage. These include both hard and soft limits. For the exact behavior of each option, refer to the upstream `systemd` documentation for systemd.resource-control on the freedesktop.org website, or the man page for `systemd.resource-control` on an AL2027 instance.

Commonly used limits are `MemoryHigh` to specify a throttling limit on memory usage, `MemoryMax` to set a hard upper limit (which, once reached, the OOM Killer is invoked), and `CPUQuota` (as illustrated in the previous section). It is also possible to configure weights and priorities rather than fixed numbers.

Example Using `systemd` to set memory usage limits on services

In this example we will set a hard memory usage limit for `memcached`, a simple key-value cache, and show how the OOM Killer is invoked for that service rather than the whole system.

1. First, we need to install the packages required for this example.

```
[ec2-user ~]$ sudo dnf install -y memcached libmemcached-awesome-tools
```

2. Enable the `memcached.service` and then start the service so that `memcached` is running.

```
[ec2-user ~]$ sudo systemctl enable memcached.service
Created symlink /etc/systemd/system/multi-user.target.wants/memcached.service # /
usr/lib/systemd/system/memcached.service.
[ec2-user ~]$ sudo systemctl start memcached.service
```

3. Check that `memcached.service` is running.

```
[ec2-user ~]$ sudo systemctl status memcached.service
# memcached.service - memcached daemon
     Loaded: loaded (/usr/lib/systemd/system/memcached.service; enabled; preset:
 disabled)
    Drop-In: /usr/lib/systemd/system/service.d
             ##10-timeout-abort.conf
     Active: active (running) since Fri 2026-08-28 04:29:12 UTC; 234ms ago
 Invocation: 14415487cabc478c9262147770c975d6
   Main PID: 66798 (memcached)
      Tasks: 10 (limit: 18904)
     Memory: 1.9M (peak: 3.4M)
        CPU: 25ms
     CGroup: /system.slice/memcached.service
             ##66798 /usr/bin/memcached -p 11211 -u memcached -m 64 -c 1024 -l
 127.0.0.1,::1
```

4. Now that `memcached` is installed and running, we can observe that it functions by inserting some random data into the cache.

In `/etc/sysconfig/memcached` the `CACHESIZE` variable is set to 64 by default, meaning 64 megabytes. By inserting more data into the cache than the maximum cache size, we can see that we fill the cache and some items are evicted using `memcached-tool`, and that the `memcached.service` is using around 64MB of memory.

```
[ec2-user ~]$ for i in $(seq 1 150); do dd if=/dev/random of=$i bs=512k count=1;
 memcp -s localhost $i; done
[ec2-user ~]$ memcached-tool localhost display
  #  Item_Size  Max_age   Pages   Count   Full?  Evicted Evict_Time OOM
  2     120B         0s       1       0      no        0        0    0
 39   512.0K         1s      63     126     yes       24        1    0
[ec2-user ~]$ sudo systemctl status memcached.service
# memcached.service - memcached daemon
     Loaded: loaded (/usr/lib/systemd/system/memcached.service; enabled; preset:
 disabled)
    Drop-In: /usr/lib/systemd/system/service.d
             ##10-timeout-abort.conf
     Active: active (running) since Fri 2026-08-28 04:37:08 UTC; 2s ago
 Invocation: f0a6ec3dbe06466c92f762fa3a1f59d6
   Main PID: 68690 (memcached)
      Tasks: 10 (limit: 18904)
     Memory: 66.8M (peak: 67.7M)
        CPU: 133ms
     CGroup: /system.slice/memcached.service
             ##68690 /usr/bin/memcached -p 11211 -u memcached -m 64 -c 1024 -l
 127.0.0.1,::1
```

5. Use the `MemoryMax` property to set a hard limit for the `memcached.service` where, if hit, the OOM Killer will be invoked. Additional options can be set for the service by adding them to an override file. This can be done either by directly editing the `/etc/systemd/system/memcached.service.d/override.conf` file, or interactively using the `edit` command of `systemctl`.

```
[ec2-user ~]$ sudo systemctl edit memcached.service
```

Add the following to the override to set a hard limit of 32MB of memory for the service.

```
[Service]
MemoryMax=32M
```

6. Tell `systemd` to reload its configuration.

```
[ec2-user ~]$ sudo systemctl daemon-reload
```

7. Observe that the `memcached.service` is now running with a memory limit of 32MB.

```
[ec2-user ~]$ sudo systemctl status memcached.service
# memcached.service - memcached daemon
     Loaded: loaded (/usr/lib/systemd/system/memcached.service; enabled; preset:
 disabled)
    Drop-In: /usr/lib/systemd/system/service.d
             ##10-timeout-abort.conf
             /etc/systemd/system/memcached.service.d
             ##override.conf
     Active: active (running) since Fri 2026-08-28 04:37:35 UTC; 271ms ago
 Invocation: 2980a4cea0af4014b452c5d011c1b33e
   Main PID: 69347 (memcached)
      Tasks: 10 (limit: 18904)
     Memory: 1.9M (max: 32M, available: 30M, peak: 3M)
        CPU: 21ms
     CGroup: /system.slice/memcached.service
             ##69347 /usr/bin/memcached -p 11211 -u memcached -m 64 -c 1024 -l
 127.0.0.1,::1

Aug 28 04:37:35 ip-172-31-37-206.us-west-2.compute.internal systemd[1]: Started
 memcached.service - memcached daemon.
```

8. The service will function normally while using less than 32MB of memory, which we can check by loading less than 32MB of random data into the cache, and then checking the status of the service.

```
[ec2-user ~]$ for i in $(seq 1 30); do dd if=/dev/random of=$i bs=512k count=1;
 memcp -s localhost $i; done
```

```
[ec2-user ~]$ sudo systemctl status memcached.service
# memcached.service - memcached daemon
     Loaded: loaded (/usr/lib/systemd/system/memcached.service; enabled; preset:
 disabled)
    Drop-In: /usr/lib/systemd/system/service.d
             ##10-timeout-abort.conf
             /etc/systemd/system/memcached.service.d
             ##override.conf
```

```
     Active: active (running) since Fri 2026-08-28 04:37:35 UTC; 7s ago
 Invocation: 2980a4cea0af4014b452c5d011c1b33e
   Main PID: 69347 (memcached)
      Tasks: 10 (limit: 18904)
     Memory: 18.3M (max: 32M, available: 13.6M, peak: 19.3M)
        CPU: 47ms
     CGroup: /system.slice/memcached.service
             ##69347 /usr/bin/memcached -p 11211 -u memcached -m 64 -c 1024 -l
 127.0.0.1,::1

Aug 28 04:37:35 ip-172-31-37-206.us-west-2.compute.internal systemd[1]: Started
 memcached.service - memcached daemon.
```

9. We can now make `memcached` to use more than 32MB of memory by attempting to use the full 64MB of cache that the default `memcached` configuration is.

```
[ec2-user ~]$ for i in $(seq 1 150); do dd if=/dev/random of=$i bs=512k count=1;
 memcp -s localhost $i; done
```

You will observe that at some point during the above command there are connection errors to the `memcached` server. This is because the OOM Killer has killed the process due to the restriction we placed on it. The rest of the system will function as normal, and no other processes will be considered by the OOM Killer, as it is only the `memcached.service` that we have restricted.

```
[ec2-user ~]$ sudo systemctl status memcached.service
× memcached.service - memcached daemon
     Loaded: loaded (/usr/lib/systemd/system/memcached.service; enabled; preset:
 disabled)
    Drop-In: /usr/lib/systemd/system/service.d
             ##10-timeout-abort.conf
             /etc/systemd/system/memcached.service.d
             ##override.conf
     Active: failed (Result: oom-kill) since Fri 2026-08-28 04:37:50 UTC; 976ms ago
   Duration: 14.899s
 Invocation: 2980a4cea0af4014b452c5d011c1b33e
    Process: 69347 ExecStart=/usr/bin/memcached -p ${PORT} -u ${USER} -m
 ${CACHESIZE} -c ${MAXCONN} $OPTIONS (code=killed, signal=KILL)
   Main PID: 69347 (code=killed, signal=KILL)
   Mem peak: 32M
        CPU: 149ms
```

```
Aug 28 04:37:35 ip-172-31-37-206.us-west-2.compute.internal systemd[1]: Started
 memcached.service - memcached daemon.
Aug 28 04:37:50 ip-172-31-37-206.us-west-2.compute.internal systemd[1]:
 memcached.service: The kernel OOM killer killed some processes in this unit.
Aug 28 04:37:50 ip-172-31-37-206.us-west-2.compute.internal systemd[1]:
 memcached.service: Main process exited, code=killed, status=9/KILL
Aug 28 04:37:50 ip-172-31-37-206.us-west-2.compute.internal systemd[1]:
 memcached.service: Failed with result 'oom-kill'.
```

Limiting process resource usage in AL2027 using cgroups

> **⚠ AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

While it is recommended to use Resource control with systemd, this section covers basic usage of the base `libcgroup-tools` utilities to limit CPU and memory usage of processes. Both methods are alternatives to using the `cpulimit` utility.

The following example covers running the `stress-ng` stress test (from the `stress-ng` package) while limiting its CPU and memory usage using utilities from the `libcgroup-tools` package, and the tunables in `sysfs`.

Use `libcgroup-tools` on the command line to limit resource usage

1. Install the `libcgroup-tools` package.

   ```
   [ec2-user ~]$ sudo dnf install libcgroup-tools
   ```

2. Create a `cgroup` with the `memory` and `cpu` controllers, and give it a name (`our-example-limits`). Using the `-a` and `-t` options to allow the `ec2-user` user to control the tunables of the `cgroup`.

   ```
   [ec2-user ~]$ sudo cgcreate -a ec2-user -t ec2-user -g memory,cpu:our-example-
   limits
   ```

There is now a `/sys/fs/cgroup/our-example-limits/` directory that contains files that can be used to control each tunable.

3. Limit memory usage of all processes in our `cgroup` to 100 million bytes.

```
[ec2-user ~]$ echo 100000000 > /sys/fs/cgroup/our-example-limits/memory.max
```

4. Limit CPU usage of all processes in our `cgroup` to 10%. The format of the `cpu.max` file is `$MAX $PERIOD`, limiting the group to consuming `$MAX` for every `$PERIOD`.

```
[ec2-user ~]$ echo 10000 100000 > /sys/fs/cgroup/our-example-limits/cpu.max
```

5. The following example runs `stress-ng` (which can be installed by running `dnf install -y stress-ng`) in the `our-example-limits` cgroup. While the `stress-ng` command is running, you can observe using `top` that it is limited to 10% of CPU time.

```
[ec2-user ~]$ sudo cgexec -g memory,cpu:our-example-limits stress-ng --cpu 1
```

6. Clean up by removing the cgroup.

```
[ec2-user ~]$ sudo cgdelete -g memory,cpu:our-example-limits
```

The Linux Kernel documentation for `cgroup-v2` on the kernel.org website contains extensive details about how they work. The documentation of the cpu and memory controllers, both on the kernel.org website, covers the details of how to use each tunable option.

List of codecs available in AL2027

AL2027 Preview

AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

AL2027 provides a selection of multimedia codecs and media processing libraries through its standard repositories. This page provides an overview of the codecs and their typical use cases.

Important

The use and distribution of codecs included in Amazon Linux may require that you obtain license rights from third parties, including owners or licensors of certain third party audio and video formats. You are solely responsible for obtaining these licenses and paying any necessary royalties or fees.

Codec	Description
bluez	The official Linux Bluetooth protocol stack, including native support for the SBC codec used for Bluetooth audio streaming
dav1d	An open source implementation of the AV1 video decoder focused on speed, efficiency, and portability. It provides high-performance AV1 decoding for a wide range of platforms and applications.
exempi	A library for parsing, manipulating, and serializing XMP (Extensible Metadata Platform) metadata embedded in media files
fdk-aac-free	An open-source implementation of the AAC (Advanced Audio Coding) standard, providing

Codec	Description
	high-quality audio encoding as an alternative to MP3, commonly used in streaming and file storage
flac	A free and open-source lossless audio codec that compresses audio without losing any data or quality, commonly used for high-quality audio storage
jasper	An open source implementation of the JPEG-2000 image compression standard for encoding and decoding JPEG-2000 images
libde265	An open source implementation of the H.265/HEVC video decoder. It provides efficient and portable decoding of HEVC video streams for use in multimedia applications and frameworks.
libheif	An open source library for reading and writing HEIF and AVIF image files. It provides efficient encoding, decoding, and conversion of images and image sequences using modern compression formats like HEVC and AV1.
libvpx	The reference implementation of the VP8 and VP9 video codecs, widely used for web video such as WebM
libwebp	A library for encoding and decoding images in the WebP format, which provides efficient lossy and lossless image compression for the web

Codec	Description
mpg123	A high-performance decoder for MPEG audio streams, including MP3 files. It provides fast and accurate audio decoding for playback, streaming, and audio processing applications.
openexr	A library for the OpenEXR high dynamic-range (HDR) image format, commonly used in visual effects and professional imaging
openh264	An open source implementation of the H.264 video codec. It provides encoding and decoding support for H.264 video used in streaming, conferencing, and multimedia applications.
openjpeg2	An open source implementation of the JPEG-2000 image codec, providing encoding and decoding of JPEG-2000 images
opus	A highly versatile and efficient audio codec designed for real-time streaming, offering low latency and support for a wide range of audio applications, including VoIP and music streaming
sbc	The reference implementation of the SBC (low-complexity subband) audio codec used for Bluetooth audio streaming
svt-av1	An open source, high-performance implement ation of the AV1 video encoder. It provides scalable encoding of AV1 video, optimized for modern CPUs and parallel processing.

Codec	Description
x265	An open source H.265/HEVC video encoder focused on high compression efficiency and performance across a wide variety of hardware platforms

Security and Compliance in AL2027

> **Important**
>
> If you want to report a vulnerability or have a security concern regarding AWS cloud services or open source projects, contact AWS Security using the Vulnerability Reporting page

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from a data center and network architecture that is built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS Compliance Programs. To learn about the compliance programs that apply to Amazon Linux, see AWS Services in Scope by Compliance Program.
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations.

AL2027 includes several security enhancements over AL2023, see: Security updates and features

Topics

- Amazon Linux Security advisories for AL2027
- Listing applicable advisories

- Applying security updates in-place
- Setting SELinux modes for AL2027
- Post-Quantum Cryptography (PQC) on AL2027
- Enable FIPS Mode on AL2027
- Enable FIPS Mode in an AL2027 Container
- Swap OpenSSL FIPS providers on AL2027
- AL2027 kernel hardening
- Repository metadata signing in AL2027
- Security patching during preview
- FIPS validation
- UEFI Secure Boot on AL2027

Amazon Linux Security advisories for AL2027

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

Although we work hard to make Amazon Linux secure, at times there will be security issues that require fixing. An *advisory* is issued when a fix is available. The primary location where we publish advisories is the Amazon Linux Security Center (ALAS). For more information, see Amazon Linux Security Center.

> **Important**
>
> If you want to report a vulnerability or have a security concern regarding AWS cloud services or open source projects, contact AWS Security using the Vulnerability Reporting page

Information on issues and the relevant updates that affect AL2027 are published by the Amazon Linux team in several locations. It's common for security tooling to fetch information from these

primary sources and present the results to you. As such, you might not directly interact with the primary sources that Amazon Linux publishes, but instead the interface provided by your preferred tooling, such as Amazon Inspector.

Amazon Linux Security Center announcements

Amazon Linux *announcements* are provided for items that do not fit into an advisory. This section contains announcements about ALAS itself, along with information that does not fit in an advisory. For more information, see Amazon Linux Security Center (ALAS) Announcements

Amazon Linux Security Center Frequently Asked Questions

For answers to some frequently asked questions about ALAS and how Amazon Linux evaluates CVEs, see Amazon Linux Security Center (ALAS) Frequently Asked Questions (FAQs).

ALAS Advisories

An Amazon Linux Advisory contains important information relevant to Amazon Linux users, typically information about security updates. The Amazon Linux Security Center is where Advisories are visible on the web. Advisory information is also part of the RPM package repository metadata.

Advisories and RPM repositories

An Amazon Linux 2027 package repository may contain metadata describing zero or more updates. The `dnf updateinfo` command is named after the repository metadata filename which contains this information, `updateinfo.xml`. While the command is named `updateinfo`, and the metadata file refers to an `update`, these all refer to package updates which are part of an Advisory.

Amazon Linux Advisories are published on the Amazon Linux Security Center web site, along with information being present in the RPM repository metadata that the `dnf` package manager refers to. The web site and repository metadata are eventually consistent, and there may be inconsistencies in the information on the web site and in repository metadata. This will typically occur when a new release of AL2027 is in the process of being released, there has been an update to an Advisory after the most recent AL2027 release.

While it is typical for a new Advisory to be issued alongside the package update which addresses the issue, this is not always the case. An Advisory can be created for a new issue which is addressed in already released packages. An existing Advisory may also be updated with new CVEs which are addressed by the existing update.

The Deterministic upgrades through versioned repositories on AL2027 feature of Amazon Linux 2027 means that the RPM repository for a particular AL2027 version contains a snapshot of the RPM repository metadata as of that version. This *includes* the metadata describing security updates. The RPM repository for particular AL2027 version *is not updated* after release. New or updated security advisories *will not be visible when looking at an older version of the AL2027 RPM repositories*. Refer to the Listing applicable advisories section for how to use the `dnf` package manager to look at either the `latest` repository version, or a specific AL2027 release.

Advisory IDs

Each Advisory is referred to by an `id`. It is currently a quirk of Amazon Linux where the Amazon Linux Security Center web site will list an Advisory as ALAS-2024-581, while the `dnf` package manager will list that advisory as having the ID of ALAS2023-2024-581. When Applying security updates in-place the package manager ID needs to be used if referring to a specific Advisory.

For Amazon Linux, each major version of the OS has its own namespace of Advisory IDs. There should be no assumptions made as to the format of Amazon Linux Advisory IDs. Historically, Amazon Linux Advisory IDs have followed the pattern of `NAMESPACE-YEAR-NUMBER`. The full range of possible values for `NAMESPACE` is not defined, but has included `ALAS`, `ALASCORRETTO8`, `ALAS2023`, `ALAS2`, `ALASPYTHON3.8`, and `ALASUNBOUND-1.17`. The `YEAR` has been the year in which the advisory was created, and `NUMBER` being a unique integer within the namespace.

While Advisory IDs will *typically* be sequential and in the order the updates are released, there are many reasons why this could not be the case, so this should not be assumed.

Treat the Advisory ID as an opaque string which is unique to each major version of Amazon Linux.

In Amazon Linux 2, each Extra was in a separate RPM repository, and the Advisory metadata is contained only within the repository to which it is relevant. An Advisory for one repository *is not applicable* to another repository. On the Amazon Linux Security Center web site, there is currently one list of Advisories for each major Amazon Linux version, and it is not separated out into per-repository lists.

Advisory Released Date and Advisory Updated Date

The Advisory Released Date for Amazon Linux Advisories indicates when the security update was first made publicly available in the RPM repository. Advisories are posted on the Amazon Linux Security Center web site immediately after fixes are made available for installation through the RPM repository.

The Advisory Updated Date indicates when new information was added to an advisory after it was previously published.

There should not be any assumptions made between the AL2027 version number and the Advisory Released Date of Advisories published alongside that release.

Advisory Types

The RPM repository metadata supports Advisories of different types. While Amazon Linux has near universally only issued Advisories which are security updates, this should not be assumed. It is possible that Advisories for events such as bug fixes, enhancements, and new packages could be issued, and the Advisory be marked as containing that type of update.

Advisory Severities

Each Advisory has its own Severity as each issue is evaluated separately. Multiple CVEs may be addressed in a single Advisory, and each CVE may have a different evaluation, but the Advisory itself has one Severity. There can be multiple Advisories referring to a single package update, thus there can be multiple Severities for a particular package update (one per Advisory).

In order of decreasing Severity, Amazon Linux has used Critical, Important, Moderate, and Low to indicate the Severity of an Advisory. Amazon Linux Advisories may also *not have a Severity*, although this is exceedingly rare.

Amazon Linux is one of the RPM based Linux distributions that uses the term Moderate, while some other RPM based Linux distributions use the equivalent term Medium. The Amazon Linux package manager treats both terms as equivalent, and third party package repositories may use the term Medium.

Amazon Linux Advisories can *change Severity* over time as more is learned about the relevant issues addressed in the Advisory.

The Severity of an Advisory will *typically* track the highest Amazon Linux evaluated CVSS score for the CVEs referenced by the Advisory. There may be cases where this is not the case. One example would be where there is an issue which is addressed for which there is not a CVE assigned.

See the ALAS FAQ for more information about how Amazon Linux uses Advisory severity ratings.

Advisories and Packages

There can be many Advisories for a single package, and not all packages will ever have an Advisory published for them. A particular package version can be referenced in multiple Advisories, each with its own Severity and CVEs.

It is possible for multiple Advisories for the same package update to be issued simultaneously in one new AL2027 release, or in rapid succession.

Like other Linux distributions, there can be one to many different binary packages built from the same source package. For example, ALAS-2024-698 is an Advisory listed on the AL2027 section of the Security Center web site as applying to the `mariadb105` package. This is the *source* package name, and the Advisory itself refers to the *binary* packages alongside the source package. In this case, over a dozen binary packages are built from the one `mariadb105` source package. While it is extremely common for there to be a binary package with the same name as the source package, this is not universal.

While Amazon Linux Advisories have typically listed all binary packages built from the updated source package, this should not be assumed. The package manager and RPM repository metadata format allows for Advisories that list a subset of the updated binary packages.

A particular Advisory may also only apply to a particular CPU Architecture. There can be packages that are not built for all architectures, or issues that do not affect all architectures. In the case where a package is available on all architectures but an issue applies only to one, Amazon Linux has typically not issued an Advisory only referencing only the affected architecture, although this should not be assumed.

Due to the nature of package dependencies, it is common that an Advisory references one package, but installing that update will require other package updates, including packages that are not listed in the Advisory. The `dnf` package manager will handle installing the required dependencies.

Advisories and CVEs

An Advisory may address zero or more CVEs, and there may be multiple Advisories referencing the same CVE. An example of when an Advisory may reference zero CVEs is when a CVE is not yet (or ever) assigned to the issue.

An example of where multiple Advisories may reference the same CVE when (for example) the CVE is applicable to multiple packages. For example, CVE-2024-21208 applies to Corretto 8, 11, 17,

and 21. Each of these Corretto versions is a separate package in Amazon Linux, and there is an Advisory for each of these packages: ALAS-2024-754 for Corretto 8, ALAS-2024-753 for Corretto 11, ALAS-2024-752 for Corretto 17, and ALAS-2024-752 for Corretto 21. While these Corretto releases all have the same list of CVEs, this should not be assumed.

A particular CVE can be evaluated differently for different packages. For example, if a particular CVE is referenced in an Advisory with a Severity of Important, it is possible that another Advisory is issued referencing the same CVE with a different Severity.

The RPM repository metadata allows for a list of References for each Advisory. While Amazon Linux has typically only referenced CVEs, the metadata format does allow for other reference types.

The RPM package repository metadata will only refer to CVEs with a fix available. The Explore section of the Amazon Linux Security Center web site contains information on CVEs that Amazon Linux has evaluated. This evaluation may result in a CVSS base score, Severity, and status for various Amazon Linux releases and packages. The status for a CVE for a particular Amazon Linux release or package may be Not Affected, Pending Fix, or No Fix Planned. The status and evaluation of CVEs may change many times and *in any way* prior to an Advisory being issued. This includes re-evaluation of the applicability of a CVE to Amazon Linux.

The list of CVEs referenced by an Advisory can change after initial publication of that Advisory.

Advisory Text

An Advisory will also contain text describing the issue or issues that were the reason for creating the Advisory. It is common that this text will be the unmodified CVE text. This text may refer to upstream version numbers where a fix is available which are different from the package version that Amazon Linux has applied a fix to. It is common that Amazon Linux will back-port fixes from newer upstream releases. In the case where the Advisory text mentions an upstream release which is different than the version shipped in an Amazon Linux version, the Amazon Linux package versions in the Advisory will be accurate for Amazon Linux.

It is possible for the Advisory text in the RPM repository metadata to be placeholder text simply referring to the Amazon Linux Security Center web site for details.

Kernel Live Patch Advisories

Advisories for live patches are unique in that they refer to a different package (the Linux kernel) than the package the Advisory is against (e.g. `kernel-livepatch-6.1.15-28.43`).

An Advisory for a Kernel Live Patch will reference the issues (such as CVEs) which the particular Live Patch package can address for the *specific* kernel version to which the live patch package applies.

Each live patch is for a *specific* kernel version. In order to apply a live patch for a CVE, the right live patch package for your kernel version needs to be installed, and the live patch applied.

For example, CVE-2023-6111 can be live patched for AL2023 kernel versions `6.1.56-82.125`, `6.1.59-84.139`, and `6.1.61-85.141`. A new kernel version with a fix for this CVE was also released, and has a separate advisory. In order for CVE-2023-6111 to be addressed on AL2023 either a kernel version equal to or later than what ALAS2023-2023-461 specifies needs to be *running*, or one of the kernel versions with a live patch for this CVE needs to be running with the applicable livepatch applied.

When there are new live patches available for a specific kernel version which already has a live patch available, a new version of the `kernel-livepatch-KERNEL_VERSION` package is released. For example, the ALASLIVEPATCH-2023-003 Advisory was issued with the `kernel-livepatch-6.1.15-28.43-1.0-1.amzn2023` package which contained live patches for the `6.1.15-28.43` kernel covering three CVEs. Later, the ALASLIVEPATCH-2023-009 Advisory was issued with the `kernel-livepatch-6.1.15-28.43-1.0-2.amzn2023` package; an update to the previous live patch package for the `6.1.15-28.43` kernel containing live patches for another three CVEs. There were also other live patch Advisories issues for other kernel versions, with packages containing live patches for those specific kernel versions.

For anyone developing tools around security advisories, it is also recommended to look at the XML Schema for Advisories and `updateinfo.xml` section for more information.

XML Schema for Advisories and `updateinfo.xml`

The `updateinfo.xml` file is part of the package repository format. It is the metadata that the `dnf` package manager parses to implement functionality such as Listing applicable advisories and Applying security updates in-place.

We recommended that the API of the `dnf` package manager is used rather than writing custom code to parse the repository metadata formats.

The RPM Software Management project documents the RPM metadata formats in the rpm-metadata repository on GitHub.

For those developing tools to directly parse the `updateinfo.xml` metadata, paying careful attention to the rpm-metadata documentation is strongly advised. The documentation covers what

has been seen in the wild, which includes many exceptions to what you may reasonably interpret as a rule for the metadata format.

There is also a growing set of real-world examples of `updateinfo.xml` files in the raw-historical-rpm-repository-examples repository on GitHub.

In case anything is unclear in the documentation, you can open an issue on the GitHub project so that we can answer the question and update the documentation appropriately. As Open Source projects, pull requests updating documentation are also welcome.

Listing applicable advisories

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

The DNF package manager has access to metadata describing which advisories are fixed in which package versions. It can list the advisories that are applicable to an instance or container image. Tools such as AWS Systems Manager use this metadata to show relevant updates across a fleet rather than a single instance.

In AL2027, advisory information is queried with the `dnf advisory` command and its `summary`, `list`, and `info` subcommands:

```
# Summary of advisories applicable to this system
dnf advisory summary

# One line per applicable advisory and package
dnf advisory list

# Full details of one advisory
dnf advisory info <advisory-id>
```

Advisory publication for AL2027 begins with the public preview, using the same Amazon Linux Security Advisory (ALAS) model as AL2023. The following outputs show the format. They were captured with example advisories, so the advisory identifiers and CVE numbers are examples:

```
dnf advisory summary
```

```
Available advisory information summary:
Security    : 2
  Critical  : 0
  Important : 1
  Moderate  : 0
  Low       : 1
  Other     : 0
Bugfix      : 0
Enhancement : 0
Other       : 0
```

```
dnf advisory list
```

```
Name              Type     Severity                  Package                  Issued
ALAS2027-2026-001 security Important zsh-5.9-21.amzn2027.x86_64 2026-08-17 18:00:00
ALAS2027-2026-002 security Low       zsh-5.9-22.amzn2027.x86_64 2026-08-17 18:00:00
```

```
dnf advisory info ALAS2027-2026-001
```

```
Name        : ALAS2027-2026-001
Title       : Amazon Linux 2027 - ALAS2027-2026-001: Important priority package update:
 zsh
Severity    : Important
Type        : security
Status      : final
Vendor      : linux-security@amazon.com
Issued      : 2026-08-17 18:00:00
Description : Package updates are available for Amazon Linux 2027 that fix the
 following vulnerabilities:
            : CVE-2026-12345
Message     :
Rights      :
Reference   :
  Title     : CVE-2026-12345
  Id        : CVE-2026-12345
  Type      : cve
  Url       : https://alas.aws.amazon.com/cve/html/CVE-2026-12345.html
Collection  :
```

```
  Packages  : zsh-5.9-21.amzn2027.x86_64
```

By default, the commands consider advisories for available package upgrades. Use the `--installed`, `--updates`, or `--all` options to change the scope, and `--contains-pkgs=`*`name`* to filter by package. As with other DNF commands, add `--releasever=`*`version`* to evaluate the metadata of a newer release than the one the system is locked to:

```
dnf advisory summary --releasever=latest
```

> **Note**
>
> After an AL2027 release is made, it is immutable. New or updated advisories are only added to the metadata of *new* releases. To see the newest advisories, query with `--releasever=latest`.

If you are migrating from AL2023: DNF5 replaces the `dnf updateinfo --list` option style with subcommands, such as `dnf advisory list`. The `dnf updateinfo` spelling is kept as an alias for `dnf advisory`, so `dnf updateinfo list` also works.

To list advisories against a specific release, name it. To list against the newest release, use `latest`. Both forms report zero advisories until advisory publication begins with the public preview:

```
dnf advisory summary --releasever=2027.0.20260817
dnf advisory summary --releasever=latest
```

For the preview patching policy, see Known issues and preview limitations. For information about Amazon Linux Security Advisories, see Amazon Linux Security advisories for AL2027.

Applying security updates in-place

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

> **Note**
>
> We recommend applying *all* updates available in a new AL2027 release. Picking only security updates, or only specific updates, should be the exception rather than the rule. For an overview of applying all updates, see Applying updates using DNF and repository versions.

To restrict an upgrade to packages that have a security advisory, use the `--security` option:

```
sudo dnf upgrade --security --releasever=latest
```

To apply only the updates from specific advisories, name them with the `--advisories` option. The advisory identifier in this example follows the Amazon Linux Security Advisory (ALAS) format:

```
sudo dnf upgrade --releasever=latest --advisories=ALAS2027-2026-001
```

This updates the packages mentioned in the advisory to the latest version available in the selected release, which might be newer than the version the advisory introduced. To update the affected packages only up to the lowest versions that fix the advisory, use `dnf upgrade-minimal` instead:

```
sudo dnf upgrade-minimal --releasever=latest --advisories=ALAS2027-2026-001
```

In the following example, the advisory is fixed by `zsh-5.9-21` and the repository also carries the newer `zsh-5.9-22`. The `upgrade-minimal` command selects the lowest version that fixes the advisory:

```
Updating and loading repositories:
Repositories loaded.
Package          Arch   Version           Repository       Size
Upgrading:
 zsh             x86_64 0:5.9-21.amzn2027 example       0.0   B
   replacing zsh x86_64 0:5.9-20.amzn2027 amazonlinux   7.9 MiB

Transaction Summary:
 Upgrading:          1 package
 Replacing:          1 package
```

If the fixed versions are already installed, both commands make no changes. You can also filter by severity with `--advisory-severities=`*`severity`*, or by CVE with `--cves=`*`cve-id`*.

> **Note**
>
> In DNF5, the DNF (version 4) option `--advisory` was renamed to `--advisories`. The old spelling is kept as a compatibility alias in AL2027.

> **Note**
>
> Unless the `system-release` package is updated, the release version that DNF is locked to *does not change*.

> **Warning**
>
> When installing updates from a newer release without changing the version that DNF is locked to, take care with subsequent mutating DNF operations. Package dependencies might have changed in the newer release, and the older release that you remain locked to might not be able to satisfy the new dependencies.

Advisory publication for AL2027 begins with the public preview. Until then, advisory-filtered commands find no matching updates. The preceding example was captured with example advisories, so the advisory identifier and package versions are examples. For listing the advisories that apply to a system, see Listing applicable advisories.

Setting SELinux modes for AL2027

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

Security-Enhanced Linux (SELinux) is a collection of kernel features and user-space utilities that provide a strong, flexible mandatory access control (MAC) implementation. By default, SELinux is

`enabled` and set to `enforcing` mode for AL2027. In `enforcing` mode, SELinux actively enforces the loaded security policy on the entire system.

SELinux provides an enhanced mechanism to enforce the separation of information based on confidentiality and integrity requirements. This separation of information reduces threats of tampering and bypassing of application security mechanisms. It also confines damage caused by malicious or flawed applications.

SELinux includes a set of sample security policy configuration files that are designed to meet everyday security goals.

For more information about SELinux features and functionality, see SELinux Notebook and Policy Languages on the GitHub website.

Topics

- Default SELinux status and modes for AL2027
- Change to permissive mode
- Option to disable SELinux for AL2027

Default SELinux status and modes for AL2027

For AL2027, SELinux is `enabled` and set to `enforcing` mode by default. In `enforcing` mode, SELinux actively enforces the loaded security policy on the entire system.

To find the current SELinux status, policy, and mode, run the **`getenforce`** or **`sestatus`** command.

With the default status set to `enabled` and `enforcing`, the **`getenforce`** command returns `Enforcing`.

The **`sestatus`** command returns the SELinux status and the current SELinux policy as shown in the following example:

```
$ sestatus
SELinux status:                 enabled
SELinuxfs mount:                /sys/fs/selinux
SELinux root directory:         /etc/selinux
Loaded policy name:             targeted
Current mode:                   enforcing
Mode from config file:          enforcing
Policy MLS status:              enabled
```

```
Policy deny_unknown status:     allowed
Memory protection checking:     actual (secure)
Max kernel policy version:      35
```

When you run SELinux in `permissive` mode, you might label files incorrectly. When SELinux is `disabled`, files aren't labeled. Both incorrectly labeled and unlabeled files can cause problems in `enforcing` mode.

SELinux automatically relabels files to avoid this problem. SELinux prevents labeling problems with automatic relabeling when you change the status to `enabled`.

Change to `permissive` mode

When you run SELinux in `permissive` mode, SELinux policy isn't enforced. In `permissive` mode, SELinux logs AVC messages but doesn't deny operations. You can use these AVC messages for troubleshooting, debugging, and SELinux policy improvements.

To find the current SELinux mode, run the `getenforce` command.

```
getenforce
Enforcing
```

Editing the config file to enable `permissive` mode

To change the mode to `permissive`, use the following steps.

1. Edit the `/etc/selinux/config` file to change to `permissive` mode. The `SELINUX` value should look like the following example.

   ```
   SELINUX=permissive
   ```

2. Restart your system to complete the change to `permissive` mode.

   ```
   $ sudo reboot
   ```

Using cloud-init to enable `permissive` mode

As an alternative, when you launch your instance, pass the following `cloud-config` as user-data to enable `permissive` mode.

```
#cloud-config
selinux:
  mode: permissive
```

By default, this setting causes the instance to reboot. For greater stability, we recommend rebooting your instance. However, if you prefer, you can skip the reboot by providing the following `cloud-config`.

```
#cloud-config
selinux:
  mode: permissive
  selinux_no_reboot: 1
```

Option to disable SELinux for AL2027

When you disable SELinux, SELinux policy isn't loaded or enforced and Access Vector Cache (AVC) messages aren't logged. You lose all benefits of running SELinux.

Instead of disabling SELinux, we recommend using `permissive` mode. It costs only a little more to run in `permissive` mode than it does to disable SELinux completely. Transitioning from `permissive` mode to `enforcing` mode requires much less configuration than transitioning back to `enforcing` mode after disabling SELinux. You can label files, and the system can track and log actions that the active policy might have denied.

For information about how to change to `permissive` mode, see Change to `permissive` mode.

Disable SELinux

When you disable SELinux, SELinux policy isn't loaded or enforced, and AVC messages aren't logged. You lose all benefits of running SELinux.

To disable SELinux, use the following steps.

1. Ensure that the `grubby` package is installed.

   ```
   rpm -q grubby
   grubby-version
   ```

2. Configure your bootloader to add `selinux=0` to the kernel command line.

```
sudo grubby --update-kernel ALL --args selinux=0
```

3. Restart your system.

```
sudo reboot
```

4. Run the `getenforce` command to confirm that SELinux is `Disabled`.

```
$ getenforce
Disabled
```

For more information about SELinux, see the SELinux Notebook and SELinux modes, both on the GitHub website.

Post-Quantum Cryptography (PQC) on AL2027

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

The system-wide cryptographic policies on AL2027 enable post-quantum cryptography (PQC) by default. Most users will not need to change this default configuration.

AL2027 includes the following updated libraries and applications with PQC support:

- **OpenSSH** enables Post-Quantum Cryptography (PQC) key exchange by default.
- **OpenSSL 3.5** added support for the ML-KEM hybrid key exchange algorithm, and ML-DSA (ML-DSA-44, ML-DSA-65, and ML-DSA-87) and SLH-DSA signature algorithms.
- **GnuTLS 3.8.10** supports ML-KEM hybrid key exchange algorithms and ML-DSA-44, ML-DSA-65, and ML-DSA-87 signature algorithms for TLS communications.
- **NSS 3.124** supports the ML-KEM hybrid key exchange algorithm and ML-DSA-44, ML-DSA-65, and ML-DSA-87 signature algorithms for TLS communications.

For more information about Post-Quantum Cryptography on AWS, see AWS Cloud Security > Post-Quantum Cryptography

How to disable Post-Quantum Cryptography (PQC) on AL2027

In some cases users may need to disable the use of post-quantum cryptography. The system-wide cryptographic policies provides the `NO-PQ` subpolicy to accomplish this. After applying the `NO-PQ` subpolicy, hybrid post-quantum key exchange using the Module-Lattice-Based Key-Encapsulation Mechanism (ML-KEM) and post-quantum digital signatures using the Module-Lattice-Based Digital Signature Standard (ML-DSA) will no longer be enabled in the LEGACY, DEFAULT, FUTURE, or FIPS cryptographic policies.

Prerequisites

- An existing AL2027 Amazon EC2 instance.
- You must connect to your Amazon EC2 instance using SSH or AWS Systems Manager.

Enable the `NO-PQ` subpolicy on AL2027

1. Ensure that the latest `crypto-policies` and `crypto-policies-scripts` packages are installed:

```
sudo dnf -y install crypto-policies-scripts
sudo dnf -y update crypto-policies crypto-policies-scripts
```

2. Use the `update-crypto-policies` command to enable the `NO-PQ` subpolicy:

```
sudo update-crypto-policies --set DEFAULT:NO-PQ
```

3. To check that you are using the `NO-PQ` subpolicy, run the following command:

```
update-crypto-policies --show
```

   For example, if you are using the `DEFAULT` policy you should see the following output:

```
DEFAULT:NO-PQ
```

Enable FIPS Mode on AL2027

AL2027 Preview

AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

This section explains how to enable Federal Information Processing Standards (FIPS) on AL2027. For more information about FIPS, see:

- Federal Information Processing Standard (FIPS)
- Compliance FAQs: Federal Information Processing Standards

Note

This section documents how to enable FIPS mode in AL2027, it does not cover the certification status of AL2027 cryptographic modules.

Prerequisites

- An existing AL2027 Amazon EC2 instance with access to the internet to download required packages.
- You must connect to your Amazon EC2 instance using SSH or AWS Systems Manager.

Important

ED25519 SSH user keys are not supported in FIPS mode. If you launched your Amazon EC2 instance using an ED25519 SSH key pair, you must generate new keys using another algorithm (such as RSA or ECDSA) or you may lose access to your instance after enabling FIPS mode. For more information see Create key pairs in the *Amazon EC2 User Guide*.

> **Note**
>
> AL2027 no longer provides the `fips-mode-setup` command-line utility. FIPS mode only needs to be enabled in the Linux kernel, and after rebooting the system-wide cryptographic FIPS policy will be automatically enabled via the `fips-crypto-policies` dracut module or the systemd `fips-crypto-policy-overlay` service.

Enable FIPS Mode

1. Connect to your AL2027 instance using SSH or AWS Systems Manager.
2. Ensure the system is up to date. For more information, see Manage package and operating system updates in AL2027.
3. Ensure the `crypto-policies` utilities are installed and up-to-date.

   ```
   sudo dnf -y install crypto-policies crypto-policies-scripts
   ```

4. Enable FIPS mode by running the following command:

   ```
   sudo /sbin/grubby --update-kernel=ALL --args="fips=1"
   ```

5. Reboot the instance using the following command.

   ```
   sudo reboot
   ```

6. To verify that the Linux kernel is in FIPS mode, reconnect to your instance and run the following command:

   ```
   cat /proc/sys/crypto/fips_enabled
   ```

   An output of 1 means that the kernel is running FIPS mode.

7. To verify that the system-wide cryptographic policies are set to `FIPS`:

   ```
   update-crypto-policies --show
   ```

   The following output shows the FIPS crypto policy is enabled:

```
FIPS
```

Enable FIPS Mode in an AL2027 Container

AL2027 Preview

AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

This section explains how to enable Federal Information Processing Standards (FIPS) in an AL2027 container. For more information about FIPS, see:

- Federal Information Processing Standard (FIPS)
- Compliance FAQs: Federal Information Processing Standards

Note

This section documents how to enable FIPS mode in an AL2027 container. It does not cover the certification status of AL2027 cryptographic modules.

Prerequisites

- An existing AL2027 Amazon EC2 instance with access to the internet to download required packages.
- You must connect to your Amazon EC2 instance using SSH or AWS Systems Manager.

Note

AL2027 no longer provides the `fips-mode-setup` command-line utility. FIPS mode only needs to be enabled in the container host's Linux kernel.

Enable FIPS Mode in an AL2027 Container

1. FIPS mode must first be enabled on the AL2027 container Host. Follow the instructions at Enable FIPS Mode on AL2027 to enable FIPS mode on the Host.
2. Connect to your AL2027 container host instance using SSH or AWS Systems Manager.
3. FIPS mode will be automatically enabled in an AL2027 container if the AL2027 host is in FIPS mode and `/proc/sys/crypto/fips_enabled` is accessible from within the container. If the contents of `/proc/sys/crypto/fips_enabled` is `0` then FIPS is not enabled, and a value of `1` indicates that FIPS mode is enabled.

   You can verify that FIPS is enabled by running the following command on both the AL2027 host and container:

   ```
   cat /proc/sys/crypto/fips_enabled
   ```

4. Next, enable the FIPS crypto-policies within the container. There are several ways to accomplish this, described in the options below. Use the option that works best for your environment.

   a. Enable the FIPS crypto-policies manually within the container using the `update-crypto-policies` command:

   ```
   # Run these commands inside the container
   dnf install -y crypto-policies-scripts
   update-crypto-policies --set FIPS
   ```

   b. Create `bind` mounts within the AL2027 container - **note:** this option may require running the container with additional privileges to allow bind mounts (i.e. `docker run --cap-add=SYS_ADMIN ...`):

   ```
   # Run these commands inside the container
   mount --bind /usr/share/crypto-policies/back-ends/FIPS /etc/crypto-policies/back-ends
   echo "FIPS" > /usr/share/crypto-policies/default-fips-config
   mount --bind /usr/share/crypto-policies/default-fips-config /etc/crypto-policies/config
   ```

   c. It is also possible to create a bind mount so that the AL2027 container matches the AL2027 host's crypto-policies. The following is only provided as an example. This configuration could cause issues if there are incompatible differences in the crypto-policies and package versions between the container and host:

```
sudo docker pull amazonlinux:2027
sudo docker run --mount type=bind,readonly,src=/etc/crypto-policies,dst=/etc/
crypto-policies -it amazonlinux:2027
```

5. After performing the steps above you can again verify that FIPS is enabled in the container with the following commands:

```
$ cat /etc/crypto-policies/config
FIPS

$ cat /proc/sys/crypto/fips_enabled
1
```

Swap OpenSSL FIPS providers on AL2027

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

This section explains how to switch between the `latest` and `certified` OpenSSL FIPS providers on AL2027.

For more information about FIPS, see:

- Federal Information Processing Standard (FIPS)
- Compliance FAQs: Federal Information Processing Standards
- FedRAMP Policy for Cryptographic Module Selection and Use

> **Important**
>
> On AL2027 the default OpenSSL FIPS provider is the `openssl-fips-provider-latest` package, which receives regular bugfix and security updates.

The instructions below are only for customers who want to pin to the `openssl-fips-provider-certified` package. This version of the FIPS provider will match the checksum on the NIST certificate, and may not have the latest updates.
See the AL2027 FAQ for more information about FIPS certified modules and package versions.

Note

The initial version of the `openssl-fips-provider-certified` package on AL2027 contains the certified FIPS provider from AL2023 (certificate #5438). See also: Amazon Linux 2023 FAQs

Prerequisites

- An existing AL2027 Amazon EC2 instance with access to the internet to download required packages.
- You must connect to your Amazon EC2 instance using SSH or AWS Systems Manager.
- To enable FIPS mode on AL2027, follow the instructions at Enable FIPS Mode on AL2027.

Switch between `openssl-fips-provider-latest` and `openssl-fips-provider-certified`

1. Use `dnf` to switch the OpenSSL FIPS provider:

```
sudo dnf -y swap openssl-fips-provider-latest openssl-fips-provider-certified
```

2. Check that you are using the certified OpenSSL FIPS provider. With AL2027 in FIPS mode, run the following command:

```
openssl list -providers
```

You should see the following output:

```
Providers:
  base
```

```
    name: OpenSSL Base Provider
    version: 3.5.7
    status: active
  default
    name: OpenSSL Default Provider
    version: 3.5.7
    status: active
  fips
    name: Amazon Linux 2023 - OpenSSL FIPS Provider
    version: 3.2.2-799901ad7ab41d45
    status: active
```

AL2027 kernel hardening

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

The 7.1 Linux kernel in AL2027 is configured and built with a number of hardening options and features. Many of these follow the Kernel Self Protection Project (KSPP) Recommended Settings. Where AL2027 deviates from a KSPP recommendation, the deviation is a deliberate choice that balances hardening against the performance, compatibility, and operational needs of a general-purpose cloud operating system.

The following table lists notable kernel hardening configuration options and their status in the AL2027 7.1 kernel for both the `x86_64` and `aarch64` architectures. The values have the following meaning.

- `y` – the option is built into the kernel.
- `m` – the option is built as a loadable kernel module.
- `n` – the option exists in this kernel but is disabled.
- A number or string (for example `0`, `65536`, or `sha512`) – the configured value of the option.
- `N/A` – the configuration symbol is not present in this kernel on this architecture. This happens when the option was removed or renamed upstream, when it belongs to the other architecture, or when a different choice within the same option group is selected.

CONFIG option	AL2027/7.1/aarch64	AL2027/7.1/x86_64
CONFIG_ACPI_CUSTOM_METHOD	N/A	N/A
CONFIG_AMD_IOMMU	N/A	y
CONFIG_AMD_IOMMU_V2	N/A	N/A
CONFIG_ARM64_BTI	y	N/A
CONFIG_ARM64_BTI_KERNEL	N/A	N/A
CONFIG_ARM64_E0PD	y	N/A
CONFIG_ARM64_EPAN	y	N/A
CONFIG_ARM64_MTE	y	N/A
CONFIG_ARM64_PTR_AUTH	y	N/A
CONFIG_ARM64_PTR_AUTH_KERNEL	y	N/A
CONFIG_ARM64_SW_TTBR0_PAN	y	N/A
CONFIG_BINFMT_MISC	m	m
CONFIG_BUG	y	y
CONFIG_BUG_ON_DATA_CORRUPTION	y	y
CONFIG_CFI_CLANG	N/A	N/A
CONFIG_CFI_PERMISSIVE	N/A	N/A

CONFIG option	AL2027/7.1/aarch64	AL2027/7.1/x86_64
CONFIG_COMPAT	y	y
CONFIG_COMPAT_BRK	n	n
CONFIG_COMPAT_VDSO	N/A	n
CONFIG_DEBUG_KERNEL	y	y
CONFIG_DEBUG_LIST	y	y
CONFIG_DEBUG_NOTIFIERS	n	n
CONFIG_DEBUG_SG	n	n
CONFIG_DEBUG_VIRTUAL	n	n
CONFIG_DEBUG_WX	n	n
CONFIG_DEFAULT_MMAP_MIN_ADDR	65536	65536
CONFIG_DEVKMEM	N/A	N/A
CONFIG_DEVMEM	n	n
CONFIG_EFI_DISABLE_PCI_DMA	n	n
CONFIG_FORTIFY_SOURCE	y	y
CONFIG_HARDENED_USERCOPY	y	y
CONFIG_HIBERNATION	y	y
CONFIG_HW_RANDOM_TPM	N/A	N/A

CONFIG option	AL2027/7.1/aarch64	AL2027/7.1/x86_64
CONFIG_IA32_EMULATION	N/A	y
CONFIG_INET_DIAG	m	m
CONFIG_INIT_ON_ALLOC_DEFAULT_ON	n	n
CONFIG_INIT_ON_FREE_DEFAULT_ON	n	n
CONFIG_INIT_STACK_ALL_ZERO	N/A	N/A
CONFIG_INTEL_IOMMU	N/A	y
CONFIG_INTEL_IOMMU_DEFAULT_ON	N/A	n
CONFIG_INTEL_IOMMU_SVM	N/A	n
CONFIG_IOMMU_DEFAULT_DMA_STRICT	n	n
CONFIG_IOMMU_SUPPORT	y	y
CONFIG_IO_STRICT_DEVMEM	N/A	N/A
CONFIG_KASAN_HW_TAGS	N/A	N/A
CONFIG_KEXEC	y	y
CONFIG_KFENCE	n	n
CONFIG_LDISC_AUTOLOAD	n	n

CONFIG option	AL2027/7.1/aarch64	AL2027/7.1/x86_64
CONFIG_LEGACY_PTYS	n	n
CONFIG_LEGACY_TIOCSTI	n	n
CONFIG_LEGACY_VSYSCALL_NONE	N/A	n
CONFIG_LIST_HARDENED	y	y
CONFIG_LOCK_DOWN_KERNEL_FORCE_CONFIDENTIALITY	n	n
CONFIG_MAGIC_SYSRQ_DEFAULT_ENABLE	0x1	0x1
CONFIG_MITIGATION_PAGE_TABLE_ISOLATION	N/A	y
CONFIG_MITIGATION_SLS	N/A	n
CONFIG_MODIFY_LDT_SYSCALL	N/A	n
CONFIG_MODULE_FORCE_LOAD	y	y
CONFIG_MODULES	y	y
CONFIG_MODULE_SIG	y	y
CONFIG_MODULE_SIG_ALL	y	y

CONFIG option	AL2027/7.1/aarch64	AL2027/7.1/x86_64
CONFIG_MODULE_SIG_FORCE	n	n
CONFIG_MODULE_SIG_HASH	sha512	sha512
CONFIG_MODULE_SIG_KEY	certs/signing_key.pem	certs/signing_key.pem
CONFIG_MODULE_SIG_SHA512	y	y
CONFIG_PAGE_TABLE_CHECK	n	n
CONFIG_PANIC_ON_OOPS	y	y
CONFIG_PANIC_TIMEOUT	0	0
CONFIG_PROC_KCORE	y	y
CONFIG_PROC_MEM_ALWAYS_FORCE	n	n
CONFIG_PROC_MEM_FORCE_PTRACE	y	y
CONFIG_PROC_MEM_NO_FORCE	n	n
CONFIG_RANDOMIZE_BASE	n	y
CONFIG_RANDOMIZE_KSTACK_OFFSET_DEFAULT	n	n

CONFIG option	AL2027/7.1/aarch64	AL2027/7.1/x86_64
CONFIG_RANDOMIZE_MEMORY	N/A	y
CONFIG_RANDOM_KMALLOC_CACHES	n	n
CONFIG_RANDOM_TRUST_BOOTLOADER	N/A	N/A
CONFIG_RANDOM_TRUST_CPU	N/A	N/A
CONFIG_RANDSTRUCT_FULL	N/A	N/A
CONFIG_RESET_ATTACK_MITIGATION	n	n
CONFIG_SCHED_CORE	N/A	y
CONFIG_SCHED_STACK_END_CHECK	y	y
CONFIG_SECCOMP	y	y
CONFIG_SECCOMP_FILTER	y	y
CONFIG_SECURITY	y	y
CONFIG_SECURITY_DMESG_RESTRICT	y	y
CONFIG_SECURITY_LANDLOCK	y	y
CONFIG_SECURITY_LOCKDOWN_LSM	y	y

CONFIG option	AL2027/7.1/aarch64	AL2027/7.1/x86_64
CONFIG_SECURITY_LOCKDOWN_LSM_EARLY	y	y
CONFIG_SECURITY_SELINUX_BOOTPARAM	y	y
CONFIG_SECURITY_SELINUX_DEBUG	n	n
CONFIG_SECURITY_SELINUX_DEVELOP	y	y
CONFIG_SECURITY_SELINUX_DISABLE	N/A	N/A
CONFIG_SECURITY_WRITABLE_HOOKS	N/A	N/A
CONFIG_SECURITY_YAMA	y	y
CONFIG_SHADOW_CALL_STACK	N/A	N/A
CONFIG_SHUFFLE_PAGE_ALLOCATOR	y	y
CONFIG_SLAB_BUCKETS	y	y
CONFIG_SLAB_FREELIST_HARDENED	y	y
CONFIG_SLAB_FREELIST_RANDOM	y	y
CONFIG_SLAB_MERGE_DEFAULT	y	y
CONFIG_SLUB_DEBUG	y	y

CONFIG option	AL2027/7.1/aarch64	AL2027/7.1/x86_64
CONFIG_STACKPROTECTOR	y	y
CONFIG_STACKPROTECTOR_STRONG	y	y
CONFIG_STATIC_USERMODEHELPER	n	n
CONFIG_STRICT_DEVMEM	n	n
CONFIG_STRICT_KERNEL_RWX	y	y
CONFIG_STRICT_MODULE_RWX	y	y
CONFIG_SYN_COOKIES	y	y
CONFIG_UBSAN	n	n
CONFIG_UNMAP_KERNEL_AT_EL0	y	N/A
CONFIG_VMAP_STACK	y	y
CONFIG_WERROR	n	n
CONFIG_X86_64	N/A	y
CONFIG_X86_KERNEL_IBT	N/A	n
CONFIG_X86_MSR	N/A	y
CONFIG_X86_USER_SHADOW_STACK	N/A	n

CONFIG option	AL2027/7.1/aarch64	AL2027/7.1/x86_64
CONFIG_X86_VSYSCALL_EMULATION	N/A	y
CONFIG_X86_X32	N/A	N/A
CONFIG_X86_X32_ABI	N/A	n
CONFIG_ZERO_CALL_USED_REGS	n	n

Repository metadata signing in AL2027

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

AL2027 repositories include cryptographic signatures for repository metadata. Each repository's `repomd.xml` file comes with a detached GPG signature file (`repomd.xml.asc`). DNF can use this signature to verify the authenticity and integrity of the repository metadata before downloading packages. AL2023 added this capability with release `2023.11.20260406`; AL2027 has it from the start.

Metadata signing is in addition to RPM package signing (`gpgcheck`), which verifies individual packages. Without a signature on the metadata index, that index is trusted based only on TLS and checksums, so a compromised mirror or transport path could serve modified metadata that hides or blocks security updates. Repository metadata signing closes this gap.

Topics

- How repository metadata signing works
- Difference between gpgcheck and repo_gpgcheck
- Enabling repository metadata verification
- Verifying that repository metadata signing is working

- Use repository metadata verification in automation
- Commands that refresh repository metadata
- Detect a skipped repository
- Pinned versions
- GPG public keys for AL2027 repositories

How repository metadata signing works

When you enable `repo_gpgcheck` in a repository configuration, DNF verifies the `repomd.xml.asc` signature against the repository's GPG public key before it uses the metadata. If verification fails, DNF does not use the metadata. Because AL2027 repositories set `skip_if_unavailable=1`, DNF then skips the repository and continues. For how to catch this in automation, see Detect a skipped repository.

The first time DNF verifies a repository's metadata, it prompts you to import the signing key into a separate keyring used for metadata checks. The prompt defaults to `No`, and declining it skips the repository.

The following AL2027 repositories include signed metadata:

- Core repository (`amazonlinux`)
- Source packages repository (`amazonlinux-source`)
- Debug information repository (`amazonlinux-debuginfo`)

Difference between gpgcheck and `repo_gpgcheck`

Setting	What it verifies	Default in AL2027
`gpgcheck=1`	Verifies the GPG signature of individual RPM packages before installation.	Enabled
`repo_gpgcheck=1`	Verifies the GPG signature of the repository metadata (`repomd.xml`) before using the repository.	Disabled

We recommend enabling both settings after you confirm that your automation is ready. Before you enable `repo_gpgcheck`, see Use repository metadata verification in automation.

Enabling repository metadata verification

> **Important**
>
> Repository metadata verification is not enabled by default. Before you enable it, confirm that every unattended DNF command in your automation passes the `-y` option. For more information, see Use repository metadata verification in automation.

Enable for a specific repository

The AL2027 repository configuration in `/etc/yum.repos.d/amazonlinux.repo` sets `repo_gpgcheck=0` by default. To enable metadata verification, change the value to `1` for each repository section:

```
[amazonlinux]
name=Amazon Linux 2027 repository
...
gpgcheck=1
repo_gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-amazon-linux-2027
```

The same key verifies both packages and repository metadata. The `system-release` package installs it at `/etc/pki/rpm-gpg/RPM-GPG-KEY-amazon-linux-2027`.

Disable repository metadata verification

To return to the default behavior, set `repo_gpgcheck=0` again. The next metadata refresh succeeds without verification:

```
sudo sed -i 's/^repo_gpgcheck=1/repo_gpgcheck=0/' /etc/yum.repos.d/amazonlinux.repo
sudo dnf -y makecache
```

Verifying that repository metadata signing is working

After enabling `repo_gpgcheck=1`, clear the DNF cache and refresh the metadata:

```
sudo dnf clean metadata
sudo dnf -y makecache
```

The first time DNF verifies a repository's metadata, it imports that repository's signing key into a separate keyring used for metadata checks, and asks for confirmation. This is the same key already used for package verification; the confirmation for the separate keyring is expected behavior. With `-y`, the key is imported and the cache is created without errors:

```
Updating and loading repositories:
 Amazon Linux 2027 repository           100% |  22.3 KiB/s |   4.5 KiB |  00m00s
>>> repomd.xml GPG signature verification error: Signing key not found
Importing OpenPGP key 0xB1E92E3D:
 UserID     : "Amazon Linux amazon-linux@amazon.com"
 Fingerprint: 37CE 7C38 33E5 A33C 4A1E 4B0D 5408 EAA4 B1E9 2E3D
 From       : file:///etc/pki/rpm-gpg/RPM-GPG-KEY-amazon-linux-2027
The key was successfully imported.
 Amazon Linux 2027 repository           100% |  10.8 MiB/s |   2.4 MiB |  00m00s
Repositories loaded.
Metadata cache created.
```

The `Signing key not found` line before the import is expected on the first run: verification is attempted, the key is imported, and verification then succeeds. If verification fails after the key is imported, DNF reports the `repomd.xml GPG signature verification error` and the metadata is not used.

Use repository metadata verification in automation

The key import prompt defaults to `No`. An unattended DNF run cannot answer the prompt, so it declines the import and metadata verification fails for the repository. The command can still exit with status `0`, so do not rely on the exit code alone. For how to catch this, see Detect a skipped repository.

To prevent this, pass the `-y` option to every unattended DNF command that refreshes metadata, including CI jobs, image and container builds, `cloud-init`, configuration management, and scheduled jobs.

Keep `-y` in your automation permanently. The key is imported once per repository URL, and AL2027 builds each repository URL from the locked release version, so moving to a different release version resolves to a new URL and prompts again. Container image builds need particular

attention. The keyring is part of the image file system and starts empty in every fresh image, so every `RUN dnf` build step is a first-time import.

Commands that refresh repository metadata

Any command that downloads repository metadata triggers verification, and with it the one-time key import. Commands that read only the local cache or the local RPM database do not. The table covers the commands that `dnf5 --help` lists on AL2027, including the preinstalled plugins and the `dnf5-plugins` package. Each behavior was measured against a cleaned cache. Command aliases, such as `dnf update` or `dnf rq`, behave like the command they map to. Options change the result: `--installed` keeps `list` and `info` local, and `-C` (`--cacheonly`) prevents any fetch. When in doubt, pass `-y`.

Command	Refreshes metadata
`dnf makecache`	Yes
`dnf check-update` , `dnf check-upgrade`	Yes
`dnf upgrade`, `dnf upgrade-minimal`	Yes
`dnf install`, `dnf reinstall`	Yes
`dnf downgrade` , `dnf swap`	Yes
`dnf distro-sync`	Yes
`dnf search`, `dnf list`, `dnf info`	Yes
`dnf provides`, `dnf repoquery`	Yes
`dnf advisory`, `dnf updateinfo`	Yes
`dnf group list`	Yes
`dnf download`	Yes
`dnf repoinfo`	Yes
`dnf needs-restarting`	Yes

Command	Refreshes metadata
`dnf changelog`	Yes
`dnf deplist`	Yes
`dnf builddep`, `dnf debuginfo-install`	Yes
`dnf repoclosure` , `dnf reposync`	Yes
`dnf supportinfo`	Yes
`dnf check-release-update`	Yes
`dnf group info`, `dnf environment list`	Yes
`dnf autoremove`	No
`dnf remove`	No
`dnf repolist` (plain or `--all`)	No
`dnf list --installed` , `dnf info --installed`	No
`dnf history`	No
`dnf leaves`, `dnf check`	No
`dnf versionlock`	No
`dnf offline status`	No
`dnf config-manager`	No
`dnf clean`	No
Any command with `-C` or `--cacheonly`	No

Note

Two behaviors changed from AL2023: `dnf autoremove` no longer refreshes metadata, and `dnf needs-restarting` now does.

Detect a skipped repository

The exit code alone does not confirm that verification succeeded. AL2027 repositories set `skip_if_unavailable=1`, so when metadata verification fails, DNF prints the error, skips the repository, and still exits with `0`:

```
sudo dnf upgrade
```

```
Updating and loading repositories:
 Amazon Linux 2027 repository           100% |  46.8 KiB/s |   4.5 KiB |  00m00s
>>> repomd.xml GPG signature verification error: Signing key not found
Repositories loaded.
Nothing to do.
```

On a system with pending updates, this looks like success. Use these two checks instead:

- Audit your automation. Confirm that every metadata-fetching DNF command passes `-y`, so the one-time key import succeeds on each new repository URL.
- Check the output for `GPG signature verification error`. The following command fails when the repository is skipped, which you can use to fail a pipeline:

```
if sudo dnf makecache 2>&1 | grep -q "GPG signature verification error"; then
  echo "repo skipped"
  exit 1
fi
```

After the key is imported, a healthy refresh prints no verification error, and the check passes.

Pinned versions

All AL2027 releases carry signed repository metadata, so pinning `releasever` (with `--releasever` or `/etc/dnf/vars/releasever`) works with `repo_gpgcheck=1`. During the

preview period, old preview releases can be removed from the repositories. A pinned version does not receive updates beyond that version, and a removed version stops resolving. Pin to a current release listed in the AL2027 Release Notes.

GPG public keys for AL2027 repositories

The GPG public key used for repository metadata verification is installed to `/etc/pki/rpm-gpg/` by the `system-release` package. The following table lists the key used by each repository.

Repository	Package signing key	Repodata signing key	Distributed in
Core (`amazonlinux`)	`RPM-GPG-KEY-amazon-linux-2027`	`RPM-GPG-KEY-amazon-linux-2027`	`system-release`
Source packages (`amazonlinux-source`)	`RPM-GPG-KEY-amazon-linux-2027`	`RPM-GPG-KEY-amazon-linux-2027`	`system-release`
Debug information (`amazonlinux-debuginfo`)	`RPM-GPG-KEY-amazon-linux-2027`	`RPM-GPG-KEY-amazon-linux-2027`	`system-release`

Security patching during preview

During the preview period, there is no guarantee that open CVEs for AL2027 packages will be patched in preview artifacts. AL2027 preview artifacts may have unpatched CVEs.

All CVE patching mechanisms and fixes will be in place before any public release.

FIPS validation

FIPS-validated kernel and crypto modules are not available in the preview. They will be available before GA.

UEFI Secure Boot on AL2027

AL2027 supports UEFI Secure Boot. You must use AL2027 with Amazon EC2 instances that support both UEFI and UEFI Secure Boot. For more information, see Requirements to launch an Amazon EC2 instance in UEFI boot mode in the *Amazon EC2 User Guide*.

With UEFI Secure Boot enabled, AL2027 instances accept only kernel-level code, including the Linux kernel and modules, that is signed by Amazon. This makes sure that your instance runs only kernel-level code signed by AWS.

For more information about Amazon EC2 instances and UEFI Secure Boot, see UEFI Secure Boot for Amazon EC2 instances in the *Amazon EC2 User Guide*.

Prerequisites

- You must be using an AMI with AL2027.
- The instance type must support UEFI Secure Boot, as described earlier.

Enable UEFI Secure Boot on AL2027

Standard AL2027 AMIs incorporate a bootloader and a kernel signed by our keys. You can enable UEFI Secure Boot in one of two ways: by enrolling an existing instance, or by creating an AMI with UEFI Secure Boot pre-enabled using a registered image from a snapshot. UEFI Secure Boot isn't enabled by default on the standard AL2027 AMIs.

The boot mode of AL2027 AMIs is set to `uefi-preferred` which makes sure that instances launched with these AMIs will use the UEFI firmware, if the instance type supports UEFI. If the instance type doesn't support UEFI, the instance is launched with Legacy BIOS firmware. When an instance launches in Legacy BIOS mode, UEFI Secure Boot isn't enforced.

For more information about AMI boot modes on Amazon EC2 instances, see Instance launch behavior with Amazon EC2 boot modes in the *Amazon EC2 User Guide*.

Topics

- Enrollment of an existing instance
- Register image from snapshot
- Revocation updates

- How UEFI Secure Boot works on AL2027
- Enrolling your own keys

Enrollment of an existing instance

To enroll an existing instance, populate the specific UEFI firmware variables with a set of keys that enable the firmware to verify the bootloader and the bootloader to verify the kernel on the next boot.

1. AL2027 provides a tool to simplify the enrollment process. Run the following command to provision the instance with the necessary set of keys and certificates.

   ```
   sudo amazon-linux-sb enroll
   ```

2. Run the following command to reboot the instance. After the instance is rebooted, UEFI Secure Boot will be enabled.

   ```
   sudo reboot
   ```

> **Note**
>
> AL2027 AMIs currently don't enable Nitro Trusted Platform Module (NitroTPM) by default with `BootMode` set to `uefi-preferred`. If you need NitroTPM in addition to UEFI Secure Boot, use the information in the following section.

Register image from snapshot

When registering an AMI from a snapshot of an Amazon EBS root volume using the Amazon EC2 `register-image` API, you can provision the AMI with a binary blob that contains the state of the UEFI variable store. By providing the AL2027 `UefiData`, you enable UEFI Secure Boot and don't need to follow the steps in the previous section.

AL2027 provides a pre-built binary blob that can be used directly on Amazon EC2 instances. The binary blob is located in `/usr/share/amazon-linux-sb-keys/uefi.vars` on a running instance. The `amazon-linux-sb-keys` RPM package provides this blob, relevant keys, and the dbx. This is installed by default on AL2027 AMIs.

> **Note**
>
> To make sure that you are using the latest version of keys and revocations, use the blob from the same release of AL2027 that you use to create the AMI.

If you would like to add custom binary blobs, use the keys in `/usr/share/amazon-linux-sb-keys`. Find more information about creating and using binary blobs at Create a binary blob containing a pre-filled variable store in the *Amazon EC2 User Guide*.

When registering an image, we recommend using the `BootMode` parameter of the `RegisterImage` API set to `uefi` instead of `uefi-preferred`. This allows you to enable NitroTPM by setting the `TpmSupport` parameter to `v2.0`. Also, setting `BootMode` to `uefi` prevents boot on BIOS instances, thus making sure that UEFI Secure Boot can't be disabled by accident when switching to an instance type that doesn't support UEFI.

For more information about NitroTPM, see NitroTPM for Amazon EC2 instances in the *Amazon EC2 User Guide*.

Revocation updates

It may be necessary for AL2027 to distribute a new version of the bootloader `grub2` or the Linux kernel signed with updated keys. In that case, the old key may need to be revoked to prevent the chance of allowing exploitable bugs from previous versions of the bootloader to bypass the UEFI Secure Boot verification process.

Package updates to the `grub2` or `kernel` packages always automatically update the list of revocations into the UEFI variable store of the running instance. This means that with UEFI Secure Boot enabled, you might no longer be able to run the old version of a package after installing a security update for the package.

How UEFI Secure Boot works on AL2027

Unlike other Linux distributions, AL2027 doesn't provide an additional component, called a shim, to act as the first stage bootloader. On distributions that use a shim, the shim (generally signed with Microsoft keys) loads the `grub2` bootloader and uses its own code to verify the Linux kernel. The shim also maintains its own set of keys and revocations in the Machine Owner Key (MOK) database, which is located in the UEFI variable store and controlled with the `mokutil` tool. For the

full explanation of how shim and MOK work with other distributions, see the linked Amazon EC2 guide topic.

AL2027 doesn't provide a shim. Because the AMI owner controls the UEFI variables, this intermediary step isn't needed. Adding it would adversely affect launch and boot times. Also, we chose not to include trust for any vendor keys by default, to reduce the chance that undesired binaries could run. As always, you can include binaries if you choose to do so.

With AL2027, UEFI directly loads and verifies our `grub2` bootloader. The `grub2` bootloader was modified to use UEFI to verify the Linux kernel after loading it. As a result, UEFI verifies the Linux kernel using the same certificates in the normal UEFI db variable (authorized key database). It tests the kernel against the same `dbx` variable (revocations database) used for the bootloader and other UEFI binaries. Because we provide our own PK and KEK keys, which control access to the db and `dbx` databases, we can distribute signed updates and revocations as needed without an intermediary such as the shim.

For more information about UEFI Secure Boot, see How UEFI Secure Boot works with Amazon EC2 instances in the *Amazon EC2 User Guide*.

Enrolling your own keys

As documented in the previous section, AL2027 does not require a `shim` for UEFI Secure Boot on Amazon EC2. When you're reading documentation for other Linux distributions, you may find documentation for managing the Machine Owner Key (MOK) database using `mokutil`, which is not present on AL2027. The `shim` and MOK environments work around some limitations of key enrollment in UEFI Firmware that aren't applicable to how Amazon EC2 implements UEFI Secure Boot. With Amazon EC2 there are mechanisms to easily directly manipulate the keys in the UEFI variable store.

If you want to enroll your own keys, you can do so either by manipulating the variable store within an existing instance (see Add keys to the variable store from within the instance) or by constructing a binary blob that's prefilled (see Create a binary blob containing a pre-filled variable store).

Known issues and preview limitations

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

Review these known issues before you deploy AL2027 workloads.

Topics

- NVIDIA drivers
- On-premises VM images
- SSM Patch Manager
- Attestable image examples
- Reporting issues and feedback

NVIDIA drivers

NVIDIA drivers are not available in the initial preview release of AL2027. Follow the AL2027 Release Notes for updates.

On-premises VM images

AL2027 VM images for on-premises use (Hyper-V, KVM, and VMware) are not available in the initial preview release of AL2027. Follow the AL2027 Release Notes for updates.

SSM Patch Manager

SSM Patch Manager patching operations will not succeed on AL2027 at this time. To receive updates, you can run `dnf update` commands manually. For more information, see Updating AL2027.

Attestable image examples

The Amazon Linux Attestable Image Examples repository, which provides recipes for building AMIs with cryptographic attestation support, does not yet include recipes for AL2027. For more information about Attestable Image recipes, see Amazon Linux Kiwi Image Descriptions Examples on GitHub.

Reporting issues and feedback

To report an issue, request a package, or share feedback on the AL2027 public preview, visit the AL2027 repository on GitHub.

Document history for the AL2027 User Guide

> **AL2027 Preview**
>
> AL2027 is currently available for preview. It is intended for evaluation and testing only and is not recommended for production workloads.

The following table describes the documentation releases for the AL2027 User Guide.

Document history

Date	Change	Description
2026-09-03	Initial release	Initial release of the AL2027 User Guide, covering the AL2027 preview.