



User Guide

Amazon Linux 2023



Amazon Linux 2023: User Guide

Copyright © 2025 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 2023?	1
Release cadence	1
Major and minor releases	2
Consuming new releases	2
Long-term support policy	2
Naming and versioning	3
Performance and operational optimizations	4
Relationship to Fedora	5
Customized cloud-init	6
Security updates and features	7
Manage updates	8
Security in the cloud	8
SELinux modes	8
Compliance program	8
SSH server default	8
Major features of OpenSSL 3	8
Networking service	9
Core toolchain packages glibc, gcc, binutils	10
Package management tool	10
Default SSH server configuration	11
Deprecated Functionality	13
compat- packages	13
Deprecated functionality discontinued in AL1, removed in AL2	13
32-bit x86 (i686) AMIs	14
aws-apitools-* replaced by AWS CLI	14
systemd replaces upstart in AL2	15
Functionality deprecated in AL2 and removed in AL2023	15
32-bit x86 (i686) Packages	16
aws-apitools-* replaced by AWS CLI	16
amazon-cloudwatch-agent replaces awslogs	17
bzip revision control system	17
cgroup v1	17
log4j hotpatch (log4j-cve-2021-44228-hotpatch)	18
lsb_release and the system-lsb-core package	18

mcrypt	18
OpenJDK 7 (java-1.7.0-openjdk)	19
Python 2.7	19
rsyslog-openssl replaces rsyslog-gnutls	19
Network Information Service (NIS) / yp	20
Multiple domain names in Amazon VPC create-dhcp-options	20
Sun RPC in glibc	20
OpenSSH key fingerprint in audit log	21
ld.gold Linker	21
ping6	21
ftp Package	21
Deprecated in AL2023	23
32bit x86 (i686) runtime support	24
aspell	24
Berkeley DB (libdb)	24
cron	24
IMDSv1	25
pcre version 1	25
System V init (sysvinit)	25
EOL Packages are deprecated	26
Comparing AL2 and AL2023	27
Added, upgraded, and removed packages	28
Support for each release	28
Naming and versioning changes	28
Optimizations	29
Sourced from multiple upstreams	29
Networking system service	29
Package manager	29
Using cloud-init	29
Graphical desktop support	30
Compiler Triplet	30
32bit x86 (i686) Packages	30
lsb_release and the system-lsb-core package	31
EPEL	31
axel - HTTP/FTP client	33
brotli and libbrotli - compression	33

collectedd - Statistics collection daemon	34
cpulimit	34
exim - mail transfer agent	34
fuse3 - File System in Userspace (FUSE) v3	34
ganglia - Distributed Monitoring System	35
git-lfs - version control large files with Git	35
haveged - an entropy source using the HAVEGE algorithm	35
inotify-tools - inotify command line tools	35
iperf - TCP/UDP Performance benchmark	36
jemalloc - alternative malloc implementation	36
libbsd - BSD-compatible function library	36
libserf - HTTP Client Library	37
libzstd - zstd compression library	37
lighttpd web server	37
lshell - a restricted shell	37
monit - process, file, directory, and devices monitor	37
nodejs	38
perl-Config-General	38
python2-lockfile - file locking	39
python2-rsa - pure Python RSA	39
python2-simplejson - JSON routines for Python 2	39
rkhunter - Rootkit Hunter	40
rssh - a restricted shell for use with OpenSSH	40
sscg - self-signed SSL certificate generator	40
stress - Stress test	40
stress-ng - Stress test	41
tmpwatch - removes files based on last accessed time	41
xmlstarlet - command line XML utilities	41
Python 2.7 has been replaced with Python 3	41
Security updates	42
SELinux	42
OpenSSL 3	42
IMDSv2	43
Removal of log4j hotpatch (log4j-cve-2021-44228-hotpatch)	43
Deterministic upgrades for stability	44
gp3 as default Amazon EBS volume type	44

Unified Control Group hierarchy (cgroup v2)	44
systemd timers replace cron	45
Improved toolchain: gcc, binutils, and glibc	45
systemd journal replaces rsyslog	46
Minimized package dependencies	46
Package changes for curl and libcurl	47
GNU Privacy Guard (GNUPG)	47
Amazon Corretto as the default JVM	47
AWS CLI v2	48
UEFI Preferred and Secure Boot	48
SSH server default configuration changes	48
Kernel changes in AL2023 from AL2	48
IPv4 TTL	48
Security focused kernel config changes	49
Other kernel configuration changes	53
Kernel Filesystem support	55
/tmp changes	60
AMI and Container Image changes	60
Amazon Linux 2 and AL2023 AMI comparison	61
Amazon Linux 2 and AL2023 Minimal AMI comparison	94
Amazon Linux 2 and AL2023 Container comparison	115
Comparing AL1 and AL2023	123
Support for each release	123
systemd replaces upstart as init system	124
Python 2.6 and 2.7 has been replaced with Python 3	124
OpenJDK 8 as oldest JDK	124
Kernel changes in AL2023 from AL1	124
Kernel Live Patching	124
Kernel file system support	124
Security focused kernel config changes	127
Other kernel configuration changes	130
AL1 and AL2023 AMI comparison	130
AL1 and AL2023 Minimal AMI comparison	165
AL1 and AL2023 Container comparison	185
System Requirements	194
CPU requirements for running AL2023	194

ARM CPU Requirements for AL2023	194
x86-64 CPU Requirements for AL2023	195
Memory (RAM) requirements for running AL2023	195
Graphical Desktop	197
Related topics	197
Supplementary Packages for Amazon Linux	198
What is Supplementary Packages for Amazon Linux (or SPAL)?	198
Benefits	198
Support of SPAL packages	199
Reporting package-related issues	199
Related topics	200
Configure SPAL on AL2023	200
Prerequisites	200
Checking prerequisites	200
Installing SPAL on your system	201
Installing SPAL packages	203
Uninstalling SPAL repository from your system	203
Related topics	204
Running applications	205
Resource control with systemd	205
Resource control with <code>systemd-run</code> for running one-off commands	205
Resource control in a systemd service	208
Using cgroups utilities	212
Using AL2023 on AWS	215
Getting started with AWS	215
Sign up for an AWS account	215
Create a user with administrative access	216
Granting programmatic access	217
AL2023 on Amazon EC2	219
Launching AL2023 using the Amazon EC2 console	220
Launching AL2023 using the SSM parameter and AWS CLI	221
Launching the latest AL2023 AMI using CloudFormation	222
Launching AL2023 using a specific AMI ID	224
AL2023 AMI deprecation and life cycle	224
Connecting to AL2023 instances	225
Comparing AL2023 standard (default) and minimal AMIs	225

AL2023 in containers	253
AL2023 Base Container Image	254
AL2023 Minimal container image	256
Building bare-bones AL2023 container images	258
AL2023 container image package list comparison	261
AL2023 Minimal AMI compared to container images	267
AL2023 on Elastic Beanstalk	284
AL2023 CloudShell	285
AL2023 for Amazon ECS container hosts	285
Amazon ECS relevant changes since AL2	286
Custom Amazon ECS-optimized AMI	287
Amazon EFS on AL2023	287
amazon-efs-utils	287
Mounting Amazon EFS file system	288
Amazon EMR on AL2023	288
AL2023 based Amazon EMR releases	288
AL2023 based Amazon EMR on EKS	288
AL2023 on AWS Lambda	289
provided.al2023 Lambda runtime	289
AL2023 based runtimes	289
Tutorials	290
Install LAMP on AL2023	290
Step 1: Prepare the LAMP server	291
Step 2: Test your LAMP server	295
Step 3: Secure the database server	298
Step 4: (Optional) Install phpMyAdmin	299
Troubleshoot	302
Related topics	302
Configure SSL/TLS on AL2023	303
Prerequisites	304
Step 1: Enable TLS on the server	305
Step 2: Obtain a CA-signed certificate	308
Step 3: Test and harden the security configuration	316
Troubleshoot	319
Host a WordPress blog on AL2023	320
Prerequisites	321

Install WordPress	322
Next steps	332
Help! My public DNS name changed and now my blog is broken	333
Redis 6 to Valkey Transition on AL2023	334
Support timeline for Redis 6	334
Introduction to Valkey	335
Migration plan and timeline	335
Migration options and steps	335
Related topics	338
Install GNOME on AL2023	339
Prerequisites	339
Installation	339
Related topics	340
Configure VNC on AL2023	340
Prerequisites	340
Step 1: Installation	341
Step 2: Configuration	341
Step 3: Connect using a VNC client	342
(Optional) Start service at boot	343
(Optional) Disable the idle lockscreen	344
Related topics	344
Using Multi-Gen LRU (MGLRU) on AL2023 kernels	344
Configuration and Tuning	345
AL2023 outside Amazon EC2	347
Download AL2023 VM Images	347
Supported Configurations	347
KVM Requirements	348
VMware Requirements	350
Hyper-V Requirements	352
AL2023 VM configuration	354
NoCloud seed.iso based configuration	355
VMware guestinfo based configuration	358
AL2023 package list comparison for the standard AMI and KVM image	361
AL2023 package list comparison for the standard AMI and VMware OVA image	386
AL2023 package list comparison for the standard AMI and Hyper-V image	411
Identifying Amazon Linux versions	437

/etc/os-release	437
Key differences	438
Field types	438
/etc/os-release examples	440
Comparison with other distributions	441
Amazon Linux Specific	443
/etc/sytem-release	444
/etc/image-id	444
Amazon Linux Specific Examples	444
Example code	446
Filesystem Layout	460
/	460
/boot	461
/boot/efi	461
/etc	461
/home	461
/root	462
/srv	462
/tmp	463
/run	464
/usr	464
/usr/bin	465
/usr/include	465
/usr/lib and /usr/lib64	465
/usr/local	465
/usr/share	465
/var	465
/var/cache	466
/var/lib	466
/var/log	466
/var/spool	466
/var/tmp	466
Updating AL2023	468
Best practices for safely deploying updates	468
Preparing for Minor Updates	471
Preparing for Major Updates	471

Receive notifications on new updates	472
Deterministic upgrades through versioned repositories	473
Control the updates received from major and minor releases	473
Differences between minor and major version upgrades	474
Knowing when updates are available	474
Control the package updates available from the AL2023 repositories	475
Instance replacement	475
In-place Deterministic upgrades	476
Managing updates	483
Checking for available package updates	484
Applying security updates using DNF and repository versions	489
Automatic service restart after (security) updates	501
When is a reboot required to apply security updates?	503
Launching an instance with the latest repository version enabled	503
Getting package support information	504
dnf check-release-update	504
Adding, enabling, or disabling new repositories	508
Adding repositories with cloud-init	511
Kernel Live Patching	512
Limitations	513
Supported configurations and prerequisites	513
Work with Kernel Live Patching	514
Kernel Updates	519
Linux kernel versions on AL2023	519
Updating AL2023 to kernel 6.12	519
AL2023 kernels - Frequently Asked Questions	522
Programming languages and runtimes	524
C/C++ and Fortran	524
GCC 14	525
Language Standard Versions Comparison	526
Go	528
AL2023 Lambda function: Go	528
Java	528
Node.js	38
Perl	530
Perl modules	530

PHP	531
Migrating to new PHP versions	531
Migrating from PHP 7.x	531
PHP modules	532
Python	532
Python modules	533
Rust	533
AL2023 Lambda function: Rust	533
TypeScript	534
AL2023 Reserved Users and Groups	539
List of AL2023 Reserved Users	539
List of AL2023 Reserved Groups	547
Codecs available in AL2023	560
Security and Compliance	563
Security advisories	564
ALAS Announcements	564
ALAS FAQs	564
ALAS Advisories	565
Advisories and RPM repositories	565
Advisory IDs	565
Advisory Released Date and Advisory Updated Date	566
Advisory Types	566
Advisory Severities	567
Advisories and Packages	567
Advisories and CVEs	568
Advisory Text	569
Kernel Live Patch Advisories	569
updateinfo.xml schema	570
Listing applicable Advisories	571
In-place updates	574
Applying updates mentioned in an Advisory	575
Setting SELinux modes for AL2023	578
Default SELinux status and modes for AL2023	579
Change to enforcing mode	579
Option to disable SELinux	581
Enable FIPS Mode on AL2023	582

Enable FIPS Mode in an AL2023 Container	584
Swap OpenSSL FIPS providers on AL2023	586
Kernel Hardening	587
Kernel Hardening options (architecture independent)	588
x86-64 specific Kernel Hardening options	603
aarch64 specific Kernel Hardening options	607
UEFI Secure Boot on AL2023	608
Enable UEFI Secure Boot on AL2023	609
Enrollment of an existing instance	610
Register image from snapshot	610
Revocation updates	611
How UEFI Secure Boot works on AL2023	611
Enrolling your own keys	612

What is Amazon Linux 2023?

Amazon Linux 2023 (AL2023) is the next generation of Amazon Linux from Amazon Web Services (AWS). With AL2023, you can develop and run cloud and enterprise applications in a secure, stable, and high-performance runtime environment. Also you get an application environment that offers long-term support with access to the latest innovations in Linux. AL2023 is provided at no additional charge.

AL2023 is the successor to Amazon Linux 2 (AL2). For information about the differences between AL2023 and AL2, see [Comparing AL2 and AL2023](#) and [Package changes in AL2023](#).

Topics

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

Release cadence

Amazon Linux 2023 (AL2023) was released in March 2023 and will be supported until June 30, 2029. There are two phases of support:

- **Standard support** – During this phase, the release receives quarterly minor version updates. The standard support phase ends June 30, 2027.
- **Maintenance** – During this phase, the release receives only security updates and critical bug fixes. These updates are published as soon as they're available. The maintenance phase ends June 30, 2029.

Major and minor releases

With every Amazon Linux release (major version, minor version, or a security release), we release a new Linux Amazon Machine Image (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, Glib C, OpenSSL, and any 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.
- **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 latest language runtimes, such as PHP. They might also include other popular software packages such as Ansible and Docker.

Consuming new releases

Updates are provided through a combination of new Amazon Machine Image (AMI) releases and corresponding new repositories. By default, a new AMI and the repository that it points to are coupled. However, you can point your running Amazon EC2 instances to newer repository versions over time to apply updates on the running instances. You can also update by launching new instances of the latest AMIs.

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 DNF package manager receive support for the lifetime of the major AL2023 release. Packages that aren't part of the core packages are supported based on their specific upstream sources. You can see the specific 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
```

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 as we are collecting feedback. Feedback on AL2023 can be provided through your designated AWS representative or by filing an issue in the [amazon-linux-2023 repo](#) on GitHub.

Naming and versioning

AL2023 provides a minor release every three months during the two years of standard support. Each release is identified by an increment from 0 to N. 0 refers to the original major release for that iteration. All releases will be called Amazon Linux 2023. When the next version of Amazon Linux is released, AL2023 will enter extended support and receive updates for security updates and critical bug fixes.

For example, minor releases of AL2023 have the following format:

- 2023.0.20230301
- 2023.1.20230601
- 2023.2.20230901

The corresponding AL2023 AMIs have the following format:

- al2023-ami-2023.0.20230301.0-kernel-6.1-x86_64
- al2023-ami-2023.1.20230601.0-kernel-6.1-x86_64
- al2023-ami-2023.2.20230901.0-kernel-6.1-x86_64

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

- al2023-ami-2023.0.**20230301**.0-kernel-6.1-x86_64
- al2023-ami-2023.0.**20230410**.0-kernel-6.1-x86_64
- al2023-ami-2023.0.**20230520**.0-kernel-6.1-x86_64

The recommended method for identifying an AL2 or AL2023 instance starts with reading the Common Platform Enumeration (CPE) string from `/etc/system-release-cpe`. Then, split the string into its fields. Finally, read the platform and version values.

AL2023 also introduces new files for platform identification:

- `/etc/amazon-linux-release` symlinks to `/etc/system-release`
- `/etc/amazon-linux-release-cpe` symlinks to `/etc/system-release-cpe`

These two files indicate that an instance is Amazon Linux. There's no need to read a file or split the string into fields, unless you want to know the specific platform and version values.

Performance and operational optimizations

Amazon Linux 6.1 kernel

- AL2023 uses the latest drivers for Elastic Network Adapter (ENA) and Elastic Fabric Adapter (EFA) devices. AL2023 focuses on performance and functionality backports for hardware in Amazon EC2 infrastructure.
- Kernel live patching is available for the `x86_64` and `aarch64` instance types. This reduces the need for frequent reboots.
- All kernel build and runtime configurations include many of the same performance and operational optimizations of AL2.

Base toolchain selection and default build flags

- AL2023 packages are built with compiler optimizations (`-O2`) enabled by default
- AL2023 packages are built requiring `x86-64v2` for `x86-64` systems (`-march=x86-64-v2`), and Graviton 2 or higher for `aarch64` (`-march=armv8.2-a+crypto -mtune=neoverse-n1`).
- AL2023 packages are built with auto-vectorization enabled (`-ftree-vectorize`).
- AL2023 packages are built with Link Time Optimization (LTO) enabled.
- AL2023 uses the updated versions of Rust, Clang/LLVM, and Go.

Package selection and versions

- Select backports to major system components include several performance improvements for running on Amazon EC2 infrastructure, especially Graviton instances.
- AL2023 is integrated with several AWS services and features. This includes the AWS CLI, SSM Agent, Amazon Kinesis Agent, and CloudFormation.
- AL2023 uses Amazon Corretto as the Java Development Kit (JDK).
- AL2023 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 AL2023 AMI and container images are frequently updated to support patching instance replacement.
- Kernel updates are included in AL2023 AMI updates. This means that you don't need to use commands such as `yum update` and `reboot` to update your kernel.
- In addition to the standard AL2023 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, AL2023 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.

Relationship to Fedora

AL2023 maintains its own release and support lifecycles independent of Fedora. AL2023 provides updated versions of open-source software, a larger variety of packages, and frequent releases. This preserves the familiar RPM-based operating systems.

The Generally Available (GA) version of AL2023 isn't directly comparable to any specific Fedora release. The AL2023 GA version includes components from Fedora 34, 35, and 36. Some of the components are the same as the components in Fedora and some are modified. Other components

more closely resemble the components in CentOS Stream 9 or were developed independently. The Amazon Linux kernel is sourced from the long-term support options that are on kernel.org, chosen independently from Fedora.

Customized cloud-init

The cloud-init package is an open-source application that bootstraps Linux images in a cloud computing environment. For more information, see [cloud-init Documentation](#).

AL2023 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 Amazon Machine Images (AMIs) for many use cases and configure them dynamically when you start an instance. AL2023 also uses cloud-init to configure the `ec2-user` account.

AL2023 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
 - By default, if the `ephemeral0` instance store volume is present and contains a valid file system, the instance store volume is mounted at `/media/ephemeral0`. Otherwise, it's not mounted.

- By default, for the `m1.small` and `c1.medium` instance types, all swap volumes that are associated with the instance are mounted.
- You can override the default instance store volume mount with the following cloud-init directive:

```
#cloud-config
mounts:
- [ ephemeral0 ]
```

For more control over mounts, see [Mounts](#) in the cloud-init documentation.

- When an instance launches, instance store volumes that support TRIM aren't formatted. Before you can mount instance store volumes, you must partition and format instance store volumes.

For more information, 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, see [Disk Setup](#) in the cloud-init documentation.

For information about using cloud-init with SELinux, see [Use cloud-init to enable enforcing mode](#).

For information about cloud-init user-data formats, see [User-Data Formats](#) in the cloud-init documentation.

Security updates and features

AL2023 provides many security updates and solutions.

Topics

- [Manage updates](#)
- [Security in the cloud](#)
- [SELinux modes](#)
- [Compliance program](#)
- [SSH server default](#)
- [Major features of OpenSSL 3](#)

Manage updates

Apply security updates using DNF and repository versions. For more information, see [Manage package and operating system updates in AL2023](#).

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 Amazon Linux 2023](#).

SELinux modes

By default, SELinux is enabled and set to permissive mode in AL2023. In permissive mode, permission denials are logged but not enforced.

The SELinux policies define permissions for users, processes, programs, files, and devices. With SELinux, you can choose one of two policies. The policies are targeted or multi-level security (MLS).

For more information about SELinux modes and policy, see [Setting SELinux modes for AL2023](#) and [the SELinux Project Wiki](#).

Compliance program

Independent auditors assess the security and compliance of AL2023 along with many AWS compliance programs.

SSH server default

AL2023 includes OpenSSH 8.7. OpenSSH 8.7 by default disables the `ssh-rsa` key exchange algorithm. For more information, see [Default SSH server configuration](#).

Major features of OpenSSL 3

- The Certificate Management Protocol (CMP, RFC 4210) includes both CRMF (RFC 4211) and HTTP transfer (RFC 6712).
- A HTTP or HTTPS client in libcrypto supports GET and POST actions, redirection, plain and ASN.1-encoded content, proxies, and timeouts.

- The `EVP_KDF` works with Key Derivation Functions.
- The `EVP_MAC` API works with MACs.
- Linux Kernel TLS support.

For more information, see the [OpenSSL migration guide](#).

Networking service

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 generates interface-specific configurations in the `/run/systemd/network` directory. These configurations enable both IPv4 and IPv6 networking on interfaces when they're attached to an instance. These configurations also install 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 Elastic Network Interface (ENI) from the associated addresses or prefixes. For more information about using ENI, see [Using ENI](#) 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`.

The [systemd.network](#) documentation 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.

AL2023 instance interface names are generated using the `systemd` slot naming scheme. For more information, see [systemd.net naming scheme](#).

Additionally, AL2023 uses the `fq_code1` active queue management network transmission scheduling algorithm by default. For more information, see [CoDel overview](#).

Core toolchain packages glibc, gcc, binutils

A subset of packages in Amazon Linux is designated as core toolchain packages. As a major part of AL2023, core packages receive five years of support. We might change the version of a package, but long-term support applies to the package included in the Amazon Linux release.

These three core packages provide a system toolchain that's used to build most software in the Amazon Linux distribution.

Package	Definition	Purpose
glibc 2.34	System C library	Used by most binary programs that provide standard functions and by the interface between programs and the kernel.
gcc 11.2	gcc compiler suite	Compiles C, C++, Fortran.
binutils 2.35	Assembler and linker plus other binary tools	Manipulates or inspects binary programs.

We recommend that updates to any glibc libraries are followed by a reboot. For updates to packages that control services, it might be sufficient to restart the services to pick up the updates. However, a system reboot ensures that all previous package and library updates are complete.

Package management tool

The default software package management tool in AL2023 is DNF. DNF is the successor to YUM, the package management tool in AL2.

DNF is similar to YUM in its usage. Many DNF commands and command options are the same as YUM commands. In a Command Line Interface (CLI) command, in most cases `dnf` replaces `yum`.

For example, for the following AL2 `yum` commands:

```
$ sudo yum install packagename
$ sudo yum search packagename
```

```
$ sudo yum remove packagename
```

In AL2023, they become the following commands:

```
$ sudo dnf install packagename  
$ sudo dnf search packagename  
$ sudo dnf remove packagename
```

In AL2023 the yum command is still available, but as a pointer to the dnf command. So, when the yum command is used in the shell or in a script, all commands and options are the same as the DNF CLI. For more information about the differences between the YUM CLI and the DNF CLI, see [Changes in DNF CLI compared to YUM](#).

For a complete reference of commands and options for the dnf command, refer to the man page `man dnf`. For more information, see [DNF Command Reference](#).

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.

Default disabling of `ssh-rsa` signatures

AL2023 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.

The default configuration accepts any of these key exchange algorithms:

- `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`

By default, AL2023 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
```

For more information about managing host keys, see [Amazon Linux Host keys](#).

Deprecated Functionality in AL2023

Functionality deprecated in AL2 and not present in AL2023 is documented here. This is functionality such as features and packages that are present in AL2, but not in AL2023 and will not be added to AL2023. For more information about how the long the functionality is supported in AL2, see [Deprecated functionality in AL2](#).

There is also functionality in AL2023 which is deprecated, and will be removed in a future release. This chapter describes what that functionality is, when it no longer supported, and when it will be removed from Amazon Linux. Understanding the deprecated functionality will help you deploy AL2023 as well as prepare for the next major version of Amazon Linux.

Topics

- [compat- packages](#)
- [Deprecated functionality discontinued in AL1, removed in AL2](#)
- [Functionality deprecated in AL2 and removed in AL2023](#)
- [Deprecated in AL2023](#)

compat- packages

Any packages in AL2 with the prefix of `compat-` are provided for binary compatibility with older binaries that have not yet been rebuilt for modern versions of the package. Each new major version of Amazon Linux will not carry forward any `compat-` package from prior releases.

All `compat-` packages in a release of Amazon Linux (e.g. AL2) are deprecated, and not present in the subsequent version (e.g. AL2023). We strongly recommend that software is rebuilt against updated versions of the libraries.

Deprecated functionality discontinued in AL1, removed in AL2

This section describes functionality that is available in AL1, and is no longer available in AL2.

Note

As part of the maintenance support phase of AL1, some packages had an end-of-life (EOL) date earlier than the EOL of AL1. For more information, see [AL1 Package support statements](#).

Note

Some AL1 functionality was discontinued in earlier releases. For information, see the [AL1 Release Notes](#).

Topics

- [32-bit x86 \(i686\) AMIs](#)
- [aws-apitools-* replaced by AWS CLI](#)
- [systemd replaces upstart in AL2](#)

32-bit x86 (i686) AMIs

As part of the [2014.09 release of AL1](#), Amazon Linux announced that it would be the last release to produce 32-bit AMIs. Therefore, starting from the [2015.03 release of AL1](#), Amazon Linux no longer supports running the system in 32-bit mode. AL2 offers limited runtime support for 32-bit binaries on x86-64 hosts and does not provide development packages to enable the building of new 32-bit binaries. AL2023 no longer includes any 32-bit user space packages. We recommend that users complete their transition to 64-bit code before migrating to AL2023.

If you need to run 32-bit binaries on AL2023, it is possible to use the 32-bit userspace from AL2 inside an AL2 container running on top of AL2023.

aws-apitools-* replaced by AWS CLI

Before the release of the AWS CLI in September 2013, AWS made a set of command line utilities available, implemented in Java, which allowed users to make Amazon EC2 API calls. These tools were discontinued in 2015, with the AWS CLI becoming the preferred way to interact with Amazon EC2 APIs from the command line. The set of command line utilities includes the following `aws-apitools-*` packages.

- `aws-apitools-as`
- `aws-apitools-cfn`
- `aws-apitools-common`
- `aws-apitools-ec2`
- `aws-apitools-elb`
- `aws-apitools-mon`

Upstream support for the `aws-apitools-*` packages ended in March of 2017. Despite the lack of upstream support, Amazon Linux continued to ship some of these command line utilities, such as `aws-apitools-ec2`, to provide backward compatibility for users. The AWS CLI is a more robust and complete tool than the `aws-apitools-*` packages as it is actively maintained and provides a means of using all AWS APIs.

The `aws-apitools-*` packages were deprecated in March 2017 and will not be receiving further updates. All users of any of these packages should migrate to the AWS CLI as soon as possible. These packages are not present in AL2023.

AL1 also provided the `aws-apitools-iam` and `aws-apitools-rds` packages, which were deprecated in AL1, and are not present in Amazon Linux from AL2 onward.

systemd replaces upstart in AL2

AL2 was the first Amazon Linux release to use the `systemd` init system, replacing `upstart` in AL1. Any `upstart` specific configuration must be changed as part of the migration from AL1 to a newer version of Amazon Linux. It is not possible to use `systemd` on AL1, so moving from `upstart` to `systemd` can only be done as part of moving to a more recent major version of Amazon Linux such as AL2 or AL2023.

Functionality deprecated in AL2 and removed in AL2023

This section describes functionality that is available in AL2, and no longer available in AL2023.

Topics

- [32-bit x86 \(i686\) Packages](#)
- [aws-apitools-* replaced by AWS CLI](#)

- [awslogs deprecated in favor of unified Amazon CloudWatch Logs agent](#)
- [bzip2 revision control system](#)
- [cgroup v1](#)
- [log4j hotpatch \(log4j-cve-2021-44228-hotpatch\)](#)
- [lsb_release and the system-lsb-core package](#)
- [mccrypt](#)
- [OpenJDK 7 \(java-1.7.0-openjdk\)](#)
- [Python 2.7](#)
- [rsyslog-openssl replaces rsyslog-gnutls](#)
- [Network Information Service \(NIS\) / yp](#)
- [Multiple domain names in Amazon VPC create-dhcp-options](#)
- [Sun RPC in glibc](#)
- [OpenSSH key fingerprint in audit log](#)
- [ld.gold Linker](#)
- [ping6](#)
- [ftp Package](#)

32-bit x86 (i686) Packages

As part of the [2014.09 release of AL1](#), we announced that it would be the last release to produce 32-bit AMIs. Therefore, starting from the [2015.03 release of AL1](#), Amazon Linux no longer supports running the system in 32-bit mode. AL2 provides limited runtime support for 32-bit binaries on x86-64 hosts and doesn't provide development packages to enable the building of new 32-bit binaries. AL2023 no longer includes any 32-bit userspace packages. We recommend customers complete their transition to 64-bit code.

If you need to run 32-bit binaries on AL2023, it is possible to use the 32-bit userspace from AL2 inside an AL2 container running on top of AL2023.

aws-apitools-* replaced by AWS CLI

Prior to release of the AWS CLI in September 2013, AWS made a set of command line utilities available, implemented in Java, which allowed customers to make Amazon EC2 API calls. These

tools were deprecated in 2015, with the AWS CLI becoming the preferred way to interact with Amazon EC2 APIs from the command line. This includes the following `aws-apitools-*` packages.

- `aws-apitools-as`
- `aws-apitools-cfn`
- `aws-apitools-common`
- `aws-apitools-ec2`
- `aws-apitools-elb`
- `aws-apitools-mon`

Upstream support for the `aws-apitools-*` packages ended in March of 2017. Despite the lack of upstream support, Amazon Linux continued to ship some of these command line utilities (such as `aws-apitools-ec2`) in order to provide backwards compatibility for customers. The AWS CLI is a more robust and complete tool than the `aws-apitools-*` packages as it is actively maintained and provides a means of using all AWS APIs.

The `aws-apitools-*` packages were deprecated in March 2017 and will not be receiving further updates. All users of any of these packages should migrate to the AWS CLI as soon as possible. These packages are not present in AL2023.

awslogs deprecated in favor of unified Amazon CloudWatch Logs agent

The [awslogs](#) package is deprecated in AL2 and is no longer present in AL2023. It is replaced by the [unified CloudWatch Logs agent](#), available in the `amazon-cloudwatch-agent` package. For more information, see the [Amazon CloudWatch Logs User Guide](#).

bzr revision control system

The [GNU Bazaar](#) (`bzr`) revision control system is discontinued in AL2 and no longer present in AL2023.

Users of `bzr` are advised to migrate their repositories to `git`.

cgroup v1

AL2023 moves to Unified Control Group hierarchy (`cgroup v2`), whereas AL2 uses `cgroup v1`. As AL2 doesn't support `cgroup v2`, this migration needs to be completed as part of moving to AL2023.

log4j hotpatch (log4j-cve-2021-44228-hotpatch)

Note

The log4j-cve-2021-44228-hotpatch package is deprecated in AL2 and removed in AL2023.

In response to [CVE-2021-44228](#), Amazon Linux released an RPM packaged version of the [Hotpatch for Apache Log4j](#) for AL1 and AL2. In the [announcement of the addition of the hotpatch to Amazon Linux](#), we noted that "Installing the hotpatch is not a replacement for updating to a log4j version that mitigates CVE-2021-44228 or CVE-2021-45046."

The hotpatch was a mitigation to allow time to patch log4j. The first general availability release of AL2023 was 15 months after [CVE-2021-44228](#), so AL2023 doesn't ship with the hotpatch (enabled or not).

Customers running their own log4j versions on Amazon Linux are advised to ensure they have updated to versions not affected by [CVE-2021-44228](#) or [CVE-2021-45046](#).

lsb_release and the system-lsb-core package

Historically, some software invoked the lsb_release command (provided in AL2 by the system-lsb-core package) to get information about the Linux distribution that it was being run on. The Linux Standards Base (LSB) introduced this command and Linux distributions adopted it. Linux distributions have evolved to use the simpler standard of holding this information in /etc/os-release and other related files.

The os-release standard comes out of systemd. For more information, see [systemd os-release documentation](#).

AL2023 doesn't ship with the lsb_release command, and doesn't include the system-lsb-core package. Software should complete the transition to the os-release standard to maintain compatibility with Amazon Linux and other major Linux distributions.

mcrypt

The mcrypt library and associated PHP extension was deprecated in AL2, and is no longer present in AL2023.

Upstream PHP [deprecated the mcrypt extension in PHP 7.1](#) which was first released in December 2016 and had its final release in October 2019.

The upstream mcrypt library [last made a release in 2007](#), and has not made the migration from cvs revision control that [SourceForge required for new commits in 2017](#), with the most recent commit (and only for 3 years prior) being from 2011 removing the mention of the project having a maintainer.

Any remaining users of mcrypt are advised to port their code to OpenSSL, as mcrypt will not be added to AL2023.

OpenJDK 7 (java-1.7.0-openjdk)

Note

AL2023 provides several versions of [Amazon Corretto](#) to support Java based workloads. The OpenJDK 7 packages are deprecated in AL2, and no longer present in AL2023. The oldest JDK available in AL2023 is provided by Corretto 8.

For more information about Java on Amazon Linux, see [Java in AL2023](#).

Python 2.7

Note

AL2023 removed Python 2.7, so any OS components requiring Python are written to work with Python 3. To continue to use a version of Python provided by and supported by Amazon Linux, convert Python 2 code to Python 3.

For more information about Python on Amazon Linux, see [Python in AL2023](#).

rsyslog-openssl replaces rsyslog-gnutls

The rsyslog-gnutls package is deprecated in AL2, and no longer present in AL2023. The rsyslog-openssl package should be a drop-in replacement for any usage of the rsyslog-gnutls package.

Network Information Service (NIS) / yp

The Network Information Service (NIS), originally called Yellow Pages or YP is deprecated in AL2, and no longer present in AL2023. This includes the following packages: `ypbind`, `ypserv`, and `yp-tools`. Other packages that integrate with NIS have this functionality removed in AL2023.

Multiple domain names in Amazon VPC `create-dhcp-options`

In Amazon Linux 2, it was possible to pass multiple domain names in the `domain-name` parameter to [create-dhcp-options](#), which would result in `/etc/resolv.conf` containing something like `search foo.example.com bar.example.com`. The Amazon VPC DHCP server sends the list of provided domain names using DHCP option 15, which only supports a single domain name (see [RFC 2132 section 3.17](#)). Since AL2023 uses `systemd-networkd` for network configuration, which follows the RFC, this accidental feature in AL2 is not present on AL2023.

The [AWS CLI](#) and [Amazon VPC documentation](#) has this to say: "Some Linux operating systems accept multiple domain names separated by spaces. However, Windows and other Linux operating systems treat the value as a single domain, which results in unexpected behavior. If your DHCP option set is associated with a Amazon VPC that has instances running operating systems that treat the value as a single domain, specify only one domain name. "

On these systems, such as AL2023, specifying two domains using DHCP option 15 (which only allows one), and since the [space character is invalid in domain names](#), this will result in the space character being encoded as `032`, resulting in `/etc/resolv.conf` containing `search foo.exmple.com032bar.example.com`.

In order to support multiple domain names, a DHCP server should use DHCP Option 119 (see [RFC 3397, section 2](#)). See the [Amazon VPC User Guide](#) for when this is supported by the Amazon VPC DHCP server.

Sun RPC in `glibc`

The implementation of Sun RPC in `glibc` is deprecated in AL2 and removed in AL2023. Customers are advised to move to using the `libtirpc` library (available in AL2 and AL2023) if Sun RPC functionality is required. Adopting `libtirpc` also enables applications to support IPv6.

This change reflects the broader community's adoption of upstream `glibc` removing this functionality, for example the [Removal of Sun RPC interfaces from glibc in Fedora](#) and a [similar change in Gentoo](#).

OpenSSH key fingerprint in audit log

Later in the lifecycle of AL2, a patch was added to the OpenSSH package to emit the key fingerprint used to authenticate. This functionality is not present in AL2023.

ld.gold Linker

The `ld.gold` linker is available in AL2, and is removed in AL2023. Customers building software that explicitly references the gold linker should migrate to the regular (`ld.bfd`) linker.

The upstream [GNU Binutils release notes for version 2.44](#) (released Feb 2025) document the removal of `ld.gold`: "In a change to our previous practice, in this release the binutils-2.44.tar tarball does not contain the sources for the gold linker. This is because the gold linker is now deprecated and will eventually be removed unless volunteers step forward and offer to continue development and maintenance."

ping6

In AL2023, the regular `ping` utility natively supports IPv6, and the separate `/bin/ping6` is no longer required. In AL2023, `/usr/sbin/ping6` is a symlink to the `/usr/bin/ping` executable.

This change follows the broader community's adoption of newer `iputils` versions which provide this functionality, for example the [Ping IPv6 change in Fedora](#).

ftp Package

The `ftp` package in AL2 is no longer available in Amazon Linux starting with AL2023. This decision was made as part of our ongoing commitment to security, maintainability, and modern software development practices. As part of (or before) migrating to AL2023, we recommend migrating any use of the legacy `ftp` package to one of its alternatives.

Background

The legacy `ftp` package has not been actively maintained upstream for many years. The last significant update to the source code occurred in the early 2000s, and the original source repository is no longer available. While some Linux distributions have carried patches for security vulnerabilities, the codebase remains largely unmaintained.

Recommended Alternatives

AL2023 provides several modern, actively maintained alternatives for FTP functionality:

`lftp` (available in AL2 and AL2023)

A sophisticated file transfer program supporting FTP, HTTP, SFTP, and other protocols. It offers more features than the traditional `ftp` client and is actively maintained.

Install with: **`dnf install lftp`**

`curl` (available in AL2 and AL2023)

A versatile command-line tool for transferring data with URLs, supporting FTP, FTPS, HTTP, HTTPS, and many other protocols.

Available by default in AL2023 via the `curl-minimal` package. For more extensive protocol support, you can optionally upgrade to `curl-full` using **`dnf swap curl-minimal curl-full`**.

`wget` (available in AL2 and AL2023)

A non-interactive command-line utility for downloading files from the web, supporting HTTP, HTTPS, and FTP protocols.

Install with: **`dnf install wget`** (not installed by default in all AL2023 images)

`sftp` (available in AL2 and AL2023)

A secure file transfer protocol that operates over SSH, providing encrypted file transfers.

Available by default as part of the OpenSSH package.

Migration Considerations

If your applications or scripts depend on the legacy `ftp` client, consider the following migration approaches:

1. **Update scripts to use modern alternatives:** Modify your scripts to use `lftp`, `curl`, `wget`, or `sftp` instead of the legacy `ftp` client.
2. **Review package dependencies:** Some applications may list the `ftp` package as a dependency in their package metadata, even though they have long since migrated to using modern protocols internally. In these cases, the application may work correctly on AL2023 despite the lack of `/usr/bin/ftp` from the `ftp` package. Review your application's actual requirements rather than relying solely on stated dependencies.

3. **Update application dependencies:** For applications you maintain that still declare a dependency on the `ftp` package but don't actually use it, update the package metadata to remove this unnecessary dependency.

Security Considerations

The FTP protocol transmits data, including authentication credentials, in plaintext. For security-sensitive applications, we strongly recommend using encrypted alternatives such as SFTP or HTTPS which are supported by the recommended alternative tools.

Deprecated in AL2023

This section describes functionality that exists in AL2023 and is likely to be removed in a future version of Amazon Linux. Each section will describe what the functionality is and when it is expected to be removed from Amazon Linux.

Note

This section will be updated over time as the Linux ecosystem evolves and future major versions of Amazon Linux are closer to release.

Topics

- [32bit x86 \(i686\) runtime support](#)
- [aspell](#)
- [Berkeley DB \(libdb\)](#)
- [cron](#)
- [IMDSv1](#)
- [pcre version 1](#)
- [System V init \(sysvinit\)](#)
- [EOL Packages are deprecated](#)

32bit x86 (i686) runtime support

AL2023 retains the ability to run 32bit x86 (i686) binaries. It is likely that the next major version of Amazon Linux will no longer support running 32bit user space binaries.

aspell

While AL2023 ships with the `aspell` package, it is deprecated and will be removed in the next major release of Amazon Linux. Customers are advised to 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`), and the library will be removed in the next major release of Amazon Linux.

Note

The `dnf` package manager in AL2023 retains read-only support for a Berkeley DB (BDB) format `rpm` database. This support will be removed in the next major release of Amazon Linux.

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.

In AL2023, 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, possibly the next major version, 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 will be removed in the next major release of Amazon Linux. 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 will retain backwards compatibility with System V service (init) scripts, but users are encouraged to migrate to using native systemd unit files in order to be prepared for when support for System V service (init) scripts is removed from Amazon Linux, likely in the next major release.

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 [the section called “PHP”](#) and [the section called “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 [the section called “PHP”](#) and [the section called “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.

Comparing AL2 and AL2023

The following topics describe key differences between AL2 and AL2023.

For more information on functionality deprecated in AL1, AL2, and AL2023, see [Deprecated Functionality in AL2023](#).

Topics

- [Added, upgraded, and removed packages](#)
- [Support for each release](#)
- [Naming and versioning changes](#)
- [Optimizations](#)
- [Sourced from multiple upstreams](#)
- [Networking system service](#)
- [Package manager](#)
- [Using cloud-init](#)
- [Graphical desktop support](#)
- [Compiler Triplet](#)
- [32bit x86 \(i686\) Packages](#)
- [lsb_release and the system-lsb-core package](#)
- [Extra Packages for Enterprise Linux \(EPEL\)](#)
- [Python 2.7 has been replaced with Python 3](#)
- [Security updates](#)
- [Deterministic upgrades for stability](#)
- [gp3 as default Amazon EBS volume type](#)
- [Unified Control Group hierarchy \(cgroup v2\)](#)
- [systemd timers replace cron](#)
- [Improved toolchain: gcc, binutils, and glibc](#)
- [systemd journal replaces rsyslog](#)
- [Minimized package dependencies](#)

- [Amazon Corretto as the default JVM](#)
- [AWS CLI v2](#)
- [UEFI Preferred and Secure Boot](#)
- [SSH server default configuration changes](#)
- [AL2023 kernel changes from AL2](#)
- [/tmp is now tmpfs](#)
- [AMI and Container Image changes](#)
- [Comparing packages installed on Amazon Linux 2 and Amazon Linux 2023 AMIs](#)
- [Comparing packages installed on Amazon Linux 2 and Amazon Linux 2023 Minimal AMIs](#)
- [Comparing packages installed on Amazon Linux 2 and Amazon Linux 2023 base container images](#)

Added, upgraded, and removed packages

AL2023 contains thousands of software packages available for use. For a full list of all packages added, upgraded, or removed in AL2023 when compared to prior Amazon Linux versions, see [Package changes in AL2023](#).

To request a package to be added or changed in AL2023, file an issue in the [amazon-linux-2023 repo](#) on GitHub.

Support for each release

For AL2023, we offer five years of support.

For more information, see [Release cadence](#).

Naming and versioning changes

AL2023 supports the same mechanisms that AL2 supports for platform identification. AL2023 also introduces new files for platform identification.

For more information, see [Naming and versioning](#).

Optimizations

AL2023 optimizes boot time to reduce the time from instance launch to running the customer workload. These optimizations span the Amazon EC2 instance kernel configuration, `cloud-init` configurations, and features that are built into packages in the OS such as `askmod` and `systemd`.

For more information about optimizations, see [Performance and operational optimizations](#).

Sourced from multiple upstreams

AL2023 is RPM-based and includes components sourced from multiple versions of Fedora and other distributions, such as CentOS 9 Stream. The Amazon Linux kernel is sourced from the long-term support (LTS) releases directly from kernel.org, chosen independently from other distributions.

For more information, see [Relationship to Fedora](#).

Networking system service

The `systemd-networkd` system service manages the network interfaces in AL2023. This is a change from AL2, which uses ISC `dhclient` or `dhc11ent`.

For more information, see [Networking service](#).

Package manager

The default software package management tool on AL2023 is DNF. DNF is the successor to YUM, the package management tool in AL2.

For more information, see [Package management tool](#).

Using cloud-init

In AL2023, `cloud-init` manages the package repository. By default, in earlier versions of Amazon Linux, `cloud-init` installed security updates. This isn't the default for AL2023. The new deterministic upgrading features for updating `releasetag` at launch describe the AL2023 way to enable package updates at launch. For more information, see [Manage package and operating system updates in AL2023](#) and [Deterministic upgrades for stability](#).

With AL2023, you can use cloud-init with SELinux. For more information, see [Use cloud-init to enable enforcing mode](#).

Cloud-init loads configuration content with cloud-init from remote locations using HTTP(S). In earlier versions, Amazon Linux doesn't alert you when remote resources are unavailable. In AL2023, unavailable remote resources creates a fatal error and fails the cloud-init execution. This change in behavior from AL2, provides a safer "fail closed" default behavior.

For more information, see [Customized cloud-init](#) and the [cloud-init Documentation](#).

Graphical desktop support

AL2023 features a GNOME-based graphical desktop environment as of release 2023.7, replacing the MATE desktop used in AL2. This version provides users with a different desktop experience while maintaining AL2023's cloud-optimized performance. The GNOME desktop environment offers various customization options, system integration features, and a distinct user interface design, providing users with an alternative to the previous MATE desktop environment. See the [AL2023 Graphical Desktop](#) page for more details.

Compiler Triplet

AL2023 sets the compiler triplet for GCC and LLVM to indicate that amazon is the vendor.

Thus, the AL2 `aarch64-redhat-linux-gcc` becomes `aarch64-amazon-linux-gcc` on AL2023.

This should be completely transparent for most users, and might only affect those who are building compilers on AL2023.

32bit x86 (i686) Packages

As part of the [2014.09 release of AL1](#) it was announced that it would be the last release to produce 32-bit AMIs. Thus, from the [2015.03 release of AL1](#), Amazon Linux no longer supported running the system in 32-bit mode. AL2 offered limited runtime support for 32bit binaries on x86-64 hosts, and did not provide development packages to enable the building of new 32-bit binaries. AL2023 no longer includes any 32bit userspace packages. We recommend that you complete your transition to 64-bit code.

If you need to run 32-bit binaries on AL2023, it is possible to use the 32-bit user-space from AL2 inside an AL2 container running on top of AL2023.

lsb_release and the system-lsb-core package

Historically, some software invoked the `lsb_release` command (provided in AL2 by the `system-lsb-core` package) to get information about the Linux distribution that it was being run on. The Linux Standards Base (LSB) introduced this command and Linux distributions adopted it. Linux distributions have evolved to use the simpler standard of holding this information in `/etc/os-release` and other related files.

The `os-release` standard comes out of `systemd`. For more information, see [systemd os-release documentation](#).

AL2023 doesn't ship with the `lsb_release` command, and doesn't include the `system-lsb-core` package. Software should complete the transition to the `os-release` standard to maintain compatibility with Amazon Linux and other major Linux distributions.

Extra Packages for Enterprise Linux (EPEL)

Warning

The AL2 `epe1` Extra enabled the third party EPEL7 repository. As of 2024-06-30 the third-party EPEL7 repository is *no longer being maintained*.

This third-party repository will have *no future updates*. This means there will be *no security fixes* for packages in the *EPEL* repository.

This section will cover options in AL2023 for packages found in EPEL.

Extra Packages for Enterprise Linux (EPEL) is a project in the Fedora community with the objective of creating a large array of packages for enterprise-level Linux operating systems. The project has primarily produced RHEL and CentOS packages. AL2 features a high level of compatibility with CentOS 7. As a result, many EPEL7 packages work on AL2.

There are no EPEL versions that are binary compatible with AL2023. However, customers that want to use their EPEL7 packages in AL2023 have a few options. Some EPEL packages have alternatives in AL2023, while others are provided as part of [Supplementary Packages for Amazon Linux](#).

⚠ Warning

Only add repositories designed to be used with AL2023.

While repositories designed for other distributions may work today, there is no guarantee they will continue to do so with any package update in AL2023 or the repository not designed for use with AL2023.

This page provides information about the EPEL7 packages used by customers on AL2 and their AL2023 counterparts.

For the rest of the packages, customers might be able to use Supplementary Packages for Amazon Linux (SPAL). SPAL provides thousands of EPEL9 packages, built specifically for Amazon Linux 2023, but these packages are not covered by AWS Enterprise Support. This means CVEs are not being tracked for SPAL packages, and patches are only provided when available upstream.

⚠ Important

Consult documentation of [Supplementary Packages for Amazon Linux](#) before using it.

Topics

- [axel - HTTP/FTP client](#)
- [brotli and libbrotli - compression](#)
- [collectd - Statistics collection daemon](#)
- [cpulimit - CPU Usage limiter](#)
- [exim - mail transfer agent](#)
- [fuse3 - File System in Userspace \(FUSE\) v3](#)
- [ganglia - Distributed Monitoring System](#)
- [git-lfs - version control large files with Git](#)
- [haveged - an entropy source using the HAVEGE algorithm](#)
- [inotify-tools - inotify command line tools](#)
- [iperf - TCP/UDP Performance benchmark](#)
- [jemalloc - alternative malloc implementation](#)
- [libbsd - BSD-compatible function library](#)

- [libserf - HTTP Client Library](#)
- [libzstd - zstd compression library](#)
- [lighttpd web server](#)
- [lshell - a restricted shell](#)
- [monit - process, file, directory, and devices monitor](#)
- [nodejs](#)
- [perl-Config-General](#)
- [python2-lockfile - file locking](#)
- [python2-rsa - pure Python RSA](#)
- [python2-simplejson - JSON routines for Python 2](#)
- [rkhunter - Rootkit Hunter](#)
- [rssh - a restricted shell for use with OpenSSH](#)
- [sscg - self-signed SSL certificate generator](#)
- [stress - Stress test](#)
- [stress-ng - Stress test](#)
- [tmpwatch - removes files based on last accessed time](#)
- [xmlstarlet - command line XML utilities](#)

axel - HTTP/FTP client

The `axel` package was in EPEL7, and has not ever shipped as part of Amazon Linux. Alternatives available in AL2023 are `curl` and `wget`.

Warning

The `-S` option to `axel` uses an *unencrypted* http connection to discover mirrors for a file.

It is highly recommended to migrate any use of `axel` over to either `curl` or `wget`.

brotli and libbrotli - compression

The `brotli` and `libbrotli` packages were in EPEL7, while just the `brotli` package was available in AL2 core.

Both the `brotli` and `libbrotli` packages are included in AL2023.

The `brotli` package can be installed on AL2023 with the following command:

```
[ec2-user ~]$ sudo dnf install brotli
```

The `libbrotli` package can be installed on AL2023 with the following command:

```
[ec2-user ~]$ sudo dnf install libbrotli
```

collectd - Statistics collection daemon

The `collect` package was in EPEL7, and was also available in the `collectd` and `collectd-python3` AL2 Extras.

The `collectd` package is included in AL2023 and can be installed by running the following command:

```
[ec2-user ~]$ sudo dnf install collectd
```

cpulimit - CPU Usage limiter

In Amazon Linux 2023, `systemd` provides functionality to limit the CPU usage of processes, or groups of processes. This functionality is also easy to use for any `systemd` service.

There are powerful resource control facilities provided by `systemd` which can be used to ensure any task or group of tasks is limited in what resources it can consume. For more information, see the upstream [systemd.resource-control](#) documentation, along with the [Limiting process resource usage in AL2023 using systemd](#).

exim - mail transfer agent

The `exim` package was in EPEL7, and previously available in AL1. Amazon Linux 2023 provides both the `postfix` and `sendmail` Mail Transfer Agents (MTAs).

fuse3 - File System in Userspace (FUSE) v3

The `fuse3` package (including `fuse3-libs` and `fuse3-devel`) were in EPEL7. These packages are part of AL2023, and each can be installed by running the relevant following command:

```
[ec2-user ~]$ sudo dnf install fuse3
```

```
[ec2-user ~]$ sudo dnf install fuse3-libs
```

```
[ec2-user ~]$ sudo dnf install fuse3-devel
```

ganglia - Distributed Monitoring System

The ganglia package was in EPEL7, and previously available in AL1. It was not shipped with AL2.

The upstream project had a period of inactivity where some open CVEs were not being addressed. While there has been recent activity in the upstream project, it is not planned to add ganglia to AL2023.

git-lfs - version control large files with Git

The git-lfs package was in EPEL7. In Amazon Linux 2023, the git-lfs package is included in the core repository. On AL2023, git-lfs can be installed by running the following command:

```
[ec2-user ~]$ sudo dnf install git-lfs
```

haveged - an entropy source using the HAVEGE algorithm

The haveged package was in EPEL7. Amazon Linux 2023 comes pre-configured with entropy sources, not requiring the use of haveged.

inotify-tools - inotify command line tools

The inotify-tools package was in EPEL7, and is included in AL2023.

Note

In AL2023, systemd supports path based activation which can be used for taking action on events such as when a path exists or changes.

Much of what inotify-tools is used for can now be better accomplished in a more reliable manner using systemd path activation. For more information, see [systemd.path](#).

The `inotify-tools` package is included in AL2023 and can be installed by running the following command:

```
[ec2-user ~]$ sudo dnf install inotify-tools
```

iperf - TCP/UDP Performance benchmark

The `iperf` version 2 package was in EPEL7, and was also available in the testing AL2 Extra. and was also available in AL1

Note

The `iperf3` package is also available, providing version 3 of `iperf`.

The `iperf` package is included in AL2023 and can be installed by running the following command:

```
[ec2-user ~]$ sudo dnf install iperf
```

jemalloc - alternative malloc implementation

The `jemalloc` package was in EPEL7, and was available in the `lamp-mariadb10.2-php7.2` and `mariadb10.5` AL2 Extras.

The `jemalloc` package is included in AL2023 and can be installed by running the following command:

```
[ec2-user ~]$ sudo dnf install jemalloc
```

libbsd - BSD-compatible function library

The `libbsd` package was in EPEL7, and was also available in the testing AL2 Extra.

The `libbsd` package is included in AL2023 and can be installed by running the following command:

```
[ec2-user ~]$ sudo dnf install libbsd
```

The development files for `libbsd` can be installed by running the following command.

```
[ec2-user ~]$ sudo dnf install libbsd-devel
```

libserf - HTTP Client Library

The `libserf` package was in EPEL7. The `libserf` package is provided in Amazon Linux 2023. It can be installed by running the following command:

```
[ec2-user ~]$ sudo dnf install libserf
```

libzstd - zstd compression library

The `libzstd` package was in AL2 core, as well as in EPEL7. The `libzstd` package is also part of AL2023.

```
[ec2-user ~]$ sudo dnf install libzstd
```

lighttpd web server

The `lighttpd` package was in EPEL7, and previously available in AL1. Amazon Linux 2023 provides both the Apache `httpd` and `nginx` web servers.

lshell - a restricted shell

The `lshell` package has never been shipped as part of Amazon Linux. It was available in EPEL6. The [Fedora packaging repository for lshell](#) covers [why it was not packaged](#) in EPEL7 or Fedora 30. It was also [removed from Debian](#).

The upstream `lshell` project is [no longer being actively maintained](#), and contains [known unpatched Critical CVEs](#): [CVE-2016-6902](#) and [CVE-2016-6903](#).

The alternative suggested in the Debian bug, [rssh](#) is also unmaintained upstream, with the author citing unfixable security issues as the reason.

For these reasons, adding `lshell` to AL2023 is not planned.

monit - process, file, directory, and devices monitor

In Amazon Linux 2023, `systemd` provides a wide array of functionality for monitoring, starting, stopping, and restarting services. This includes rate limiting restarts, waiting between restart

attempts, and starting another service on failure. For more information, see the [systemd.service](#) documentation.

In AL2023, systemd also supports path based activation which can be used for taking action on events such as when a path exists or changes. For more information, see [systemd.path](#).

There are common configuration options for systemd units which allow specifying dependencies, conditionals, and actions to take on success or failure. For more information, see the [systemd.unit](#) documentation.

There are powerful resource control facilities provided by systemd which can be used to ensure any monitoring task does not use excessive CPU or memory. For more information, see [systemd.resource-control](#).

nodejs

The nodejs version 16 package was in EPEL7, and nodejs is now included in AL2023. At the time of writing, both nodejs version 18 and 20 were available in AL2023. You can install nodejs 18 on AL2023 with the following command:

```
[ec2-user ~]$ sudo dnf install nodejs
```

You can install nodejs 20 on AL2023 with the following command:

```
[ec2-user ~]$ sudo dnf install nodejs20
```

perl-Config-General

The perl-Config-General package was in EPEL7, and is now included in AL2023. You can install the perl-Config-General package in AL2023 with the following command:

```
[ec2-user ~]$ sudo dnf install perl-Config-General
```

Perl modules can also be installed by asking DNF to install the package that provides a particular Perl module. With this method, you can use the more familiar Perl module name rather than the OS package name.

```
[ec2-user ~]$ sudo dnf install 'perl(Config::General)'
```

python2-lockfile - file locking

The python2-lockfile package was in EPEL7, and AL2 included a python-lockfile package. In AL2023 [Python 2.7 has been replaced with Python 3](#), so a *Python 2* variant of this package will not be added to AL2023.

The *Python 3* version of this package *is included in AL2023*. You can install the python3-lockfile package in AL2023 with one of the following commands:

```
[ec2-user ~]$ sudo dnf install python3-lockfile
```

Python modules can also be installed by asking DNF to install the package that provides a particular Python module.

```
[ec2-user ~]$ sudo dnf install 'python3dist(lockfile)'
```

python2-rsa - pure Python RSA

The python2-rsa package was in EPEL7, and AL2 included a python2-rsa package. In AL2023 [Python 2.7 has been replaced with Python 3](#), so a *Python 2* variant of this package will not be added to AL2023.

The *Python 3* version of this package *is included in AL2023*. You can install the python3-rsa package in AL2023 with one of the following commands:

```
[ec2-user ~]$ sudo dnf install python3-rsa
```

Python modules can also be installed by asking DNF to install the package that provides a particular Python module.

```
[ec2-user ~]$ sudo dnf install 'python3dist(rsa)'
```

python2-simplejson - JSON routines for Python 2

The python2-simplejson package was in EPEL7. In AL2023 [Python 2.7 has been replaced with Python 3](#), so a *Python 2* variant of this package will not be added to AL2023.

The *Python 3* version of this package *is included in AL2023*. You can install the `python3-simplejson` package in AL2023 with the following command:

```
[ec2-user ~]$ sudo dnf install python3-simplejson
```

Python modules can also be installed by asking DNF to install the package that provides a particular Python module.

```
[ec2-user ~]$ sudo dnf install 'python3dist(simplejson)'
```

rkhunter - Rootkit Hunter

The `rkhunter` package is included in AL2023 along with `chkrootkit`.

```
[ec2-user ~]$ sudo dnf install rkhunter
```

```
[ec2-user ~]$ sudo dnf install chkrootkit
```

rsch - a restricted shell for use with OpenSSH

The `rsch` package was in EPEL7. The upstream [rsch](#) package is unmaintained, with the author citing unfixable security issues as the reason.

With the author citing unfixable security issues, adding `rsch` to AL2023 is not planned.

sscg - self-signed SSL certificate generator

The `sscg` package was in AL2 core, as well as in EPEL7. The `sscg` package is also part of AL2023.

```
[ec2-user ~]$ sudo dnf install sscg
```

stress - Stress test

The `stress` package was in EPEL7, and was also available in AL1

The `stress` package is included in AL2023 and can be installed by running the following command:

```
[ec2-user ~]$ sudo dnf install stress
```

stress-ng - Stress test

The `stress-ng` package was in EPEL7, and was also available in the testing AL2 Extra.

The `stress-ng` package is included in AL2023 and can be installed by running the following command:

```
[ec2-user ~]$ sudo dnf install stress-ng
```

tmpwatch - removes files based on last accessed time

In Amazon Linux 2023, this functionality is provided by [systemd-tmpfiles](#).

xmlstarlet - command line XML utilities

The `xmlstarlet` package was in EPEL7, and is not available in AL2023.

The upstream package has not been touched in over 9 years (last touched in August 2014). For an additional four years prior (since at least July 2010), a request for a new maintainer has gone unanswered. It is for this reason that it is not planned to add `xmlstarlet` to AL2023.

Python 2.7 has been replaced with Python 3

AL2 provides support and security patches for Python 2.7 until June 2025, as part of our long-term support (LTS) commitment for AL2 core packages. This support extends beyond the upstream Python community declaration of Python 2.7 end-of-life of January 2020.

AL2 uses the `yum` package manager, which has a hard dependency on Python 2.7. In AL2023 the `dnf` package manager has migrated to Python 3, and no longer requires Python 2.7. AL2023 has completely moved to Python 3.

Note

AL2023 removed Python 2.7, so any OS components requiring Python are written to work with Python 3. To continue to use a version of Python provided by and supported by Amazon Linux, convert Python 2 code to Python 3.

For more information on Python on Amazon Linux, see [Python in AL2023](#).

Security updates

Amazon Linux 2023 improves upon the hardening present in AL2. For more information, see [Security and Compliance in Amazon Linux 2023](#). For more information on kernel hardening changes from AL2, see [Security focused kernel config changes](#).

Topics

- [SELinux](#)
- [OpenSSL 3](#)
- [IMDSv2](#)
- [Removal of log4j hotpatch \(log4j-cve-2021-44228-hotpatch\)](#)

SELinux

By default, Security Enhanced Linux (SELinux) for AL2023 is enabled and set to permissive mode. In permissive mode, permission denials are logged but not enforced.

SELinux is a security feature of the Amazon Linux kernel, which was disabled in AL2. SELinux is a collection of kernel features and utilities that provides mandatory access control (MAC) architecture into the major subsystems of the kernel.

For more information, see [Setting SELinux modes for AL2023](#).

For more information about SELinux repositories, tools, and policies, see [SELinux Notebook](#), [Types of SELinux policy](#), and [SELinux Project](#).

OpenSSL 3

AL2023 features the Open Secure Sockets Layer version 3 (OpenSSL 3) cryptography toolkit. AL2023 supports TLS 1.3 and TLS 1.2 network protocols.

By default, AL2 comes with OpenSSL 1.0.2. You can build applications against OpenSSL 1.1.1.

For more information about OpenSSL, see the [OpenSSL migration guide](#).

For more information about security, see [Security updates and features](#).

IMDSv2

By default, any instances launched with the AL2023 AMI require IMDSv2-only and your default hop limit will be set to 2 to allow for containerized workload support. This is done by setting the `imds-support` parameter to `v2.0`. For more information, see [Configure the AMI](#) in the *Amazon EC2 User Guide*.

Note

The session token's time of validity can be anywhere between 1 second and 6 hours. The addresses to direct the API requests for IMDSv2 queries are the following:

- IPv4: 169.254.169.254
- IPv6: fd00:ec2::254

You can manually override these settings and enable IMDSv1 using Instance Metadata option launch properties. You can also use IAM controls to enforce different IMDS settings. For more information about setting up and using the Instance Metadata Service, see [Use IMDSv2](#), [Configure instance metadata options for new instances](#), and [Modify instance metadata options for existing instances](#), in the *Amazon EC2 User Guide*.

Removal of log4j hotpatch (log4j - cve-2021-44228-hotpatch)

Note

AL2023 doesn't ship with the `log4j - cve-2021-44228-hotpatch` package.

In response to [CVE-2021-44228](#), Amazon Linux released an RPM packaged version of the [Hotpatch for Apache Log4j](#) for AL1 and AL2. In the [announcement of the addition of the hotpatch to Amazon Linux](#) we noted that "Installing the hotpatch is not a replacement for updating to a log4j version that mitigates CVE-2021-44228 or CVE-2021-45046."

The hotpatch was a mitigation to allow time to patch log4j. The first General Availability (GA) release of AL2023 was 15 months after [CVE-2021-44228](#), thus AL2023 doesn't ship with the hotpatch (enabled or not).

Users running their own `log4j` versions on Amazon Linux should ensure that they have updated to versions not affected by [CVE-2021-44228](#) or [CVE-2021-45046](#).

AL2023 provides guidance on [Updating AL2023](#) so that you can keep up to date with security patches. Security advisories are published on the [Amazon Linux Security Center](#).

Deterministic upgrades for stability

With the deterministic upgrades through versioned repositories feature, every AL2023 AMI by default is locked to a specific repository version. You can use deterministic upgrades to achieve greater consistency among package versions and updates. Each release, major or minor, includes a specific repository version.

New with AL2023, deterministic upgrading by default is enabled. This is an improvement over the manual, incremental method of locking that's used in AL2 and other earlier versions.

For more information, see [Deterministic upgrades through versioned repositories on AL2023](#).

gp3 as default Amazon EBS volume type

The AL2023 AMI and AL2 both use the XFS file system on the root file system. For AL2023, the `mkfs` options for the root device file system are further optimized for Amazon EC2. AL2023 also supports a number of other file systems that you can use on other volumes to meet your specific requirements.

AL2023 AMIs use Amazon EBS gp3 volumes by default, whereas AL2 AMIs use Amazon EBS gp2 volumes by default. You can change the volume type when you launch an instance.

For more information about Amazon EBS volume types, see [Amazon EBS General Purpose Volumes](#).

For more information about launching an Amazon EC2 instance, see [Launch an instance](#) in the *Amazon EC2 User Guide*.

Unified Control Group hierarchy (cgroup v2)

A Control Group (cgroup) is a Linux kernel feature to hierarchically organize processes and distribute system resources between them. Control Groups are used extensively to implement a container runtime, and by `systemd`.

AL2 supports cgroupv1, and AL2023 supports cgroupv2. This is notable if running containerized workloads, such as when [Using AL2023 based Amazon ECS AMIs to host containerized workloads](#).

Although AL2023 still includes code that can make the system run using cgroupv1, this is not a recommended or supported configuration, and will be completely removed in a future major release of Amazon Linux.

There is extensive documentation regarding the [low-level Linux Kernel interfaces](#), as well as [systemd cgroup delegation documentation](#).

A common use case outside of containers is for creating systemd units that have limits placed on the system resources they can use. For more information, see [systemd.resource-control](#).

systemd timers replace 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`.

Improved toolchain: gcc, binutils, and glibc

AL2023 includes many of the same core packages as AL2.

We updated the following three core toolchain packages for AL2023.

Package name	AL2	AL2023
glibc	2.26	2.34
gcc	7.3	11.3
binutils	2.29	2.39

For more information, see [Core toolchain packages glibc, gcc, binutils](#).

For more information about C, C++, and Fortran language runtimes, including updated default language standards, see [C, C++, and Fortran in AL2023](#).

For more information about optimizations, see [Performance and operational optimizations](#).

systemd journal replaces rsyslog

In AL2023 the logging system package has changed from AL2. AL2023 doesn't install `rsyslog` by default, so the text based log files such as `/var/log/messages` that were available in AL2 aren't available by default. The default configuration for AL2023 is `systemd-journal`, which can be examined using `journalctl`. Although `rsyslog` is an optional package in AL2023, we recommend the new `systemd` based `journalctl` interface and related packages. For more information, see the [journalctl](#) manual page.

The `systemd` journal equivalent to some commonly used `syslog` commands are covered in the following table.

AL2 syslog command	AL2023 systemd journal equivalent
<code>[ec2-user ~]\$ cat /var/log/messages</code>	<code>[ec2-user ~]\$ journalctl</code>
<code>[ec2-user ~]\$ tail -f /var/log/messages</code>	<code>[ec2-user ~]\$ journalctl -f</code>
<code>[ec2-user ~]\$ grep foo /var/log/messages</code>	<code>[ec2-user ~]\$ journalctl grep foo</code>

Minimized package dependencies

Amazon Linux 2023 minimizes the dependency graph of many packages to provide a smaller footprint for applications. Notable changes from AL2 include the `curl-minimal` and `gnupg-minimal` packages, which significantly reduce the number of required packages while retaining commonly used functionality.

Topics

- [Package changes for curl and libcurl](#)
- [GNU Privacy Guard \(GNUPG\)](#)

Package changes for curl and libcurl

AL2023 separates out the common protocols and functionality of the `curl` and `libcurl` packages into `curl-minimal` and `libcurl-minimal`. This reduces the disk, memory, and dependency footprint for most users, and is the default package for AL2023 AMIs and containers.

If the full functionality of `curl` is required, for example for `gopher://` support, run the following commands to install the `curl-full` and `libcurl-full` packages.

```
$ dnf swap libcurl-minimal libcurl-full
```

```
$ dnf swap curl-minimal curl-full
```

GNU Privacy Guard (GNUPG)

AL2023 separates out minimal and complete functionality for the `gnupg2` package into `gnupg2-minimal` and `gnupg2-full` packages. By default, only the `gnupg2-minimal` package is installed. This provides the minimal functionality required to verify the digital signatures on `rpm` packages.

For more functionality from `gnupg2`, such as the ability to download keys from a key server, ensure that the `gnupg2-full` package is installed. Run the following command to swap `gnupg2-minimal` for `gnupg2-full`.

```
$ dnf swap gnupg2-minimal gnupg2-full
```

Amazon Corretto as the default JVM

AL2023 ships with [Amazon Corretto](#) as the default (and only) Java Development Kit (JDK). All Java based packages in AL2023 are all built with Amazon Corretto 17.

If you are migrating from AL2, you can smoothly transition from the equivalent OpenJDK version on AL2 to Amazon Corretto.

AWS CLI v2

AL2023 ships with AWS CLI version 2, whereas AL2 ships with version 1 of the AWS CLI.

UEFI Preferred and Secure Boot

By default, any instances launched with the AL2023 AMI on instance types that support UEFI firmware will launch in UEFI mode. This is done by setting the Boot Mode AMI parameter to `uefi-preferred`. For more information, see [Boot Modes](#) in the *Amazon EC2 User Guide*.

On Amazon EC2 instance types that support UEFI Secure Boot, it is possible to enable Secure Boot in Amazon Linux 2023. For more information, see [UEFI Secure Boot on AL2023](#).

SSH server default configuration changes

For the AL2023 AMI, we changed the types of `sshd` host keys that we generate with the release. We also dropped some legacy key types to avoid generating them at launch time. Clients must support the `rsa-sha2-256` and `rsa-sha2-512` protocols or `ssh-ed25519` with use of an `ed25519` key. By default, `ssh-rsa` signatures are disabled.

Additionally, AL2023 configuration settings in the default `sshd_config` file contain `UseDNS=no`. This new setting means that DNS impairments are less likely to block your ability to establish `ssh` sessions with your instances. The tradeoff is that the `from=hostname.domain,hostname.domain` line entries in your `authorized_keys` files won't be resolved. Because `sshd` no longer attempts to resolve the DNS names, each comma separated `hostname.domain` value must be translated to a corresponding IP address.

For more information, see [Default SSH server configuration](#).

AL2023 kernel changes from AL2

AL2023 brings the 6.1 kernel, as well as many configuration changes to further optimize Amazon Linux for the cloud. For most users, these changes should be completely transparent.

IPv4 TTL

The TTL for IPv4 is configured via `sysctl`, with the default values being present in `/etc/sysctl.d/00-defaults.conf`. This value can be customized through the usual `sysctl` methods. For more information, see the `sysctl` man page.

AL2 set the `net.ipv4.ip_default_ttl` value to 255, while AL2023 sets it to 127. This brings Amazon Linux defaults in line with other major Linux distributions. It is not recommended to change this default without a demonstrated need to.

Security focused kernel config changes

CONFIG option	AL2/4.14 aarch64	AL2/4.14 x86_64	AL2/5.10 aarch64	AL2/5.10 x86_64	AL2023/6.1/ aarch64	AL2023/6.1/ x86_64	AL2023/6.12/ aarch64	AL2023/6.12/ x86_64
CONFIG_INET_DIAG_DESTROY	n	y	n	y	y	y	y	y
CONFIG_FAULT_INJECTION	4096	4096	4096	4096	65536	65536	65536	65536
CONFIG_VMEM	n	y	n	y	n	n	n	n
CONFIG_VPORT	n	y	n	y	n	n	n	n
CONFIG_RTIFY_SOURCE	n	y	n	y	y	y	y	y
CONFIG_RDENED_COPY_LLBACK	N/A	N/A	y	y	N/A	N/A	N/A	N/A
CONFIG_IT_ON_ARM	N/A	N/A	n	n	n	n	n	n

CONFIG option	AL2/4.14 aarch64	AL2/4.14 x86_64	AL2/5.10 aarch64	AL2/5.10 x86_64	AL2023/6.1/ aarch64	AL2023/6.1/ x86_64	AL2023/6.12/ aarch64	AL2023/6.12/ x86_64
OC_DEFAULT_ON								
CONFIG_INIT_ON_FSI_E_DEFAULT_ON	N/A	N/A	n	n	n	n	n	n
CONFIG_MMU_DEFAULT_DMA_SRICT	N/A	N/A	N/A	N/A	n	n	n	n
CONFIG_ISC_AUTOLOAD	y	y	y	y	n	n	n	n
CONFIG_HED_CORRUPT	N/A	N/A	N/A	N/A	N/A	y	N/A	y
CONFIG_HED_STACK_END_CHECK	n	y	n	y	y	y	y	y
CONFIG_SECURITY_ESG_RESTRICT	n	n	n	n	y	y	y	y

CONFIG option	AL2/4.14/aarch64	AL2/4.14/x86_64	AL2/5.10/aarch64	AL2/5.10/x86_64	AL2023/6.1/aarch64	AL2023/6.1/x86_64	AL2023/6.12/aarch64	AL2023/6.12/x86_64
CONFIG_SECURITY_SELINUX_DISABLE	y	y	y	y	n	n	N/A	N/A
CONFIG_SECURITY_SELINUX_DISABLE	N/A	N/A	y	y	y	y	y	y
CONFIG_SECURITY_SELINUX_DISABLE	n	y	y	y	y	y	y	y
CONFIG_SECURITY_SELINUX_DISABLE	n	n	y	y	y	y	y	y

x86-64 Specific Security focused kernel config changes

CONFIG option	AL2/4.14/x86_64	AL2/5.10/x86_64	AL2023/6.1/x86_64	AL2023/6.12/x86_64
CONFIG_AMD_IOMMU	y	y	y	y
CONFIG_AMD_IOMMU_V2	m	m	y	N/A

CONFIG option	AL2/4.14/ x86_64	AL2/5.10/ x86_64	AL2023/6.1/ x86_64	AL2023/6.12/ x86_64
CONFIG_RA NDOMIZE_M EMORY	N/A	y	y	y

aarch64 (ARM/Graviton) Specific Security focused kernel config changes

CONFIG option	AL2/4.14/ aarch64	AL2/5.10/ aarch64	AL2023/6.1/ aarch64	AL2023/6.12/ aarch64
CONFIG_AR M64_PTR_A UTH	N/A	y	y	y
CONFIG_AR M64_PTR_A UTH_KERNEL	N/A	N/A	y	y
CONFIG_AR M64_SW_TT BR0_PAN	y	y	y	y

/dev/mem, /dev/kmem and /dev/port

Amazon Linux 2023 disables /dev/mem, and /dev/port (CONFIG_DEVMEM and CONFIG_DEVPORT) completely, building on the restrictions already in place in AL2.

The /dev/kmem code was completely removed from Linux in the 5.13 kernel, and while it was disabled in AL2, it is now not applicable to AL2023.

This option is one of the [Kernel Self Protection Project Recommended Settings](#).

FORTIFY_SOURCE

AL2023 enables `CONFIG_FORTIFY_SOURCE` on all supported architectures. This feature is a security hardening feature. Where the compiler can determine and validate the buffer sizes, this feature can detect buffer overflows in common string and memory functions.

This option is one of the [Kernel Self Protection Project Recommended Settings](#).

Line Discipline autoload (`CONFIG_LDISC_AUTOLOAD`)

The AL2023 kernel will not automatically load line disciplines, such as by software using the `TIOCSETD ioctl`, unless the request comes from a process with the `CAP_SYS_MODULE` permissions.

This option is one of the [Kernel Self Protection Project Recommended Settings](#).

dmesg access for unprivileged users (`CONFIG_SECURITY_DMESG_RESTRICT`)

By default, AL2023 does not allow unprivileged users access to `dmesg`.

This option is one of the [Kernel Self Protection Project Recommended Settings](#).

SELinux `selinuxfs` disable

AL2023 disables the deprecated `CONFIG_SECURITY_SELINUX_DISABLE` kernel option, which enabled a runtime method of disabling SELinux prior to policy being loaded.

This option is one of the [Kernel Self Protection Project Recommended Settings](#).

Other kernel configuration changes

CONFIG option	AL2/4.14 aarch64	AL2/4.14 x86_64	AL2/5.10 aarch64	AL2/5.10 x86_64	AL2023/6.1/ aarch64	AL2023/6.1/ x86_64	AL2023/6.12/ aarch64	AL2023/6.12/ x86_64
CONFIG_I	100	250	100	250	100	100	100	100
CONFIG_I	4096	8192	4096	8192	4096	8192	4096	8192
_CPUS								

CONFIG option	AL2/4.14 aarch64	AL2/4.14 x86_64	AL2/5.10 aarch64	AL2/5.10 x86_64	AL2023/6.1/ aarch64	AL2023/6.1/ x86_64	AL2023/6.12/ aarch64	AL2023/6.12/ x86_64
CONFIG_NIC_ON_OFF	y	n	y	n	y	y	y	y
CONFIG_NIC_ON_OFF_PS_VALUE	1	0	1	0	1	1	1	1
CONFIG_PANIC_ON_OOPS	m	m	m	m	n	n	n	n
CONFIG_IP	m	m	m	m	n	n	n	n
CONFIG_N_PV	N/A	y	N/A	n	N/A	n	N/A	n

CONFIG_HZ

AL2023 sets CONFIG_HZ to 100 on both x86-64 and aarch64 platforms.

CONFIG_NR_CPUS

AL2023 sets CONFIG_NR_CPUS to a number closer to the maximum number of CPU cores found in Amazon EC2.

Panic on OOPS

The AL2023 kernel will panic when it oopses. This feature is equivalent to booting with `oops=panic` on the kernel command line.

A kernel oops is where the kernel has detected an internal error which may affect the further reliability of the system.

PPP and SLIP Support

AL2023 does not support the PPP or SLIP protocols.

Xen PV Guest Support

AL2023 does not support running as a Xen PV guest.

Kernel Filesystem support

There have been several changes in the file systems that the kernel in AL2 will support mounting, along with changes in the partitioning schemes that the kernel will parse.

CONFIG option	AL2/4.14 aarch64	AL2/4.14 x86_64	AL2/5.10 aarch64	AL2/5.10 x86_64	AL2023/6.1/ aarch64	AL2023/6.1/ x86_64	AL2023/6.12/ aarch64	AL2023/6.12/ x86_64
CONFIG_SFS	n	m	n	m	n	n	n	n
CONFIG_RRPC	n	m	n	m	n	n	n	n
CONFIG_DISKDE	y	y	y	y	n	n	n	n
CONFIG_AMFS	m	m	m	m	n	n	n	n
CONFIG_AMFS_BLOCKDEV	N/A	N/A	y	n	N/A	N/A	N/A	N/A
CONFIG_CLONE	N/A	N/A	n	n	n	n	n	n
CONFIG_ERA	m	n	m	n	n	n	n	n

CONFIG option	AL2/4.14 aarch64	AL2/4.14 x86_64	AL2/5.10 aarch64	AL2/5.10 x86_64	AL2023/6.1/ aarch64	AL2023/6.1/ x86_64	AL2023/6.12/ aarch64	AL2023/6.12/ x86_64
CONFIG_INTEGRITY	n	m	n	m	m	m	m	m
CONFIG_LOG_WRITES	n	n	m	m	m	m	m	m
CONFIG_SWITCH	m	n	m	n	n	n	n	n
CONFIG_VERIFY	m	n	m	n	m	m	m	m
CONFIG_RYPT_FS	n	m	n	m	n	n	n	n
CONFIG_FAT_FS	N/A	N/A	m	m	m	m	m	m
CONFIG_T2_FS	n	m	n	m	n	n	n	n
CONFIG_T3_FS	n	m	n	m	n	n	n	n
CONFIG_S2_FS	m	m	m	m	n	n	n	n
CONFIG_SPLUS_FS	n	m	n	m	n	n	n	n
CONFIG_S_FS	n	m	n	m	n	n	n	n

CONFIG option	AL2/4.14 aarch64	AL2/4.14 x86_64	AL2/5.10 aarch64	AL2/5.10 x86_64	AL2023/6.1/ aarch64	AL2023/6.1/ x86_64	AL2023/6.12/ aarch64	AL2023/6.12/ x86_64
CONFIG_SFS	n	n	n	n	n	n	n	n
CONFIG_M_PARTITION	n	y	n	y	n	n	n	n
CONFIG_C_PARTITION	n	y	n	y	n	n	n	n
CONFIG_S_V2	n	m	n	m	n	n	n	n
CONFIG_FS_FS	n	m	n	n	n	n	n	n
CONFIG_MFS_FS	n	m	n	m	n	n	n	n
CONFIG_XLARISS_XLARISS_PARTITION	n	y	n	y	n	n	n	n
CONFIG_UASHFS_TDT	n	y	n	y	y	y	y	y
CONFIG_N_PARTITION	n	y	n	y	n	n	n	n

Andrew File System support (AFS)

The kernel is no longer built with support for the `afs` file system. AL2 did not ship with user-space support for `afs`.

cramfs support

The kernel is no longer built with support for the `cramfs` file system. The successor in AL2023 is the `squashfs` file system.

BSD disklabel support

The kernel is no longer built with support for BSD disk labels. If reading volumes with BSD disk labels is required, various BSDs can be launched.

Device Mapper changes

There have been several changes to the Device Mapper targets configured in the AL2023 kernel.

eCryptFs support

The `ecryptfs` file system has been deprecated in Amazon Linux. The user-space components of `ecryptfs` were present in AL1, removed in AL2, and AL2023 no longer builds the kernel with `ecryptfs` support.

exFAT

Support for the `exFAT` file system was added in the 5.10 kernel in AL2. It was not present at AL2 launch with a 4.14 kernel. AL2023 continues to support the `exFAT` file system.

The ext2, ext3, and ext4 file systems

AL2023 ships with the `CONFIG_EXT4_USE_FOR_EXT2` option, which means that the `ext4` file system code will be used to read legacy `ext2` file systems.

CONFIG_GFS2_FS

The kernel is no longer built with `CONFIG_GFS2_FS`.

Apple Extended HFS file system support (HFS+)

In AL2, only the x86-64 kernels were built with the `hfsplus` file system support. The AL2 5.15 kernel does not include `hfsplus` support on any architecture. In AL2023, we complete the deprecation of `hfsplus` support in Amazon Linux.

HFS file system support

In AL2, only the x86-64 kernels were built with the `hfs` file system support. The AL2 5.15 kernel does not include `hfs` support on any architecture. In AL2023, we complete the deprecation of `hfs` support in Amazon Linux.

JFS file system support

Older AL2 x86-64 kernels were built with `jfs` file system support. The AL2 5.15 kernel does not include `jfs` support on any architecture. Neither AL1 or AL2 shipped with JFS userspace. In AL2023, we complete the deprecation of `jfs` support in Amazon Linux.

The upstream Linux kernel is [considering the removal of JFS](#). Therefore, if you have data on a JFS file system, you should migrate it to another file system. In 2024, JFS was removed from all current Amazon Linux kernels.

Windows Logical Disk Manager (Dynamic Disk) support (CONFIG_LDM_PARTITION)

AL2023 no longer supports Windows 2000, Windows XP, or Windows Vista *dynamic disks* with MS-DOS style partitions. This code did not ever support the newer GPT based dynamic disks introduced with Windows Vista.

Macintosh partition map support

AL2023 no longer supports the classic Macintosh partition map. Modern macOS versions will create modern GPT partition tables by default over this older type.

NFSv2 support

AL2023 no longer supports NFSv2, but continues to support NFSv3, NFSv4, NFSv4.1, and NFSv4.2. We recommend that you migrate to NFSv3 or newer.

NTFS (CONFIG_NTFS_FS)

The `ntfs3` code replaced `ntfs` for accessing NTFS file systems on Amazon Linux as of the 5.10 kernel in AL2. AL2023 no longer includes the `ntfs` code, and relies exclusively on the `ntfs3` code for accessing NTFS file systems.

romfs file system

The `squashfs` file system is the successor of the `romfs` file system in Amazon Linux, and the AL2023 kernel is no longer built with support for `romfs`.

Solaris x86 hard disk partition format

AL2023 no longer supports the Solaris x86 hard disk partition format.

squashfszstd compression

AL2023 adds support for `zstd` compressed `squashfs` file systems on all supported architectures.

Sun partition table support

AL2023 no longer includes support for the Sun partition table format (`CONFIG_SUN_PARTITION`).

/tmp is now tmpfs

Amazon Linux 2023 introduces changes to how `/tmp` behaves when compared to Amazon Linux 2. The default configuration for AL2 was that both `/tmp` and `/var/tmp` were on the root file system. Amazon Linux 2023 defaults to using `tmpfs` for `/tmp` with a limit of 50% of RAM and a maximum of one million inodes. These changes bring Amazon Linux in line with the behavior of other Linux distributions.

For full details of the file system layout of AL2023, see [/tmp](#) and [/var/tmp](#) in the [Filesystem Layout](#) section.

AMI and Container Image changes

There have been some changes to the packages included in AMIs and containers.

Amazon Linux 2023 introduces a [the section called “AL2023 Minimal container image”](#), and support for building [the section called “Building bare-bones AL2023 container images”](#). For more information, see [Using AL2023 in containers](#).

Comparing packages installed on Amazon Linux 2 and Amazon Linux 2023 AMIs

A comparison of the RPMs present on the Amazon Linux 2 and AL2023 standard AMIs.

Package	AL2 AMI	AL2023 AMI
acl	2.2.51	2.3.1
acpid	2.0.19	2.0.32
alternatives		1.15
amazon-chrony-config		4.3
amazon-ec2-net-utils		2.5.1
amazon-linux-extras	2.0.3	
amazon-linux-extras-yum-plugin	2.0.3	
amazon-linux-repo-s3		2023.6.20241031
amazon-linux-sb-keys		2023.1
amazon-rpm-config		228
amazon-ssm-agent	3.3.987.0	3.3.987.0
amd-ucode-firmware	20200421 (noarch)	20210208 (noarch)
at	3.1.13	3.1.23
attr	2.4.46	2.5.1

Package	AL2 AMI	AL2023 AMI
audit	2.8.1	3.0.6
audit-libs	2.8.1	3.0.6
authconfig	6.2.8	
aws-cfn-bootstrap	2.0	2.0
awscli	1.18.147	
awscli-2		2.15.30
basesystem	10.0	11
bash	4.2.46	5.2.15
bash-completion	2.1	2.11
bc	1.06.95	1.07.1
bind-export-libs	9.11.4	
bind-libs	9.11.4	9.18.28
bind-libs-lite	9.11.4	
bind-license	9.11.4	9.18.28
bind-utils	9.11.4	9.18.28
binutils	2.29.1	2.39
blktrace	1.0.5	
boost-date-time	1.53.0 (x86_64)	
boost-filesystem		1.75.0
boost-system	1.53.0 (x86_64)	1.75.0

Package	AL2 AMI	AL2023 AMI
boost-thread	1.53.0 (x86_64)	1.75.0
bridge-utils	1.5	
bzip2	1.0.6	1.0.8
bzip2-libs	1.0.6	1.0.8
ca-certificates	2023.2.68	2023.2.68
c-ares		1.19.1
checkpolicy		3.4
chkconfig	1.7.4	1.15
chrony	4.2	4.3
cloud-init	19.3	22.2.2
cloud-init-cfg-ec2		22.2.2
cloud-utils-growpart	0.31	0.31
coreutils	8.22	8.32
coreutils-common		8.32
cpio	2.12	2.13
cracklib	2.9.0	2.9.6
cracklib-dicts	2.9.0	2.9.6
cronie	1.4.11	
cronie-anacron	1.4.11	
crontabs	1.11	1.11

Package	AL2 AMI	AL2023 AMI
crypto-policies		20220428
crypto-policies-scripts		20220428
cryptsetup	1.7.4	2.6.1
cryptsetup-libs	1.7.4	2.6.1
curl	8.3.0	
curl-minimal		8.5.0
cyrus-sasl-lib	2.1.26	2.1.27
cyrus-sasl-plain	2.1.26	2.1.27
dbus	1.10.24	1.12.28
dbus-broker		32
dbus-common		1.12.28
dbus-libs	1.10.24	1.12.28
device-mapper	1.02.170	1.02.185
device-mapper-event	1.02.170	
device-mapper-event-libs	1.02.170	
device-mapper-libs	1.02.170	1.02.185
device-mapper-persistent-data	0.7.3	
dhclient	4.2.5	

Package	AL2 AMI	AL2023 AMI
dhcp-common	4.2.5	
dhcp-libs	4.2.5	
diffutils	3.3	3.8
dmidecode	3.2	
dmraid	1.0.0.rc16	
dmraid-events	1.0.0.rc16	
dnf		4.14.0
dnf-data		4.14.0
dnf-plugin-release-notification		1.2
dnf-plugins-core		4.3.0
dnf-plugin-support-info		1.2
dnf-utils		4.3.0
dosfstools	3.0.20	4.2
dracut	033	055
dracut-config-ec2	2.0	3.0
dracut-config-generic	033	055
dwz		0.14
dyninst	9.3.1 (x86_64)	10.2.1

Package	AL2 AMI	AL2023 AMI
e2fsprogs	1.42.9	1.46.5
e2fsprogs-libs	1.42.9	1.46.5
ec2-hibinit-agent	1.0.8	1.0.8
ec2-instance-connect	1.1	1.1
ec2-instance-connect-selinux	1.1	1.1
ec2-net-utils	1.7.3	
ec2-utils	1.2	2.2.0
ed	1.9	1.14.2
efibootmgr	15 (aarch64)	
efi-filesystem		5
efi-srpm-macros		5
efivar		38
efivar-libs	31 (aarch64)	38
elfutils-debuginfod-client		0.188
elfutils-default-yama-scope	0.176	0.188
elfutils-libelf	0.176	0.188
elfutils-libs	0.176	0.188
ethtool	4.8	5.15

Package	AL2 AMI	AL2023 AMI
expat	2.1.0	2.5.0
file	5.11	5.39
file-libs	5.11	5.39
filesystem	3.2	3.14
findutils	4.5.11	4.8.0
fipscheck	1.4.1	
fipscheck-lib	1.4.1	
fonts-srpm-macros		2.0.5
freetype	2.8	
fstrm		0.6.1
fuse-libs	2.9.2	2.9.9
gawk	4.0.2	5.1.0
gdbm	1.13	
gdbm-libs		1.19
gdisk	0.8.10	1.0.8
generic-logos	18.0.0	
GeoIP	1.5.0	
gettext	0.19.8.1	0.21
gettext-libs	0.19.8.1	0.21
ghc-srpm-macros		1.5.0

Package	AL2 AMI	AL2023 AMI
glib2	2.56.1	2.74.7
glibc	2.26	2.34
glibc-all-langpacks	2.26	2.34
glibc-common	2.26	2.34
glibc-gconv-extra		2.34
glibc-locale-source	2.26	2.34
glibc-minimal-lang pack	2.26	
gmp	6.0.0	6.2.1
gnupg2	2.0.22	
gnupg2-minimal		2.3.7
gnutls		3.8.0
go-srpm-macros		3.2.0
gpgme	1.3.2	1.15.1
gpm-libs	1.20.7	1.20.7
grep	2.20	3.8
groff-base	1.22.2	1.22.4
grub2	2.06	
grub2-common	2.06	2.06
grub2-efi-aa64	2.06 (aarch64)	
grub2-efi-aa64-ec2	2.06 (aarch64)	2.06 (aarch64)

Package	AL2 AMI	AL2023 AMI
grub2-efi-aa64-modules	2.06 (noarch)	
grub2-efi-x64-ec2	2.06 (x86_64)	2.06 (x86_64)
grub2-pc	2.06 (x86_64)	
grub2-pc-modules	2.06 (noarch)	2.06
grub2-tools	2.06	2.06
grub2-tools-minimal	2.06	2.06
grubby	8.28	8.40
gssproxy	0.7.0	0.8.4
gzip	1.5	1.12
hardlink	1.3	
hibagent	1.1.0	
hostname	3.13	3.23
hunspell	1.3.2	1.7.0
hunspell-en	0.20121024	0.20140811.1
hunspell-en-GB	0.20121024	0.20140811.1
hunspell-en-US	0.20121024	0.20140811.1
hunspell-filesystem		1.7.0
hwdata	0.252	0.384
info	5.1	6.7
inih		49

Package	AL2 AMI	AL2023 AMI
initscripts	9.49.47	10.09
iproute	5.10.0	6.10.0
iptables	1.8.4	
iptables-lib	1.8.4	
iputils	20180629	20210202
irqbalance	1.7.0	1.9.0
jansson	2.10	2.14
jbigkit-lib	2.0	
jemalloc		5.2.1
jitterentropy		3.4.1
jq		1.7.1
json-c	0.11	0.14
kbd	1.15.5	2.4.0
kbd-legacy	1.15.5	
kbd-misc	1.15.5	2.4.0
kernel	5.10.228	6.1.112
kernel-libbpf		6.1.112
kernel-livepatch-r epo-s3		2023.6.20241031
kernel-srpm-macros		1.0
kernel-tools	5.10.228	6.1.112

Package	AL2 AMI	AL2023 AMI
keyutils	1.5.8	1.6.3
keyutils-libs	1.5.8	1.6.3
kmod	25	29
kmod-libs	25	29
kpartx	0.4.9	
kpatch-runtime	0.9.4	0.9.7
krb5-libs	1.15.1	1.21.3
langtable	0.0.31	
langtable-data	0.0.31	
langtable-python	0.0.31	
less	458	608
libacl	2.2.51	2.3.1
libaio	0.3.109	0.3.111
libarchive		3.7.4
libargon2		20171227
libassuan	2.1.0	2.5.5
libattr	2.4.46	2.5.1
libbasicobjects	0.1.1	0.1.1
libblkid	2.30.2	2.37.4
libcap	2.54	2.48

Package	AL2 AMI	AL2023 AMI
libcap-ng	0.7.5	0.8.2
libcbor		0.7.0
libcollection	0.7.0	0.7.0
libcom_err	1.42.9	1.46.5
libcomps		0.1.20
libconfig	1.4.9	1.7.2
libcroco	0.6.12	
libcrypt	2.26	
libcurl	8.3.0	
libcurl-minimal		8.5.0
libdaemon	0.14	
libdb	5.3.21	5.3.28
libdb-utils	5.3.21	
libdhash		0.5.0
libdnf		0.69.0
libdrm	2.4.97	
libdwarf	20130207 (x86_64)	
libeconf		0.4.0
libedit	3.0	3.1
libestr	0.1.9	

Package	AL2 AMI	AL2023 AMI
libev		4.33
libevent	2.0.21	2.1.12
libfastjson	0.99.4	
libfdisk	2.30.2	2.37.4
libffi	3.0.13	3.4.4
libfido2		1.10.0
libgcc	7.3.1	11.4.1
libgcrypt	1.5.3	1.10.2
libgomp	7.3.1	11.4.1
libgpg-error	1.12	1.42
libibverbs		48.0
libicu	50.2	
libidn	1.28	
libidn2	2.3.0	2.3.2
libini_config	1.3.1	1.3.1
libjpeg-turbo	2.0.90	
libkcapi		1.4.0
libkcapi-hmacalc		1.4.0
libldb		2.6.2
libmaxminddb		1.5.2

Package	AL2 AMI	AL2023 AMI
libmetalink	0.1.3	0.1.3
libmnl	1.0.3	1.0.4
libmodulemd		2.13.0
libmount	2.30.2	2.37.4
libnetfilter_connt rack	1.0.6	
libnfnetlink	1.0.1	
libnfsidmap	0.25	2.5.4
libnghttp2	1.41.0	1.59.0
libnl3	3.2.28	3.5.0
libnl3-cli	3.2.28	
libpath_utils	0.2.1	0.2.1
libpcap	1.5.3	1.10.1
libpciaccess	0.14 (x86_64)	
libpipeline	1.2.3	1.5.3
libpkgconf		1.8.0
libpng	1.5.13	
libpsl	0.21.5	0.21.1
libpwquality	1.2.3	1.4.4
libref_array	0.1.5	0.1.5
librepo		1.14.5

Package	AL2 AMI	AL2023 AMI
libreport-filessystem		2.15.2
libseccomp	2.5.2	2.5.3
libselinux	2.5	3.4
libselinux-utils	2.5	3.4
libsemanage	2.5	3.4
libsepol	2.5	3.4
libsigsegv		2.13
libsmartcols	2.30.2	2.37.4
libsolv		0.7.22
libss	1.42.9	1.46.5
libssh2	1.4.3	
libsss_certmap		2.9.4
libsss_idmap	1.16.5	2.9.4
libsss_nss_idmap	1.16.5	2.9.4
libsss_sudo		2.9.4
libstdc++	7.3.1	11.4.1
libstoragemgmt	1.6.1	1.9.4
libstoragemgmt-python	1.6.1	
libstoragemgmt-python-clibs	1.6.1	

Package	AL2 AMI	AL2023 AMI
libsysfs	2.1.0	
libtalloc		2.3.4
libtasn1	4.10	4.19.0
libtdb		1.4.7
libteam	1.27	
libtevent		0.13.0
libtextstyle		0.21
libtiff	4.0.3	
libtirpc	0.2.4	1.3.3
libunistring	0.9.3	0.9.10
libuser	0.60	0.63
libutempter	1.1.6	1.2.1
libuuid	2.30.2	2.37.4
libuv		1.47.0
libverto	0.2.5	0.3.2
libverto-libev		0.3.2
libverto-libevent	0.2.5	
libwebp	0.3.0	
libxcrypt		4.4.33
libxml2	2.9.1	2.10.4

Package	AL2 AMI	AL2023 AMI
libxml2-python	2.9.1	
libyaml	0.1.4	0.2.5
libzstd		1.5.5
linux-firmware-whe nce		20210208 (noarch)
lm_sensors-libs	3.4.0	3.6.0
lmdb-libs		0.9.29
logrotate	3.8.6	3.20.1
lsof	4.87	4.94.0
lua	5.1.4	
lua-libs		5.4.4
lua-srpm-macros		1
lvm2	2.02.187	
lvm2-libs	2.02.187	
lz4	1.7.5	
lz4-libs		1.9.4
make	3.82	
man-db	2.6.3	2.9.3
man-pages	3.53	5.10
man-pages-overrides	7.5.2	
mariadb-libs	5.5.68	

Package	AL2 AMI	AL2023 AMI
mdadm	4.0	
microcode_ctl	2.1 (x86_64)	2.1 (x86_64)
mlocate	0.26	
mpfr		4.1.0
mtr	0.92	
nano	2.9.8	5.8
ncurses	6.0	6.2
ncurses-base	6.0	6.2
ncurses-libs	6.0	6.2
nettle	2.7.1	3.8
net-tools	2.0	2.0
newt	0.52.15	0.52.21
newt-python	0.52.15	
nfs-utils	1.3.0	2.5.4
npth		1.6
nspr	4.35.0	4.35.0
nss	3.90.0	3.90.0
nss-pem	1.0.3	
nss-softokn	3.90.0	3.90.0
nss-softokn-freebl	3.90.0	3.90.0

Package	AL2 AMI	AL2023 AMI
nss-sysinit	3.90.0	3.90.0
nss-tools	3.90.0	
nss-util	3.90.0	3.90.0
ntsysv	1.7.4	1.15
numactl-libs	2.0.9	2.0.14
ocaml-srpm-macros		6
oniguruma		6.9.7.1
openblas-srpm-macros		2
openldap	2.4.44	2.4.57
openssh	7.4p1	8.7p1
openssh-clients	7.4p1	8.7p1
openssh-server	7.4p1	8.7p1
openssl	1.0.2k	3.0.8
openssl-libs	1.0.2k	3.0.8
openssl-pkcs11		0.4.12
os-prober	1.58	1.77
p11-kit	0.23.22	0.24.1
p11-kit-trust	0.23.22	0.24.1
package-notes-srpm-macros		0.4
pam	1.1.8	1.5.1

Package	AL2 AMI	AL2023 AMI
parted	3.1	3.4
passwd	0.79	0.80
pciutils	3.5.1	3.7.0
pciutils-libs	3.5.1	3.7.0
pcre	8.32	
pcre2	10.23	10.40
pcre2-syntax		10.40
perl	5.16.3	
perl-Carp	1.26	1.50
perl-Class-Struct		0.66
perl-constant	1.27	1.33
perl-DynaLoader		1.47
perl-Encode	2.51	3.15
perl-Errno		1.30
perl-Exporter	5.68	5.74
perl-Fcntl		1.13
perl-File-Basename		2.85
perl-File-Path	2.09	2.18
perl-File-stat		1.09
perl-File-Temp	0.23.01	0.231.100

Package	AL2 AMI	AL2023 AMI
perl-Filter	1.49	
perl-Getopt-Long	2.40	2.52
perl-Getopt-Std		1.12
perl-HTTP-Tiny	0.033	0.078
perl-if		0.60.800
perl-interpreter		5.32.1
perl-IO		1.43
perl-IPC-Open3		1.21
perl-libs	5.16.3	5.32.1
perl-macros	5.16.3	
perl-MIME-Base64		3.16
perl-mro		1.23
perl-overload		1.31
perl-overloading		0.02
perl-parent	0.225	0.238
perl-PathTools	3.40	3.78
perl-Pod-Escapes	1.04	1.07
perl-podlators	2.5.1	4.14
perl-Pod-Perldoc	3.20	3.28.01
perl-Pod-Simple	3.28	3.42

Package	AL2 AMI	AL2023 AMI
perl-Pod-Usage	1.63	2.01
perl-POSIX		1.94
perl-Scalar-List-Utils	1.27	1.56
perl-SelectSaver		1.02
perl-Socket	2.010	2.032
perl-srpm-macros		1
perl-Storable	2.45	3.21
perl-subs		1.03
perl-Symbol		1.08
perl-Term-ANSIColor		5.01
perl-Term-Cap		1.17
perl-Text-ParseWords	3.29	3.30
perl-Text-Tabs+Wrap		2021.0726
perl-threads	1.87	
perl-threads-shared	1.43	
perl-Time-HiRes	1.9725	
perl-Time-Local	1.2300	1.300
perl-vars		1.05
pinentry	0.8.1	
pkgconf		1.8.0

Package	AL2 AMI	AL2023 AMI
pkgconfig	0.27.1	
pkgconf-m4		1.8.0
pkgconf-pkg-config		1.8.0
plymouth	0.8.9	
plymouth-core-libs	0.8.9	
plymouth-scripts	0.8.9	
pm-utils	1.4.1	
polycoreutils	2.5	3.4
polycoreutils-python-utils		3.4
popt	1.13	1.18
postfix	2.10.1	
procps-ng	3.3.10	3.3.17
protobuf-c		1.4.1
psacct	6.6.1	6.6.4
psmisc	22.20	23.4
pth	2.0.7	
publicsuffix-list-dafsa	20240208	20240212
pygpgme	0.3	
pyliblzma	0.5.3	

Package	AL2 AMI	AL2023 AMI
pystache	0.5.3	
python	2.7.18	
python2-botocore	1.18.6	
python2-colorama	0.3.9	
python2-cryptography	1.7.2	
python2-dateutil	2.6.1	
python2-futures	3.0.5	
python2-jmespath	0.9.3	
python2-jsonschema	2.5.1	
python2-oauthlib	2.0.1	
python2-pyasn1	0.1.9	
python2-rpm	4.11.3	
python2-rsa	3.4.1	
python2-s3transfer	0.3.3	
python2-setuptools	41.2.0	
python2-six	1.11.0	
python3	3.7.16	3.9.16
python3-attrs		20.3.0
python3-audit		3.0.6
python3-awscrt		0.19.19

Package	AL2 AMI	AL2023 AMI
python3-babel		2.9.1
python3-cffi		1.14.5
python3-chardet		4.0.0
python3-colorama		0.4.4
python3-configobj		5.0.6
python3-cryptography		36.0.1
python3-daemon	2.2.3	2.3.0
python3-dateutil		2.8.1
python3-dbus		1.2.18
python3-distro		1.5.0
python3-dnf		4.14.0
python3-dnf-plugins-core		4.3.0
python3-docutils	0.14	0.16
python3-gpg		1.15.1
python3-hawkey		0.69.0
python3-idna		2.10
python3-jinja2		2.11.3
python3-jmespath		0.10.0
python3-jsonpatch		1.21
python3-jsonpointer		2.0

Package	AL2 AMI	AL2023 AMI
python3-jsonschema		3.2.0
python3-libcomps		0.1.20
python3-libdnf		0.69.0
python3-libs	3.7.16	3.9.16
python3-libselinux		3.4
python3-libsemanage		3.4
python3-libstorage mgmt		1.9.4
python3-lockfile	0.11.0	0.12.2
python3-markupsafe		1.1.1
python3-netifaces		0.10.6
python3-oauthlib		3.0.2
python3-pip	20.2.2	
python3-pip-wheel		21.3.1
python3-ply		3.11
python3-policycore utils		3.4
python3-prettytable		0.7.2
python3-prompt-too lkit		3.0.24
python3-pycparser		2.20

Package	AL2 AMI	AL2023 AMI
python3-pyrsistent		0.17.3
python3-pyserial		3.4
python3-pysocks		1.7.1
python3-pystache	0.5.4	
python3-pytz		2022.7.1
python3-pyyaml		5.4.1
python3-requests		2.25.1
python3-rpm		4.16.1.3
python3-ruamel-yaml		0.16.6
python3-ruamel-yaml-clib		0.1.2
python3-setools		4.4.1
python3-setuptools	49.1.3	59.6.0
python3-setuptools-wheel		59.6.0
python3-simplejson	3.2.0	
python3-six		1.15.0
python3-systemd		235
python3-urllib3		1.25.10
python3-wcwidth		0.2.5
python-babel	0.9.6	

Package	AL2 AMI	AL2023 AMI
python-backports	1.0	
python-backports-s sl_match_hostname	3.5.0.1	
python-cffi	1.6.0	
python-chardet	2.2.1	
python-chevron		0.13.1
python-configobj	4.7.2	
python-daemon	1.6	
python-devel	2.7.18	
python-docutils	0.12	
python-enum34	1.0.4	
python-idna	2.4	
python-iniparse	0.4	
python-ipaddress	1.0.16	
python-jinja2	2.7.2	
python-jsonpatch	1.2	
python-jsonpointer	1.9	
python-jwcrypto	0.4.2	
python-kitchen	1.1.1	
python-libs	2.7.18	
python-lockfile	0.9.1	

Package	AL2 AMI	AL2023 AMI
python-markupsafe	0.11	
python-pillow	2.0.0	
python-ply	3.4	
python-pycparser	2.14	
python-pycurl	7.19.0	
python-repoze-lru	0.4	
python-requests	2.6.0	
python-simplejson	3.2.0	
python-srpm-macros		3.9
python-urlgrabber	3.10	
python-urllib3	1.25.9	
pyxattr	0.5.1	
PyYAML	3.10	
qrencode-libs	3.4.1	
quota	4.01	4.06
quota-nls	4.01	4.06
rdate	1.4	
readline	6.2	8.1
rng-tools	6.8	6.14
rootfiles	8.1	8.1

Package	AL2 AMI	AL2023 AMI
rpcbind	0.2.0	1.2.6
rpm	4.11.3	4.16.1.3
rpm-build-libs	4.11.3	4.16.1.3
rpm-libs	4.11.3	4.16.1.3
rpm-plugin-selinux		4.16.1.3
rpm-plugin-systemd-inhibit	4.11.3	4.16.1.3
rpm-sign-libs		4.16.1.3
rsync	3.1.2	3.2.6
rsyslog	8.24.0	
rust-srpm-macros		21
sbsigntools		0.9.4
scl-utils	20130529	
screen	4.1.0	4.8.0
sed	4.2.2	4.8
selinux-policy	3.13.1	38.1.45
selinux-policy-targeted	3.13.1	38.1.45
setserial	2.17	
setup	2.8.71	2.13.7
setuptools	1.19.11	

Package	AL2 AMI	AL2023 AMI
sgpio	1.2.0.10	
shadow-utils	4.1.5.1	4.9
shared-mime-info	1.8	
slang	2.2.4	2.3.2
sqlite	3.7.17	
sqlite-libs		3.40.0
sssd-client	1.16.5	2.9.4
sssd-common		2.9.4
sssd-kcm		2.9.4
sssd-nfs-idmap		2.9.4
strace	4.26	6.8
sudo	1.8.23	1.9.15
sysctl-defaults	1.0	1.0
sysstat	10.1.5	12.5.6
systemd	219	252.23
systemd-libs	219	252.23
systemd-networkd		252.23
systemd-pam		252.23
systemd-resolved		252.23
systemd-sysv	219	

Package	AL2 AMI	AL2023 AMI
systemd-udev		252.23
system-release	2	2023.6.20241031
systemtap-runtime	4.5	4.8
sysvinit-tools	2.88	
tar	1.26	1.34
tbb		2020.3
tcp_wrappers	7.6	
tcp_wrappers-libs	7.6	
tcpdump	4.9.2	4.99.1
tcsh	6.18.01	6.24.07
teamd	1.27	
time	1.7	1.9
traceroute	2.0.22	2.1.3
tzdata	2024a	2024a
unzip	6.0	6.0
update-motd	1.1.2	2.2
usermode	1.111	
userspace-rcu		0.12.1
ustr	1.0.4	
util-linux	2.30.2	2.37.4

Package	AL2 AMI	AL2023 AMI
util-linux-core		2.37.4
vim-common	9.0.2153	9.0.2153
vim-data	9.0.2153	9.0.2153
vim-enhanced	9.0.2153	9.0.2153
vim-filesystem	9.0.2153	9.0.2153
vim-minimal	9.0.2153	9.0.2153
virt-what	1.18	
wget	1.14	1.21.3
which	2.20	2.21
words	3.0	3.0
xfsdump	3.1.8	3.1.11
xfsprogs	5.0.0	5.18.0
xxd	9.0.2153	9.0.2153
xxhash-libs		0.8.0
xz	5.2.2	5.2.5
xz-libs	5.2.2	5.2.5
yajl	2.0.4	
yum	3.4.3	4.14.0
yum-langpacks	0.4.2	
yum-metadata-parser	1.1.4	

Package	AL2 AMI	AL2023 AMI
yum-plugin-priorities	1.1.31	
yum-utils	1.1.31	
zip	3.0	3.0
zlib	1.2.7	1.2.11
zram-generator		1.1.2
zram-generator-defaults		1.1.2
zstd		1.5.5

Comparing packages installed on Amazon Linux 2 and Amazon Linux 2023 Minimal AMIs

A comparison of the RPMs present on the Amazon Linux 2 and AL2023 Minimal AMIs.

Package	AL2 Minimal	AL2023 Minimal
acl	2.2.51	
alternatives		1.15
amazon-chrond-config		4.3
amazon-ec2-net-utils		2.5.1
amazon-linux-extras	2.0.3	
amazon-linux-repo-s3		2023.6.20241031
amazon-linux-sb-keys		2023.1

Package	AL2 Minimal	AL2023 Minimal
amd-ucode-firmware	20200421 (noarch)	20210208 (noarch)
audit	2.8.1	3.0.6
audit-libs	2.8.1	3.0.6
authconfig	6.2.8	
awscli-2		2.15.30
basesystem	10.0	11
bash	4.2.46	5.2.15
bind-export-libs	9.11.4	
bzip2-libs	1.0.6	1.0.8
ca-certificates	2023.2.68	2023.2.68
checkpolicy		3.4
chkconfig	1.7.4	
chrony	4.2	4.3
cloud-init	19.3	22.2.2
cloud-init-cfg-ec2		22.2.2
cloud-utils-growpart	0.31	0.31
coreutils	8.22	8.32
coreutils-common		8.32
cpio	2.12	2.13
cracklib	2.9.0	2.9.6

Package	AL2 Minimal	AL2023 Minimal
cracklib-dicts	2.9.0	2.9.6
cronie	1.4.11	
cronie-anacron	1.4.11	
crontabs	1.11	
crypto-policies		20220428
cryptsetup-libs	1.7.4	2.6.1
curl	8.3.0	
curl-minimal		8.5.0
cyrus-sasl-lib	2.1.26	2.1.27
dbus	1.10.24	1.12.28
dbus-broker		32
dbus-common		1.12.28
dbus-libs	1.10.24	1.12.28
device-mapper	1.02.170	1.02.185
device-mapper-libs	1.02.170	1.02.185
dhclient	4.2.5	
dhcp-common	4.2.5	
dhcp-libs	4.2.5	
diffutils	3.3	3.8
dnf		4.14.0

Package	AL2 Minimal	AL2023 Minimal
dnf-data		4.14.0
dnf-plugin-release-notification		1.2
dnf-plugins-core		4.3.0
dnf-plugin-support-info		1.2
dracut	033	055
dracut-config-ec2	2.0	3.0
dracut-config-generic	033	055
e2fsprogs	1.42.9	1.46.5
e2fsprogs-libs	1.42.9	1.46.5
ec2-utils	1.2	2.2.0
efibootmgr	15 (aarch64)	
efi-filesystem		5
efivar		38
efivar-libs	31 (aarch64)	38
elfutils-default-yama-scope	0.176	0.188
elfutils-libelf	0.176	0.188
elfutils-libs	0.176	0.188
expat	2.1.0	2.5.0

Package	AL2 Minimal	AL2023 Minimal
file	5.11	5.39
file-libs	5.11	5.39
filesystem	3.2	3.14
findutils	4.5.11	4.8.0
fipscheck	1.4.1	
fipscheck-lib	1.4.1	
freetype	2.8	
fuse-libs	2.9.2	2.9.9
gawk	4.0.2	5.1.0
gdbm	1.13	
gdbm-libs		1.19
gdisk	0.8.10	1.0.8
gettext	0.19.8.1	0.21
gettext-libs	0.19.8.1	0.21
glib2	2.56.1	2.74.7
glibc	2.26	2.34
glibc-all-langpacks	2.26	2.34
glibc-common	2.26	2.34
glibc-locale-source	2.26	2.34
glibc-minimal-lang pack	2.26	

Package	AL2 Minimal	AL2023 Minimal
gmp	6.0.0	6.2.1
gnupg2	2.0.22	
gnupg2-minimal		2.3.7
gnutls		3.8.0
gpgme	1.3.2	1.15.1
grep	2.20	3.8
groff-base	1.22.2	1.22.4
grub2	2.06	
grub2-common	2.06	2.06
grub2-efi-aa64	2.06 (aarch64)	
grub2-efi-aa64-ec2	2.06 (aarch64)	2.06 (aarch64)
grub2-efi-aa64-modules	2.06 (noarch)	
grub2-efi-x64-ec2	2.06 (x86_64)	2.06 (x86_64)
grub2-pc	2.06 (x86_64)	
grub2-pc-modules	2.06 (noarch)	2.06
grub2-tools	2.06	2.06
grub2-tools-minimal	2.06	2.06
grubby	8.28	8.40
gzip	1.5	1.12
hardlink	1.3	

Package	AL2 Minimal	AL2023 Minimal
hostname	3.13	3.23
hwdata		0.384
info	5.1	
inih		49
initscripts	9.49.47	10.09
iproute	5.10.0	6.10.0
iptables	1.8.4	
iptables-libs	1.8.4	
iputils	20180629	20210202
irqbalance	1.7.0	1.9.0
jansson		2.14
jitterentropy		3.4.1
jq		1.7.1
json-c		0.14
kbd		2.4.0
kbd-misc		2.4.0
kernel	4.14.355	6.1.112
kernel-libbpf		6.1.112
kernel-livepatch-r epo-s3		2023.6.20241031
keyutils-libs	1.5.8	1.6.3

Package	AL2 Minimal	AL2023 Minimal
kmod	25	29
kmod-libs	25	29
kpartx	0.4.9	
krb5-libs	1.15.1	1.21.3
less	458	608
libacl	2.2.51	2.3.1
libarchive		3.7.4
libargon2		20171227
libassuan	2.1.0	2.5.5
libattr	2.4.46	2.5.1
libblkid	2.30.2	2.37.4
libcap	2.54	2.48
libcap-ng	0.7.5	0.8.2
libcbor		0.7.0
libcom_err	1.42.9	1.46.5
libcomps		0.1.20
libcroco	0.6.12	
libcrypt	2.26	
libcurl	8.3.0	
libcurl-minimal		8.5.0

Package	AL2 Minimal	AL2023 Minimal
libdb	5.3.21	5.3.28
libdb-utils	5.3.21	
libdnf		0.69.0
libeconf		0.4.0
libedit	3.0	3.1
libestr	0.1.9	
libfastjson	0.99.4	
libfdisk	2.30.2	2.37.4
libffi	3.0.13	3.4.4
libfido2		1.10.0
libgcc	7.3.1	11.4.1
libgcrypt	1.5.3	1.10.2
libgomp	7.3.1	11.4.1
libgpg-error	1.12	1.42
libicu	50.2	
libidn	1.28	
libidn2	2.3.0	2.3.2
libkcapi		1.4.0
libkcapi-hmacalc		1.4.0
libmetalink	0.1.3	

Package	AL2 Minimal	AL2023 Minimal
libmnl	1.0.3	1.0.4
libmodulemd		2.13.0
libmount	2.30.2	2.37.4
libnetfilter_conntrack	1.0.6	
libnfnetlink	1.0.1	
libnhttp2	1.41.0	1.59.0
libpcap	1.5.3	
libpipeline	1.2.3	1.5.3
libpng	1.5.13	
libpsl	0.21.5	0.21.1
libpwquality	1.2.3	1.4.4
librepo		1.14.5
libreport-filesystem		2.15.2
libseccomp	2.5.2	2.5.3
libselinux	2.5	3.4
libselinux-utils	2.5	3.4
libsemanage	2.5	3.4
libsepol	2.5	3.4
libsigsegv		2.13
libsmartcols	2.30.2	2.37.4

Package	AL2 Minimal	AL2023 Minimal
libsolv		0.7.22
libss	1.42.9	1.46.5
libssh2	1.4.3	
libstdc++	7.3.1	11.4.1
libsysfs	2.1.0	
libtasn1	4.10	4.19.0
libtextstyle		0.21
libunistring	0.9.3	0.9.10
libuser	0.60	0.63
libutempter	1.1.6	1.2.1
libuuid	2.30.2	2.37.4
libverto	0.2.5	0.3.2
libxcrypt		4.4.33
libxml2	2.9.1	2.10.4
libyaml	0.1.4	0.2.5
libzstd		1.5.5
linux-firmware-whe nce		20210208 (noarch)
logrotate	3.8.6	3.20.1
lua	5.1.4	
lua-libs		5.4.4

Package	AL2 Minimal	AL2023 Minimal
lz4	1.7.5	
lz4-libs		1.9.4
make	3.82	
man-db	2.6.3	2.9.3
mariadb-libs	5.5.68	
microcode_ctl	2.1 (x86_64)	2.1 (x86_64)
mpfr		4.1.0
ncurses	6.0	6.2
ncurses-base	6.0	6.2
ncurses-libs	6.0	6.2
nettle	2.7.1	3.8
net-tools	2.0	2.0
newt	0.52.15	
newt-python	0.52.15	
npth		1.6
nspr	4.35.0	
nss	3.90.0	
nss-pem	1.0.3	
nss-softokn	3.90.0	
nss-softokn-freebl	3.90.0	

Package	AL2 Minimal	AL2023 Minimal
nss-sysinit	3.90.0	
nss-tools	3.90.0	
nss-util	3.90.0	
numactl-libs	2.0.9	2.0.14
oniguruma		6.9.7.1
openldap	2.4.44	2.4.57
openssh	7.4p1	8.7p1
openssh-clients	7.4p1	8.7p1
openssh-server	7.4p1	8.7p1
openssl	1.0.2k	3.0.8
openssl-libs	1.0.2k	3.0.8
openssl-pkcs11		0.4.12
os-prober	1.58	1.77
p11-kit	0.23.22	0.24.1
p11-kit-trust	0.23.22	0.24.1
pam	1.1.8	1.5.1
passwd	0.79	0.80
pciutils		3.7.0
pciutils-libs		3.7.0
pcre	8.32	

Package	AL2 Minimal	AL2023 Minimal
pcr2	10.23	10.40
pcr2-syntax		10.40
pinentry	0.8.1	
pkgconfig	0.27.1	
policycoreutils	2.5	3.4
popt	1.13	1.18
postfix	2.10.1	
procps-ng	3.3.10	3.3.17
psmisc	22.20	23.4
pth	2.0.7	
publicsuffix-list-dafsa	20240208	20240212
pygpgme	0.3	
pyliblzma	0.5.3	
python	2.7.18	
python2-cryptography	1.7.2	
python2-jsonschema	2.5.1	
python2-oauthlib	2.0.1	
python2-pyasn1	0.1.9	
python2-rpm	4.11.3	
python2-setuptools	41.2.0	

Package	AL2 Minimal	AL2023 Minimal
python2-six	1.11.0	
python3		3.9.16
python3-attrs		20.3.0
python3-audit		3.0.6
python3-awscrt		0.19.19
python3-babel		2.9.1
python3-cffi		1.14.5
python3-chardet		4.0.0
python3-colorama		0.4.4
python3-configobj		5.0.6
python3-cryptography		36.0.1
python3-dateutil		2.8.1
python3-dbus		1.2.18
python3-distro		1.5.0
python3-dnf		4.14.0
python3-dnf-plugins-core		4.3.0
python3-docutils		0.16
python3-gpg		1.15.1
python3-hawkey		0.69.0
python3-idna		2.10

Package	AL2 Minimal	AL2023 Minimal
python3-jinja2		2.11.3
python3-jmespath		0.10.0
python3-jsonpatch		1.21
python3-jsonpointer		2.0
python3-jsonschema		3.2.0
python3-libcomps		0.1.20
python3-libdnf		0.69.0
python3-libs		3.9.16
python3-libselenium		3.4
python3-libsemanage		3.4
python3-markupsafe		1.1.1
python3-netifaces		0.10.6
python3-oauthlib		3.0.2
python3-pip-wheel		21.3.1
python3-ply		3.11
python3-policycore utils		3.4
python3-prettytable		0.7.2
python3-prompt-too lkit		3.0.24
python3-pycparser		2.20

Package	AL2 Minimal	AL2023 Minimal
python3-pyrsistent		0.17.3
python3-pyserial		3.4
python3-pysocks		1.7.1
python3-pytz		2022.7.1
python3-pyyaml		5.4.1
python3-requests		2.25.1
python3-rpm		4.16.1.3
python3-ruamel-yaml		0.16.6
python3-ruamel-yaml-clib		0.1.2
python3-setools		4.4.1
python3-setuptools		59.6.0
python3-setuptools-wheel		59.6.0
python3-six		1.15.0
python3-systemd		235
python3-urllib3		1.25.10
python3-wcwidth		0.2.5
python-babel	0.9.6	
python-backports	1.0	

Package	AL2 Minimal	AL2023 Minimal
python-backports-s sl_match_hostname	3.5.0.1	
python-cffi	1.6.0	
python-chardet	2.2.1	
python-configobj	4.7.2	
python-devel	2.7.18	
python-enum34	1.0.4	
python-idna	2.4	
python-iniparse	0.4	
python-ipaddress	1.0.16	
python-jinja2	2.7.2	
python-jsonpatch	1.2	
python-jsonpointer	1.9	
python-jwcrypto	0.4.2	
python-libs	2.7.18	
python-markupsafe	0.11	
python-ply	3.4	
python-pycparser	2.14	
python-pycurl	7.19.0	
python-repoze-lru	0.4	
python-requests	2.6.0	

Package	AL2 Minimal	AL2023 Minimal
python-urlgrabber	3.10	
python-urllib3	1.25.9	
pyxattr	0.5.1	
PyYAML	3.10	
qrencode-libs	3.4.1	
readline	6.2	8.1
rng-tools	6.8	6.14
rootfiles	8.1	8.1
rpm	4.11.3	4.16.1.3
rpm-build-libs	4.11.3	4.16.1.3
rpm-libs	4.11.3	4.16.1.3
rpm-plugin-selinux		4.16.1.3
rpm-plugin-systemd-inhibit	4.11.3	4.16.1.3
rpm-sign-libs		4.16.1.3
rsyslog	8.24.0	
sbsigntools		0.9.4
sed	4.2.2	4.8
selinux-policy	3.13.1	38.1.45
selinux-policy-targeted	3.13.1	38.1.45

Package	AL2 Minimal	AL2023 Minimal
setup	2.8.71	2.13.7
shadow-utils	4.1.5.1	4.9
shared-mime-info	1.8	
slang	2.2.4	
sqlite	3.7.17	
sqlite-libs		3.40.0
sudo	1.8.23	1.9.15
sysctl-defaults	1.0	1.0
systemd	219	252.23
systemd-libs	219	252.23
systemd-networkd		252.23
systemd-pam		252.23
systemd-resolved		252.23
systemd-sysv	219	
systemd-udev		252.23
system-release	2	2023.6.20241031
sysvinit-tools	2.88	
tar	1.26	1.34
tcp_wrappers-libs	7.6	
tzdata	2024a	2024a

Package	AL2 Minimal	AL2023 Minimal
update-motd	1.1.2	2.2
userspace-rcu		0.12.1
ustr	1.0.4	
util-linux	2.30.2	2.37.4
util-linux-core		2.37.4
vim-data	9.0.2153	9.0.2153
vim-minimal	9.0.2153	9.0.2153
which	2.20	2.21
xfspgrog	5.0.0	5.18.0
xz	5.2.2	5.2.5
xz-libs	5.2.2	5.2.5
yum	3.4.3	4.14.0
yum-metadata-parser	1.1.4	
yum-plugin-priorities	1.1.31	
zlib	1.2.7	1.2.11
zram-generator		1.1.2
zram-generator-defaults		1.1.2
zstd		1.5.5

Comparing packages installed on Amazon Linux 2 and Amazon Linux 2023 base container images

A comparison of the RPMs present on the Amazon Linux 2 and AL2023 base container images.

Package	AL2 Container	AL2023 Container
alternatives		1.15
amazon-linux-extras	2.0.3	
amazon-linux-repo-cdn		2023.6.20241031
audit-libs		3.0.6
basesystem	10.0	11
bash	4.2.46	5.2.15
bzip2-libs	1.0.6	1.0.8
ca-certificates	2023.2.68	2023.2.68
chkconfig	1.7.4	
coreutils	8.22	
coreutils-single		8.32
cpio	2.12	
crypto-policies		20220428
curl	8.3.0	
curl-minimal		8.5.0
cyrus-sasl-lib	2.1.26	

Package	AL2 Container	AL2023 Container
diffutils	3.3	
dnf		4.14.0
dnf-data		4.14.0
elfutils-default-yama-scope		0.188
elfutils-libelf	0.176	0.188
elfutils-libs		0.188
expat	2.1.0	2.5.0
file-libs	5.11	5.39
filesystem	3.2	3.14
findutils	4.5.11	
gawk	4.0.2	5.1.0
gdbm	1.13	
gdbm-libs		1.19
glib2	2.56.1	2.74.7
glibc	2.26	2.34
glibc-common	2.26	2.34
glibc-langpack-en	2.26	
glibc-minimal-langpack	2.26	2.34
gmp	6.0.0	6.2.1

Package	AL2 Container	AL2023 Container
gnupg2	2.0.22	
gnupg2-minimal		2.3.7
gpgme	1.3.2	1.15.1
grep	2.20	3.8
info	5.1	
json-c		0.14
keyutils-libs	1.5.8	1.6.3
krb5-libs	1.15.1	1.21.3
libacl	2.2.51	2.3.1
libarchive		3.7.4
libassuan	2.1.0	2.5.5
libattr	2.4.46	2.5.1
libblkid	2.30.2	2.37.4
libcap	2.54	2.48
libcap-ng		0.8.2
libcom_err	1.42.9	1.46.5
libcomps		0.1.20
libcrypt	2.26	
libcurl	8.3.0	
libcurl-minimal		8.5.0

Package	AL2 Container	AL2023 Container
libdb	5.3.21	
libdb-utils	5.3.21	
libdnf		0.69.0
libffi	3.0.13	3.4.4
libgcc	7.3.1	11.4.1
libgcrypt	1.5.3	1.10.2
libgomp		11.4.1
libgpg-error	1.12	1.42
libidn2	2.3.0	2.3.2
libmetalink	0.1.3	
libmodulemd		2.13.0
libmount	2.30.2	2.37.4
libnghttp2	1.41.0	1.59.0
libpsl	0.21.5	0.21.1
librepo		1.14.5
libreport-filessystem		2.15.2
libselinux	2.5	3.4
libsepol	2.5	3.4
libsigsegv		2.13
libsmartcols		2.37.4

Package	AL2 Container	AL2023 Container
libsolv		0.7.22
libssh2	1.4.3	
libstdc++	7.3.1	11.4.1
libtasn1	4.10	4.19.0
libunistring	0.9.3	0.9.10
libuuid	2.30.2	2.37.4
libverto	0.2.5	0.3.2
libxcrypt		4.4.33
libxml2	2.9.1	2.10.4
libyaml		0.2.5
libzstd		1.5.5
lua	5.1.4	
lua-libs		5.4.4
lz4-libs		1.9.4
mpfr		4.1.0
ncurses	6.0	
ncurses-base	6.0	6.2
ncurses-libs	6.0	6.2
npth		1.6
nspr	4.35.0	

Package	AL2 Container	AL2023 Container
nss	3.90.0	
nss-pem	1.0.3	
nss-softokn	3.90.0	
nss-softokn-freebl	3.90.0	
nss-sysinit	3.90.0	
nss-tools	3.90.0	
nss-util	3.90.0	
openldap	2.4.44	
openssl-lib	1.0.2k	3.0.8
p11-kit	0.23.22	0.24.1
p11-kit-trust	0.23.22	0.24.1
pcre	8.32	
pcre2		10.40
pcre2-syntax		10.40
pinentry	0.8.1	
popt	1.13	1.18
pth	2.0.7	
publicsuffix-list-dafsa	20240208	20240212
pygpgme	0.3	
pyliblzma	0.5.3	

Package	AL2 Container	AL2023 Container
python	2.7.18	
python2-rpm	4.11.3	
python3		3.9.16
python3-dnf		4.14.0
python3-gpg		1.15.1
python3-hawkey		0.69.0
python3-libcomps		0.1.20
python3-libdnf		0.69.0
python3-libs		3.9.16
python3-pip-wheel		21.3.1
python3-rpm		4.16.1.3
python3-setuptools-wheel		59.6.0
python-iniparse	0.4	
python-libs	2.7.18	
python-pycurl	7.19.0	
python-urlgrabber	3.10	
pyxattr	0.5.1	
readline	6.2	8.1
rpm	4.11.3	4.16.1.3
rpm-build-libs	4.11.3	4.16.1.3

Package	AL2 Container	AL2023 Container
rpm-libs	4.11.3	4.16.1.3
rpm-sign-libs		4.16.1.3
sed	4.2.2	4.8
setup	2.8.71	2.13.7
shared-mime-info	1.8	
sqlite	3.7.17	
sqlite-libs		3.40.0
system-release	2	2023.6.20241031
tzdata	2024a	2024a
vim-data	9.0.2153	
vim-minimal	9.0.2153	
xz-libs	5.2.2	5.2.5
yum	3.4.3	4.14.0
yum-metadata-parser	1.1.4	
yum-plugin-ovl	1.1.31	
yum-plugin-priorities	1.1.31	
zlib	1.2.7	1.2.11

Comparing AL1 and AL2023

The following topics describe key differences between AL1 and AL2023 that aren't already covered by the [comparison with AL2](#).

Note

AL1 reached its end-of-life (EOL) on December 31, 2023 and will not receive any security updates or bug fixes starting January 1, 2024. For more information about AL1 EOL and maintenance support, see the blog post [Update on Amazon Linux AMI end-of-life](#). We recommend that you upgrade applications to AL2023, which includes long-term support until 2028.

Topics

- [Support for each release](#)
- [systemd replaces upstart as init system](#)
- [Python 2.6 and 2.7 has been replaced with Python 3](#)
- [OpenJDK 8 as oldest JDK](#)
- [AL2023 kernel changes from Amazon Linux 1 \(AL1\)](#)
- [Comparing packages installed on Amazon Linux 1 \(AL1\) and Amazon Linux 2023 AMIs](#)
- [Comparing packages installed on Amazon Linux 1 \(AL1\) and Amazon Linux 2023 Minimal AMIs](#)
- [Comparing packages installed on Amazon Linux 1 \(AL1\) and Amazon Linux 2023 base container images](#)

Support for each release

For AL2023, we offer five years of support from the release date. AL1 ended standard support as of December 31, 2020 and ended maintenance support as of December 31, 2023.

For more information, see [Release cadence](#).

systemd replaces upstart as init system

In AL2 upstart was replaced by systemd as the init system. AL2023 also uses systemd as its init system, further adopting new features and functionality of systemd.

Python 2.6 and 2.7 has been replaced with Python 3

Although AL1 marked Python 2.6 as EOL with the 2018.03 release, the packages were still available in the repositories to install. AL2 shipped with Python 2.7 as the earliest supported Python version, and AL2023 completes the transition to Python 3. No Python 2.x versions are included in the AL2023 repositories.

For more information on Python on Amazon Linux, see [Python in AL2023](#).

OpenJDK 8 as oldest JDK

AL2023 ships with [Amazon Corretto](#) as the default (and only) Java Development Kit (JDK). All Java based packages in AL2023 are built with Amazon Corretto 17.

In AL1, OpenJDK 1.6.0 (java-1.6.0-openjdk) went EOL with the first 2018.03 release, and OpenJDK 1.7.0 (java-1.7.0-openjdk) went EOL in mid-2020, although both versions were available in the AL1 repositories. The earliest OpenJDK version available in AL2023 is OpenJDK 8, provided by Amazon Corretto 8.

AL2023 kernel changes from Amazon Linux 1 (AL1)

Kernel Live Patching

Both AL2023 and AL2 add support for kernel live-patching functionality. This allows you to patch critical and important security vulnerabilities in the Linux kernel without reboot or downtime. For more information, see [Kernel Live Patching on AL2023](#).

Kernel file system support

There have been several changes in the file systems that the kernel in AL1 will support mounting, along with changes in the partitioning schemes that the kernel will parse.

CONFIG option	AL1/4.14/x86_64	AL2023/6.1/aarch64	AL2023/6.1/x86_64	AL2023/6.12/aarch64	AL2023/6.12/x86_64
<u>CONFIG_AFS_FS</u>	m	n	n	n	n
<u>CONFIG_AFS_RRPC</u>	m	n	n	n	n
<u>CONFIG_BSD_DISKLABEL</u>	y	n	n	n	n
<u>CONFIG_CRAMFS</u>	m	n	n	n	n
<u>CONFIG_CRAMFS_BLOCKDEV</u>	N/A	N/A	N/A	N/A	N/A
<u>CONFIG_DM_CLONE</u>	N/A	n	n	n	n
<u>CONFIG_DM_ERA</u>	n	n	n	n	n
<u>CONFIG_DM_INTEGRITY</u>	m	m	m	m	m
<u>CONFIG_DM_LOG_WRITES</u>	n	m	m	m	m
<u>CONFIG_DM_SWITCH</u>	n	n	n	n	n
<u>CONFIG_DM_VERITY</u>	n	m	m	m	m

CONFIG option	AL1/4.14/x86_64	AL2023/6.1/aarch64	AL2023/6.1/x86_64	AL2023/6.12/aarch64	AL2023/6.12/x86_64
<u>CONFIG_ECRYPT_FS</u>	m	n	n	n	n
<u>CONFIG_EXFAT_FS</u>	N/A	m	m	m	m
<u>CONFIG_EXT2_FS</u>	m	n	n	n	n
<u>CONFIG_EXT3_FS</u>	m	n	n	n	n
<u>CONFIG_GFS2_FS</u>	n	n	n	n	n
<u>CONFIG_HFSPLUS_FS</u>	m	n	n	n	n
<u>CONFIG_HFS_FS</u>	m	n	n	n	n
<u>CONFIG_JFS_FS</u>	n	n	n	n	n
<u>CONFIG_LDM_PARTITION</u>	y	n	n	n	n
<u>CONFIG_MACE_PARTITION</u>	y	n	n	n	n
<u>CONFIG_NFS_V2S_V2</u>	m	n	n	n	n

CONFIG option	AL1/4.14/x86_64	AL2023/6.1/aarch64	AL2023/6.1/x86_64	AL2023/6.12/aarch64	AL2023/6.12/x86_64
CONFIG_NTFS_FS	m	n	n	n	n
CONFIG_ROMFS_FS	m	n	n	n	n
CONFIG_SOLARIS_X86_PARTITION	y	n	n	n	n
CONFIG_SQUASHFS_ZSTD	y	y	y	y	y
CONFIG_SUN_PARTITION	y	n	n	n	n

Security focused kernel config changes

CONFIG option	AL1/4.14/x86_64	AL2023/6.1/aarch64	AL2023/6.1/x86_64	AL2023/6.12/aarch64	AL2023/6.12/x86_64
CONFIG_BUG_ON_DATA_CORRUPTION	y	y	y	y	y
CONFIG_DEFAULT_MMAP_MIN_ADDR	4096	65536	65536	65536	65536

CONFIG option	AL1/4.14/x86_64	AL2023/6.1/aarch64	AL2023/6.1/x86_64	AL2023/6.12/aarch64	AL2023/6.12/x86_64
<u>CONFIG_DEVMEM</u>	y	n	n	n	n
<u>CONFIG_DEVPORT</u>	y	n	n	n	n
<u>CONFIG_FORTIFY_SOURCE</u>	y	y	y	y	y
<u>CONFIG_HARDENED_USERCOPY_FALLBACK</u>	N/A	N/A	N/A	N/A	N/A
<u>CONFIG_INIT_ON_ALLOC_DEFAULT_ON</u>	N/A	n	n	n	n
<u>CONFIG_INIT_ON_FREE_DEFAULT_ON</u>	N/A	n	n	n	n
<u>CONFIG_IOMMU_DEFAULT_DMA_STRICT</u>	N/A	n	n	n	n
<u>CONFIG_LDISC_AUTOLOAD</u>	y	n	n	n	n

CONFIG option	AL1/4.14/x86_64	AL2023/6.1/aarch64	AL2023/6.1/x86_64	AL2023/6.12/aarch64	AL2023/6.12/x86_64
CONFIG_SCH_HED_CORE	N/A	N/A	y	N/A	y
CONFIG_SCH_HED_STACK_END_CHECK	y	y	y	y	y
CONFIG_SECURITY_DMESG_RESTRICT	n	y	y	y	y
CONFIG_SECURITY_SELINUX_DISABLE	y	n	n	N/A	N/A
CONFIG_SHUFFLE_PAGE_ALLOCATOR	N/A	y	y	y	y
CONFIG_SLAB_FREELIST_HARDENED	y	y	y	y	y
CONFIG_SLAB_FREELIST_RANDOM	n	y	y	y	y

Other kernel configuration changes

CONFIG option	AL1/4.14/x86_64	AL2023/6.1/aarch64	AL2023/6.1/x86_64	AL2023/6.12/aarch64	AL2023/6.12/x86_64
CONFIG_HZ	250	100	100	100	100
CONFIG_NR_CPUS	8192	4096	8192	4096	8192
CONFIG_PANIC_ON_OOPS	n	y	y	y	y
CONFIG_PANIC_ON_OOPS_VALUE	0	1	1	1	1
CONFIG_PREEMPT	m	n	n	n	n
CONFIG_SMP	m	n	n	n	n
CONFIG_XEN_PV	y	N/A	n	N/A	n

Comparing packages installed on Amazon Linux 1 (AL1) and Amazon Linux 2023 AMIs

A comparison of the RPMs present on the AL1 and AL2023 standard AMIs.

Package	AL1 AMI	AL2023 AMI
acl	2.2.49	2.3.1
acpid	2.0.19	2.0.32

Package	AL1 AMI	AL2023 AMI
alsa-lib	1.0.22	
alternatives		1.15
amazon-chrony-config		4.3
amazon-ec2-net-utils		2.5.1
amazon-linux-repo-s3		2023.6.20241031
amazon-linux-sb-keys		2023.1
amazon-rpm-config		228
amazon-ssm-agent	3.2.2222.0	3.3.987.0
amd-ucode-firmware		20210208
at	3.1.10	3.1.23
attr	2.4.46	2.5.1
audit	2.6.5	3.0.6
audit-libs	2.6.5	3.0.6
authconfig	6.2.8	
aws-amitools-ec2	1.5.13	
aws-cfn-bootstrap	1.4	2.0
aws-cli	1.18.107	
awscli-2		2.15.30
basesystem	10.0	11
bash	4.2.46	5.2.15

Package	AL1 AMI	AL2023 AMI
bash-completion		2.11
bc	1.06.95	1.07.1
bind-libs	9.8.2	9.18.28
bind-license		9.18.28
bind-utils	9.8.2	9.18.28
binutils	2.27	2.39
boost-filesystem		1.75.0
boost-system		1.75.0
boost-thread		1.75.0
bzip2	1.0.6	1.0.8
bzip2-libs	1.0.6	1.0.8
ca-certificates	2023.2.62	2023.2.68
c-ares		1.19.1
checkpolicy	2.1.10	3.4
chkconfig	1.3.49.3	1.15
chrony		4.3
cloud-disk-utils	0.27	
cloud-init	0.7.6	22.2.2
cloud-init-cfg-ec2		22.2.2
cloud-utils-growpart		0.31

Package	AL1 AMI	AL2023 AMI
copy-jdk-configs	3.3	
coreutils	8.22	8.32
coreutils-common		8.32
cpio	2.10	2.13
cracklib	2.8.16	2.9.6
cracklib-dicts	2.8.16	2.9.6
cronie	1.4.4	
cronie-anacron	1.4.4	
crontabs	1.10	1.11
crypto-policies		20220428
crypto-policies-scripts		20220428
cryptsetup	1.6.7	2.6.1
cryptsetup-libs	1.6.7	2.6.1
curl	7.61.1	
curl-minimal		8.5.0
cyrus-sasl	2.1.23	
cyrus-sasl-lib	2.1.23	2.1.27
cyrus-sasl-plain	2.1.23	2.1.27
dash	0.5.5.1	
db4	4.7.25	

Package	AL1 AMI	AL2023 AMI
db4-utils	4.7.25	
dbus	1.6.12	1.12.28
dbus-broker		32
dbus-common		1.12.28
dbus-libs	1.6.12	1.12.28
dejavu-fonts-common	2.33	
dejavu-sans-fonts	2.33	
dejavu-serif-fonts	2.33	
device-mapper	1.02.135	1.02.185
device-mapper-event	1.02.135	
device-mapper-event-libs	1.02.135	
device-mapper-libs	1.02.135	1.02.185
device-mapper-persistent-data	0.6.3	
dhclient	4.1.1	
dhcp-common	4.1.1	
diffutils	3.3	3.8
dmraid	1.0.0.rc16	
dmraid-events	1.0.0.rc16	
dnf		4.14.0

Package	AL1 AMI	AL2023 AMI
dnf-data		4.14.0
dnf-plugin-release-notification		1.2
dnf-plugins-core		4.3.0
dnf-plugin-support-info		1.2
dnf-utils		4.3.0
dosfstools		4.2
dracut	004	055
dracut-config-ec2		3.0
dracut-config-generic		055
dracut-modules-growroot	0.20	
dump	0.4	
dwz		0.14
dyninst		10.2.1
e2fsprogs	1.43.5	1.46.5
e2fsprogs-libs	1.43.5	1.46.5
ec2-hibinit-agent	1.0.0	1.0.8
ec2-instance-connect		1.1

Package	AL1 AMI	AL2023 AMI
ec2-instance-connect-selinux		1.1
ec2-net-utils	0.7	
ec2-utils	0.7	2.2.0
ed	1.1	1.14.2
efi-filesystem		5
efi-srpm-macros		5
efivar		38
efivar-libs		38
elfutils-debuginfod-client		0.188
elfutils-default-yama-scope		0.188
elfutils-libelf	0.168	0.188
elfutils-libs		0.188
epel-release	6	
ethtool	3.15	5.15
expat	2.1.0	2.5.0
file	5.37	5.39
file-libs	5.37	5.39
filesystem	2.4.30	3.14

Package	AL1 AMI	AL2023 AMI
findutils	4.4.2	4.8.0
fipscheck	1.3.1	
fipscheck-lib	1.3.1	
fontconfig	2.8.0	
fontpackages-files system	1.41	
fonts-srpm-macros		2.0.5
freetype	2.3.11	
fstrm		0.6.1
fuse-libs	2.9.4	2.9.9
gawk	3.1.7	5.1.0
gdbm	1.8.0	
gdbm-libs		1.19
gdisk	0.8.10	1.0.8
generic-logos	17.0.0	
get_reference_source	1.2	
gettext		0.21
gettext-libs		0.21
ghc-srpm-macros		1.5.0
giflib	4.1.6	
glib2	2.36.3	2.74.7

Package	AL1 AMI	AL2023 AMI
glibc	2.17	2.34
glibc-all-langpacks		2.34
glibc-common	2.17	2.34
glibc-gconv-extra		2.34
glibc-locale-source		2.34
gmp	6.0.0	6.2.1
gnupg2	2.0.28	
gnupg2-minimal		2.3.7
gnutls		3.8.0
go-srpm-macros		3.2.0
gpgme	1.4.3	1.15.1
gpm-libs	1.20.6	1.20.7
grep	2.20	3.8
groff	1.22.2	
groff-base	1.22.2	1.22.4
grub	0.97	
grub2-common		2.06
grub2-efi-x64-ec2		2.06
grub2-pc-modules		2.06
grub2-tools		2.06

Package	AL1 AMI	AL2023 AMI
grub2-tools-minimal		2.06
grubby	7.0.15	8.40
gssproxy		0.8.4
gzip	1.5	1.12
hesiod	3.1.0	
hibagent	1.0.0	
hmaccalc	0.9.12	
hostname		3.23
hunspell		1.7.0
hunspell-en		0.20140811.1
hunspell-en-GB		0.20140811.1
hunspell-en-US		0.20140811.1
hunspell-filesystem		1.7.0
hwdata	0.233	0.384
info	5.1	6.7
inih		49
initscripts	9.03.58	10.09
iproute	4.4.0	6.10.0
iptables	1.4.21	
iputils	20121221	20210202

Package	AL1 AMI	AL2023 AMI
irqbalance	1.5.0	1.9.0
jansson		2.14
java-1.7.0-openjdk	1.7.0.321	
javapackages-tools	0.9.1	
jemalloc		5.2.1
jitterentropy		3.4.1
jpackage-utils	1.7.5	
jq		1.7.1
json-c		0.14
kbd	1.15	2.4.0
kbd-misc	1.15	2.4.0
kernel	4.14.336	6.1.112
kernel-libbpf		6.1.112
kernel-livepatch-r epo-s3		2023.6.20241031
kernel-srpm-macros		1.0
kernel-tools	4.14.336	6.1.112
keyutils	1.5.8	1.6.3
keyutils-libs	1.5.8	1.6.3
kmod	14	29
kmod-libs	14	29

Package	AL1 AMI	AL2023 AMI
kpartx	0.4.9	
kpatch-runtime		0.9.7
krb5-libs	1.15.1	1.21.3
lcms2	2.6	
less	436	608
libacl	2.2.49	2.3.1
libaio	0.3.109	0.3.111
libarchive		3.7.4
libargon2		20171227
libassuan	2.0.3	2.5.5
libattr	2.4.46	2.5.1
libbasicobjects		0.1.1
libblkid	2.23.2	2.37.4
libcap	2.16	2.48
libcap54	2.54	
libcap-ng	0.7.5	0.8.2
libcbor		0.7.0
libcgroupp	0.40.rc1	
libcollection		0.7.0
libcom_err	1.43.5	1.46.5

Package	AL1 AMI	AL2023 AMI
libcomps		0.1.20
libconfig		1.7.2
libcurl	7.61.1	
libcurl-minimal		8.5.0
libdb		5.3.28
libdhash		0.5.0
libdnf		0.69.0
libeconf		0.4.0
libedit	2.11	3.1
libev		4.33
libevent	2.0.21	2.1.12
libfdisk		2.37.4
libffi	3.0.13	3.4.4
libfido2		1.10.0
libfontenc	1.0.5	
libgcc		11.4.1
libgcc72	7.2.1	
libgcrypt	1.5.3	1.10.2
libgomp		11.4.1
libgpg-error	1.11	1.42

Package	AL1 AMI	AL2023 AMI
libgssglue	0.1	
libibverbs		48.0
libICE	1.0.6	
libicu	50.2	
libidn	1.18	
libidn2	2.3.0	2.3.2
libini_config		1.3.1
libjpeg-turbo	1.2.90	
libkcapi		1.4.0
libkcapi-hmaccalc		1.4.0
libldb		2.6.2
libmaxminddb		1.5.2
libmetalink		0.1.3
libmnl	1.0.3	1.0.4
libmodulemd		2.13.0
libmount	2.23.2	2.37.4
libnetfilter_connt rack	1.0.4	
libnfnetlink	1.0.1	
libnfsidmap	0.25	2.5.4
libnhttp2	1.33.0	1.59.0

Package	AL1 AMI	AL2023 AMI
libnih	1.0.1	
libnl	1.1.4	
libnl3		3.5.0
libpath_utils		0.2.1
libpcap		1.10.1
libpipeline	1.2.3	1.5.3
libpkgconf		1.8.0
libpng	1.2.49	
libpsl	0.6.2	0.21.1
libpwquality	1.2.3	1.4.4
libref_array		0.1.5
librepo		1.14.5
libreport-filesystem		2.15.2
libseccomp		2.5.3
libselinux	2.1.10	3.4
libselinux-utils	2.1.10	3.4
libsemanage	2.1.6	3.4
libsepol	2.1.7	3.4
libsigsegv		2.13
libSM	1.2.1	

Package	AL1 AMI	AL2023 AMI
libsmartcols	2.23.2	2.37.4
libsolv		0.7.22
libss	1.43.5	1.46.5
libssh2	1.4.2	
libsss_certmap		2.9.4
libsss_idmap		2.9.4
libsss_nss_idmap		2.9.4
libsss_sudo		2.9.4
libstdc++		11.4.1
libstdc++72	7.2.1	
libstoragemgmt		1.9.4
libsysfs	2.1.0	
libtalloc		2.3.4
libtasn1	2.3	4.19.0
libtdb		1.4.7
libtevent		0.13.0
libtextstyle		0.21
libtirpc	0.2.4	1.3.3
libudev	173	
libunistring	0.9.3	0.9.10

Package	AL1 AMI	AL2023 AMI
libuser	0.60	0.63
libutempter	1.1.5	1.2.1
libuuid	2.23.2	2.37.4
libuv		1.47.0
libverto	0.2.5	0.3.2
libverto-libev		0.3.2
libX11	1.6.0	
libX11-common	1.6.0	
libXau	1.0.6	
libxcb	1.11	
libXcomposite	0.4.3	
libxcrypt		4.4.33
libXext	1.3.2	
libXfont	1.4.5	
libXi	1.7.2	
libxml2	2.9.1	2.10.4
libxml2-python27	2.9.1	
libXrender	0.9.8	
libxslt	1.1.28	
libXtst	1.2.2	

Package	AL1 AMI	AL2023 AMI
libyaml	0.1.6	0.2.5
libzstd		1.5.5
linux-firmware-wheel		20210208
lm_sensors-libs		3.6.0
lmdb-libs		0.9.29
log4j-cve-2021-44228-hotpatch	1.3	
logrotate	3.7.8	3.20.1
lsof	4.82	4.94.0
lua	5.1.4	
lua-libs		5.4.4
lua-srpm-macros		1
lvm2	2.02.166	
lvm2-libs	2.02.166	
lz4-libs		1.9.4
mailcap	2.1.31	
make	3.82	
man-db	2.6.3	2.9.3
man-pages	4.10	5.10
mdadm	3.2.6	

Package	AL1 AMI	AL2023 AMI
microcode_ctl	2.1	2.1
mingetty	1.08	
mpfr		4.1.0
nano	2.5.3	5.8
nc	1.84	
ncurses	5.7	6.2
ncurses-base	5.7	6.2
ncurses-libs	5.7	6.2
nettle		3.8
net-tools	1.60	2.0
newt	0.52.11	0.52.21
newt-python27	0.52.11	
nfs-utils	1.3.0	2.5.4
npth		1.6
nspr	4.25.0	4.35.0
nss	3.53.1	3.90.0
nss-pem	1.0.3	
nss-softokn	3.53.1	3.90.0
nss-softokn-freebl	3.53.1	3.90.0
nss-sysinit	3.53.1	3.90.0

Package	AL1 AMI	AL2023 AMI
nss-tools	3.53.1	
nss-util	3.53.1	3.90.0
ntp	4.2.8p15	
ntpd	4.2.8p15	
ntsysv	1.3.49.3	1.15
numactl	2.0.7	
numactl-libs		2.0.14
ocaml-srpm-macros		6
oniguruma		6.9.7.1
openblas-srpm-macros		2
openldap	2.4.40	2.4.57
openssh	7.4p1	8.7p1
openssh-clients	7.4p1	8.7p1
openssh-server	7.4p1	8.7p1
openssl	1.0.2k	3.0.8
openssl-libs		3.0.8
openssl-pkcs11		0.4.12
os-prober		1.77
p11-kit	0.18.5	0.24.1
p11-kit-trust	0.18.5	0.24.1

Package	AL1 AMI	AL2023 AMI
package-notes-srpm-macros		0.4
pam	1.1.8	1.5.1
pam_ccreds	10	
pam_krb5	2.3.11	
pam_passwdqc	1.0.5	
parted	2.1	3.4
passwd	0.79	0.80
pciutils	3.1.10	3.7.0
pciutils-libs	3.1.10	3.7.0
pcre	8.21	
pcre2		10.40
pcre2-syntax		10.40
perl	5.16.3	
perl-Carp	1.26	1.50
perl-Class-Struct		0.66
perl-constant	1.27	1.33
perl-Digest	1.17	
perl-Digest-HMAC	1.03	
perl-Digest-MD5	2.52	
perl-Digest-SHA	5.85	

Package	AL1 AMI	AL2023 AMI
perl-DynaLoader		1.47
perl-Encode	2.51	3.15
perl-Errno		1.30
perl-Exporter	5.68	5.74
perl-Fcntl		1.13
perl-File-Basename		2.85
perl-File-Path	2.09	2.18
perl-File-stat		1.09
perl-File-Temp	0.23.01	0.231.100
perl-Filter	1.49	
perl-Getopt-Long	2.40	2.52
perl-Getopt-Std		1.12
perl-HTTP-Tiny	0.033	0.078
perl-if		0.60.800
perl-interpreter		5.32.1
perl-IO		1.43
perl-IPC-Open3		1.21
perl-libs	5.16.3	5.32.1
perl-macros	5.16.3	
perl-MIME-Base64		3.16

Package	AL1 AMI	AL2023 AMI
perl-mro		1.23
perl-overload		1.31
perl-overloading		0.02
perl-parent	0.225	0.238
perl-PathTools	3.40	3.78
perl-Pod-Escapes	1.04	1.07
perl-podlators	2.5.1	4.14
perl-Pod-Perldoc	3.20	3.28.01
perl-Pod-Simple	3.28	3.42
perl-Pod-Usage	1.63	2.01
perl-POSIX		1.94
perl-Scalar-List-Utils	1.27	1.56
perl-SelectSaver		1.02
perl-Socket	2.010	2.032
perl-srpm-macros		1
perl-Storable	2.45	3.21
perl-subs		1.03
perl-Symbol		1.08
perl-Term-ANSIColor		5.01
perl-Term-Cap		1.17

Package	AL1 AMI	AL2023 AMI
perl-Text-ParseWords	3.29	3.30
perl-Text-Tabs+Wrap		2021.0726
perl-threads	1.87	
perl-threads-shared	1.43	
perl-Time-HiRes	1.9725	
perl-Time-Local	1.2300	1.300
perl-vars		1.05
pinentry	0.7.6	
pkgconf		1.8.0
pkgconfig	0.27.1	
pkgconf-m4		1.8.0
pkgconf-pkg-config		1.8.0
pm-utils	1.4.1	
policycoreutils	2.1.12	3.4
policycoreutils-python-utils		3.4
popt	1.13	1.18
procmail	3.22	
procps	3.2.8	
procps-ng		3.3.17
protobuf-c		1.4.1

Package	AL1 AMI	AL2023 AMI
psacct	6.3.2	6.6.4
psmisc	22.20	23.4
pth	2.0.7	
publicsuffix-list-dafsa		20240212
python27	2.7.18	
python27-babel	0.9.4	
python27-backports	1.0	
python27-backports-ssl_match_hostname	3.4.0.2	
python27-boto	2.48.0	
python27-botocore	1.17.31	
python27-chardet	2.0.1	
python27-colorama	0.4.1	
python27-configobj	4.7.2	
python27-crypto	2.6.1	
python27-daemon	1.5.2	
python27-dateutil	2.1	
python27-devel	2.7.18	
python27-docutils	0.11	
python27-ecdsa	0.11	

Package	AL1 AMI	AL2023 AMI
python27-futures	3.0.3	
python27-imaging	1.1.6	
python27-iniparse	0.3.1	
python27-jinja2	2.7.2	
python27-jmespath	0.9.2	
python27-jsonpatch	1.2	
python27-jsonpointer	1.0	
python27-kitchen	1.1.1	
python27-libs	2.7.18	
python27-lockfile	0.8	
python27-markupsafe	0.11	
python27-paramiko	1.15.1	
python27-pip	9.0.3	
python27-ply	3.4	
python27-pyasn1	0.1.7	
python27-pycurl	7.19.0	
python27-pygpme	0.3	
python27-pyliblzma	0.5.3	
python27-pystache	0.5.3	
python27-pyxdattr	0.5.0	

Package	AL1 AMI	AL2023 AMI
python27-PyYAML	3.10	
python27-requests	1.2.3	
python27-rsa	3.4.1	
python27-setuptools	36.2.7	
python27-simplejson	3.6.5	
python27-six	1.8.0	
python27-urlgrabber	3.10	
python27-urllib3	1.24.3	
python27-virtualenv	15.1.0	
python3		3.9.16
python3-attrs		20.3.0
python3-audit		3.0.6
python3-awscrt		0.19.19
python3-babel		2.9.1
python3-cffi		1.14.5
python3-chardet		4.0.0
python3-colorama		0.4.4
python3-configobj		5.0.6
python3-cryptography		36.0.1
python3-daemon		2.3.0

Package	AL1 AMI	AL2023 AMI
python3-dateutil		2.8.1
python3-dbus		1.2.18
python3-distro		1.5.0
python3-dnf		4.14.0
python3-dnf-plugins-core		4.3.0
python3-docutils		0.16
python3-gpg		1.15.1
python3-hawkey		0.69.0
python3-idna		2.10
python3-jinja2		2.11.3
python3-jmespath		0.10.0
python3-jsonpatch		1.21
python3-jsonpointer		2.0
python3-jjsonschema		3.2.0
python3-libcomps		0.1.20
python3-libdnf		0.69.0
python3-libs		3.9.16
python3-libselenium		3.4
python3-libsemanage		3.4

Package	AL1 AMI	AL2023 AMI
python3-libstorage mgmt		1.9.4
python3-lockfile		0.12.2
python3-markupsafe		1.1.1
python3-netifaces		0.10.6
python3-oauthlib		3.0.2
python3-pip-wheel		21.3.1
python3-ply		3.11
python3-policycore utils		3.4
python3-prettytable		0.7.2
python3-prompt-too lkit		3.0.24
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
python3-requests		2.25.1
python3-rpm		4.16.1.3

Package	AL1 AMI	AL2023 AMI
python3-ruamel-yaml		0.16.6
python3-ruamel-yaml-clib		0.1.2
python3-setuptools		4.4.1
python3-setuptools		59.6.0
python3-setuptools-wheel		59.6.0
python3-six		1.15.0
python3-systemd		235
python3-urllib3		1.25.10
python3-wcwidth		0.2.5
python-chevron		0.13.1
python-srpm-macros		3.9
quota	4.00	4.06
quota-nls	4.00	4.06
readline	6.2	8.1
rmt	0.4	
rng-tools	5	6.14
rootfiles	8.1	8.1
rpcbind	0.2.0	1.2.6
rpm	4.11.3	4.16.1.3

Package	AL1 AMI	AL2023 AMI
rpm-build-libs	4.11.3	4.16.1.3
rpm-libs	4.11.3	4.16.1.3
rpm-plugin-selinux		4.16.1.3
rpm-plugin-systemd-inhibit		4.16.1.3
rpm-python27	4.11.3	
rpm-sign-libs		4.16.1.3
rsync	3.0.6	3.2.6
rsyslog	5.8.10	
ruby	2.0	
ruby20	2.0.0.648	
ruby20-irb	2.0.0.648	
ruby20-libs	2.0.0.648	
rubygem20-bigdecimal	1.2.0	
rubygem20-json	1.8.3	
rubygem20-psych	2.0.0	
rubygem20-rdoc	4.2.2	
rubygems20	2.0.14.1	
rust-srpm-macros		21
sbsigntools		0.9.4
screen	4.0.3	4.8.0

Package	AL1 AMI	AL2023 AMI
sed	4.2.1	4.8
selinux-policy		38.1.45
selinux-policy-targeted		38.1.45
sendmail	8.14.4	
setserial	2.17	
setup	2.8.14	2.13.7
sgpio	1.2.0.10	
shadow-utils	4.1.4.2	4.9
shared-mime-info	1.1	
slang	2.2.1	2.3.2
sqlite	3.7.17	
sqlite-libs		3.40.0
sssd-client		2.9.4
sssd-common		2.9.4
sssd-kcm		2.9.4
sssd-nfs-idmap		2.9.4
strace		6.8
sudo	1.8.23	1.9.15
sysctl-defaults	1.0	1.0
sysfsutils	2.1.0	

Package	AL1 AMI	AL2023 AMI
sysstat		12.5.6
systemd		252.23
systemd-libs		252.23
systemd-networkd		252.23
systemd-pam		252.23
systemd-resolved		252.23
systemd-udev		252.23
system-release	2018.03	2023.6.20241031
systemtap-runtime		4.8
sysvinit	2.87	
tar	1.26	1.34
tbb		2020.3
tcp_wrappers	7.6	
tcp_wrappers-libs	7.6	
tcpdump		4.99.1
tcsh		6.24.07
time	1.7	1.9
tmpwatch	2.9.16	
traceroute	2.0.14	2.1.3
ttmkfdir	3.0.9	

Package	AL1 AMI	AL2023 AMI
tzdata	2023c	2024a
tzdata-java	2023c	
udev	173	
unzip	6.0	6.0
update-motd	1.0.1	2.2
upstart	0.6.5	
userspace-rcu		0.12.1
ustr	1.0.4	
util-linux	2.23.2	2.37.4
util-linux-core		2.37.4
vim-common	9.0.2120	9.0.2153
vim-data	9.0.2120	9.0.2153
vim-enhanced	9.0.2120	9.0.2153
vim-filesystem	9.0.2120	9.0.2153
vim-minimal	9.0.2120	9.0.2153
wget	1.18	1.21.3
which	2.19	2.21
words	3.0	3.0
xfsdump		3.1.11
xfsprogs		5.18.0

Package	AL1 AMI	AL2023 AMI
xorg-x11-fonts-Type1	7.2	
xorg-x11-font-utils	7.2	
xxd	9.0.2120	9.0.2153
xxhash-libs		0.8.0
xz	5.2.2	5.2.5
xz-libs	5.2.2	5.2.5
yum	3.4.3	4.14.0
yum-metadata-parser	1.1.4	
yum-plugin-priorities	1.1.31	
yum-plugin-upgrade-helper	1.1.31	
yum-utils	1.1.31	
zip	3.0	3.0
zlib	1.2.8	1.2.11
zram-generator		1.1.2
zram-generator-defaults		1.1.2
zstd		1.5.5

Comparing packages installed on Amazon Linux 1 (AL1) and Amazon Linux 2023 Minimal AMIs

A comparison of the RPMs present on the AL1 and AL2023 Minimal AMIs.

Package	AL1 Minimal	AL2023 Minimal
acpid	2.0.19	
alternatives		1.15
amazon-chrony-config		4.3
amazon-ec2-net-utils		2.5.1
amazon-linux-repo-s3		2023.6.20241031
amazon-linux-sb-keys		2023.1
amd-ucode-firmware		20210208
audit	2.6.5	3.0.6
audit-libs	2.6.5	3.0.6
authconfig	6.2.8	
awscli-2		2.15.30
basesystem	10.0	11
bash	4.2.46	5.2.15
binutils	2.27	
bzip2	1.0.6	
bzip2-libs	1.0.6	1.0.8
ca-certificates	2023.2.62	2023.2.68

Package	AL1 Minimal	AL2023 Minimal
checkpolicy	2.1.10	3.4
chkconfig	1.3.49.3	
chrony		4.3
cloud-disk-utils	0.27	
cloud-init	0.7.6	22.2.2
cloud-init-cfg-ec2		22.2.2
cloud-utils-growpart		0.31
coreutils	8.22	8.32
coreutils-common		8.32
cpio	2.10	2.13
cracklib	2.8.16	2.9.6
cracklib-dicts	2.8.16	2.9.6
cronie	1.4.4	
cronie-anacron	1.4.4	
crontabs	1.10	
crypto-policies		20220428
cryptsetup-libs		2.6.1
curl	7.61.1	
curl-minimal		8.5.0
cyrus-sasl	2.1.23	

Package	AL1 Minimal	AL2023 Minimal
cyrus-sasl-lib	2.1.23	2.1.27
dash	0.5.5.1	
db4	4.7.25	
db4-utils	4.7.25	
dbus		1.12.28
dbus-broker		32
dbus-common		1.12.28
dbus-libs	1.6.12	1.12.28
device-mapper		1.02.185
device-mapper-libs		1.02.185
dhclient	4.1.1	
dhcp-common	4.1.1	
diffutils	3.3	3.8
dnf		4.14.0
dnf-data		4.14.0
dnf-plugin-release-notification		1.2
dnf-plugins-core		4.3.0
dnf-plugin-support-info		1.2
dracut	004	055

Package	AL1 Minimal	AL2023 Minimal
dracut-config-ec2		3.0
dracut-config-generic		055
dracut-modules-growroot	0.20	
e2fsprogs	1.43.5	1.46.5
e2fsprogs-libs	1.43.5	1.46.5
ec2-utils	0.7	2.2.0
ed	1.1	
efi-filesystem		5
efivar		38
efivar-libs		38
elfutils-default-yama-scope		0.188
elfutils-libelf	0.168	0.188
elfutils-libs		0.188
ethtool	3.15	
expat	2.1.0	2.5.0
file	5.37	5.39
file-libs	5.37	5.39
filesystem	2.4.30	3.14

Package	AL1 Minimal	AL2023 Minimal
findutils	4.4.2	4.8.0
fipscheck	1.3.1	
fipscheck-lib	1.3.1	
fuse-libs	2.9.4	2.9.9
gawk	3.1.7	5.1.0
gdbm	1.8.0	
gdbm-libs		1.19
gdisk	0.8.10	1.0.8
generic-logos	17.0.0	
get_reference_source	1.2	
gettext		0.21
gettext-libs		0.21
glib2	2.36.3	2.74.7
glibc	2.17	2.34
glibc-all-langpacks		2.34
glibc-common	2.17	2.34
glibc-locale-source		2.34
gmp	6.0.0	6.2.1
gnupg2	2.0.28	
gnupg2-minimal		2.3.7

Package	AL1 Minimal	AL2023 Minimal
gnutls		3.8.0
gpgme	1.4.3	1.15.1
grep	2.20	3.8
groff	1.22.2	
groff-base	1.22.2	1.22.4
grub	0.97	
grub2-common		2.06
grub2-efi-x64-ec2		2.06
grub2-pc-modules		2.06
grub2-tools		2.06
grub2-tools-minimal		2.06
grubby	7.0.15	8.40
gzip	1.5	1.12
hesiod	3.1.0	
hmaccalc	0.9.12	
hostname		3.23
hwdata	0.233	0.384
info	5.1	
inih		49
initscripts	9.03.58	10.09

Package	AL1 Minimal	AL2023 Minimal
iproute	4.4.0	6.10.0
iptables	1.4.21	
iputils	20121221	20210202
irqbalance		1.9.0
jansson		2.14
jitterentropy		3.4.1
jq		1.7.1
json-c		0.14
kbd	1.15	2.4.0
kbd-misc	1.15	2.4.0
kernel	4.14.336	6.1.112
kernel-libbpf		6.1.112
kernel-livepatch-r epo-s3		2023.6.20241031
keyutils-libs	1.5.8	1.6.3
kmod	14	29
kmod-libs	14	29
krb5-libs	1.15.1	1.21.3
less	436	608
libacl	2.2.49	2.3.1
libarchive		3.7.4

Package	AL1 Minimal	AL2023 Minimal
libargon2		20171227
libassuan	2.0.3	2.5.5
libattr	2.4.46	2.5.1
libblkid	2.23.2	2.37.4
libcap	2.16	2.48
libcap54	2.54	
libcap-ng	0.7.5	0.8.2
libcbor		0.7.0
libcgroup	0.40.rc1	
libcom_err	1.43.5	1.46.5
libcomps		0.1.20
libcurl	7.61.1	
libcurl-minimal		8.5.0
libdb		5.3.28
libdnf		0.69.0
libeconf		0.4.0
libedit	2.11	3.1
libfdisk		2.37.4
libffi	3.0.13	3.4.4
libfido2		1.10.0

Package	AL1 Minimal	AL2023 Minimal
libgcc		11.4.1
libgcc72	7.2.1	
libgcrypt	1.5.3	1.10.2
libgomp		11.4.1
libgpg-error	1.11	1.42
libicu	50.2	
libidn	1.18	
libidn2	2.3.0	2.3.2
libkcapi		1.4.0
libkcapi-hmaccalc		1.4.0
libmnl	1.0.3	1.0.4
libmodulemd		2.13.0
libmount	2.23.2	2.37.4
libnetfilter_conntrack	1.0.4	
libnfnetlink	1.0.1	
libnghttp2	1.33.0	1.59.0
libnih	1.0.1	
libpipeline		1.5.3
libpsl	0.6.2	0.21.1
libpwquality	1.2.3	1.4.4

Package	AL1 Minimal	AL2023 Minimal
librepo		1.14.5
libreport-filessystem		2.15.2
libseccomp		2.5.3
libselenium	2.1.10	3.4
libselenium-utils	2.1.10	3.4
libsemanage	2.1.6	3.4
libsepol	2.1.7	3.4
libsigsegv		2.13
libsmartcols	2.23.2	2.37.4
libsolv		0.7.22
libss	1.43.5	1.46.5
libssh2	1.4.2	
libstdc++		11.4.1
libstdc++72	7.2.1	
libsysfs	2.1.0	
libtasn1	2.3	4.19.0
libtextstyle		0.21
libudev	173	
libunistring	0.9.3	0.9.10
libuser	0.60	0.63

Package	AL1 Minimal	AL2023 Minimal
libutempter	1.1.5	1.2.1
libuuid	2.23.2	2.37.4
libverto	0.2.5	0.3.2
libxcrypt		4.4.33
libxml2	2.9.1	2.10.4
libyaml	0.1.6	0.2.5
libzstd		1.5.5
linux-firmware-wheel		20210208
logrotate	3.7.8	3.20.1
lua	5.1.4	
lua-libs		5.4.4
lz4-libs		1.9.4
make	3.82	
man-db		2.9.3
microcode_ctl	2.1	2.1
mingetty	1.08	
mpfr		4.1.0
ncurses	5.7	6.2
ncurses-base	5.7	6.2
ncurses-libs	5.7	6.2

Package	AL1 Minimal	AL2023 Minimal
nettle		3.8
net-tools	1.60	2.0
newt	0.52.11	
newt-python27	0.52.11	
npth		1.6
nspr	4.25.0	
nss	3.53.1	
nss-pem	1.0.3	
nss-softokn	3.53.1	
nss-softokn-freebl	3.53.1	
nss-sysinit	3.53.1	
nss-tools	3.53.1	
nss-util	3.53.1	
ntp	4.2.8p15	
ntpd	4.2.8p15	
numactl-libs		2.0.14
oniguruma		6.9.7.1
openldap	2.4.40	2.4.57
openssh	7.4p1	8.7p1
openssh-clients		8.7p1

Package	AL1 Minimal	AL2023 Minimal
openssh-server	7.4p1	8.7p1
openssl	1.0.2k	3.0.8
openssl-lib		3.0.8
openssl-pkcs11		0.4.12
os-prober		1.77
p11-kit	0.18.5	0.24.1
p11-kit-trust	0.18.5	0.24.1
pam	1.1.8	1.5.1
passwd	0.79	0.80
pciutils	3.1.10	3.7.0
pciutils-lib	3.1.10	3.7.0
pcre	8.21	
pcre2		10.40
pcre2-syntax		10.40
pinentry	0.7.6	
pkgconfig	0.27.1	
policycoreutils	2.1.12	3.4
popt	1.13	1.18
procmail	3.22	
procps	3.2.8	

Package	AL1 Minimal	AL2023 Minimal
procps-ng		3.3.17
psmisc	22.20	23.4
pth	2.0.7	
publicsuffix-list-dafsa		20240212
python27	2.7.18	
python27-babel	0.9.4	
python27-backports	1.0	
python27-backports-ssl_match_hostname	3.4.0.2	
python27-chardet	2.0.1	
python27-configobj	4.7.2	
python27-iniparse	0.3.1	
python27-jinja2	2.7.2	
python27-jsonpatch	1.2	
python27-jsonpointer	1.0	
python27-libs	2.7.18	
python27-markupsafe	0.11	
python27-pycurl	7.19.0	
python27-pygpme	0.3	
python27-pyliblzma	0.5.3	

Package	AL1 Minimal	AL2023 Minimal
python27-pyxattr	0.5.0	
python27-PyYAML	3.10	
python27-requests	1.2.3	
python27-setuptools	36.2.7	
python27-six	1.8.0	
python27-urlgrabber	3.10	
python27-urllib3	1.24.3	
python3		3.9.16
python3-attrs		20.3.0
python3-audit		3.0.6
python3-awscrt		0.19.19
python3-babel		2.9.1
python3-cffi		1.14.5
python3-chardet		4.0.0
python3-colorama		0.4.4
python3-configobj		5.0.6
python3-cryptography		36.0.1
python3-dateutil		2.8.1
python3-dbus		1.2.18
python3-distro		1.5.0

Package	AL1 Minimal	AL2023 Minimal
python3-dnf		4.14.0
python3-dnf-plugins-core		4.3.0
python3-docutils		0.16
python3-gpg		1.15.1
python3-hawkey		0.69.0
python3-idna		2.10
python3-jinja2		2.11.3
python3-jmespath		0.10.0
python3-jsonpatch		1.21
python3-jsonpointer		2.0
python3-jjsonschema		3.2.0
python3-libcomps		0.1.20
python3-libdnf		0.69.0
python3-libs		3.9.16
python3-libselenium		3.4
python3-libsemanage		3.4
python3-markupsafe		1.1.1
python3-netifaces		0.10.6
python3-oauthlib		3.0.2
python3-pip-wheel		21.3.1

Package	AL1 Minimal	AL2023 Minimal
python3-ply		3.11
python3-policycore utils		3.4
python3-prettytable		0.7.2
python3-prompt-too lkit		3.0.24
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
python3-requests		2.25.1
python3-rpm		4.16.1.3
python3-ruamel-yaml		0.16.6
python3-ruamel-yaml- clib		0.1.2
python3-setools		4.4.1
python3-setuptools		59.6.0
python3-setuptools- wheel		59.6.0
python3-six		1.15.0

Package	AL1 Minimal	AL2023 Minimal
python3-systemd		235
python3-urllib3		1.25.10
python3-wcwidth		0.2.5
readline	6.2	8.1
rng-tools		6.14
rootfiles	8.1	8.1
rpm	4.11.3	4.16.1.3
rpm-build-libs	4.11.3	4.16.1.3
rpm-libs	4.11.3	4.16.1.3
rpm-plugin-selinux		4.16.1.3
rpm-plugin-systemd-inhibit		4.16.1.3
rpm-python27	4.11.3	
rpm-sign-libs		4.16.1.3
rsyslog	5.8.10	
sbsigntools		0.9.4
sed	4.2.1	4.8
selinux-policy		38.1.45
selinux-policy-targeted		38.1.45
sendmail	8.14.4	

Package	AL1 Minimal	AL2023 Minimal
setserial	2.17	
setup	2.8.14	2.13.7
shadow-utils	4.1.4.2	4.9
shared-mime-info	1.1	
slang	2.2.1	
sqlite	3.7.17	
sqlite-libs		3.40.0
sudo	1.8.23	1.9.15
sysctl-defaults	1.0	1.0
sysfsutils	2.1.0	
systemd		252.23
systemd-libs		252.23
systemd-networkd		252.23
systemd-pam		252.23
systemd-resolved		252.23
systemd-udev		252.23
system-release	2018.03	2023.6.20241031
sysvinit	2.87	
tar	1.26	1.34
tcp_wrappers-libs	7.6	

Package	AL1 Minimal	AL2023 Minimal
tzdata	2023c	2024a
udev	173	
update-motd	1.0.1	2.2
upstart	0.6.5	
userspace-rcu		0.12.1
ustr	1.0.4	
util-linux	2.23.2	2.37.4
util-linux-core		2.37.4
vim-data	9.0.2120	9.0.2153
vim-minimal	9.0.2120	9.0.2153
which	2.19	2.21
xfsprogs		5.18.0
xz	5.2.2	5.2.5
xz-libs	5.2.2	5.2.5
yum	3.4.3	4.14.0
yum-metadata-parser	1.1.4	
yum-plugin-priorities	1.1.31	
yum-plugin-upgrade-helper	1.1.31	
zlib	1.2.8	1.2.11

Package	AL1 Minimal	AL2023 Minimal
zram-generator		1.1.2
zram-generator-def aults		1.1.2
zstd		1.5.5

Comparing packages installed on Amazon Linux 1 (AL1) and Amazon Linux 2023 base container images

A comparison of the RPMs present on the AL1 and AL2023 base container images.

Package	AL1 Container	AL2023 Container
alternatives		1.15
amazon-linux-repo- cdn		2023.6.20241031
audit-libs		3.0.6
basesystem	10.0	11
bash	4.2.46	5.2.15
bzip2-libs	1.0.6	1.0.8
ca-certificates	2023.2.62	2023.2.68
chkconfig	1.3.49.3	
coreutils	8.22	
coreutils-single		8.32
crypto-policies		20220428

Package	AL1 Container	AL2023 Container
curl	7.61.1	
curl-minimal		8.5.0
cyrus-sasl-lib	2.1.23	
db4	4.7.25	
db4-utils	4.7.25	
dnf		4.14.0
dnf-data		4.14.0
elfutils-default-y ama-scope		0.188
elfutils-libelf	0.168	0.188
elfutils-libs		0.188
expat	2.1.0	2.5.0
file-libs	5.37	5.39
filesystem	2.4.30	3.14
gawk	3.1.7	5.1.0
gdbm	1.8.0	
gdbm-libs		1.19
glib2	2.36.3	2.74.7
glibc	2.17	2.34
glibc-common	2.17	2.34

Package	AL1 Container	AL2023 Container
glibc-minimal-langpack		2.34
gmp	6.0.0	6.2.1
gnupg2	2.0.28	
gnupg2-minimal		2.3.7
gpgme	1.4.3	1.15.1
grep	2.20	3.8
gzip	1.5	
info	5.1	
json-c		0.14
keyutils-libs	1.5.8	1.6.3
krb5-libs	1.15.1	1.21.3
libacl	2.2.49	2.3.1
libarchive		3.7.4
libassuan	2.0.3	2.5.5
libattr	2.4.46	2.5.1
libblkid		2.37.4
libcap	2.16	2.48
libcap-ng		0.8.2
libcom_err	1.43.5	1.46.5
libcomps		0.1.20

Package	AL1 Container	AL2023 Container
libcurl	7.61.1	
libcurl-minimal		8.5.0
libdnf		0.69.0
libffi	3.0.13	3.4.4
libgcc		11.4.1
libgcc72	7.2.1	
libgcrypt	1.5.3	1.10.2
libgomp		11.4.1
libgpg-error	1.11	1.42
libicu	50.2	
libidn2	2.3.0	2.3.2
libmodulemd		2.13.0
libmount		2.37.4
libnghttp2	1.33.0	1.59.0
libpsl	0.6.2	0.21.1
librepo		1.14.5
libreport-filessystem		2.15.2
libselinux	2.1.10	3.4
libsepol	2.1.7	3.4
libsigsegv		2.13

Package	AL1 Container	AL2023 Container
libsmartcols		2.37.4
libsolv		0.7.22
libssh2	1.4.2	
libstdc++		11.4.1
libstdc++72	7.2.1	
libtasn1	2.3	4.19.0
libunistring	0.9.3	0.9.10
libuuid		2.37.4
libverto	0.2.5	0.3.2
libxcrypt		4.4.33
libxml2	2.9.1	2.10.4
libxml2-python27	2.9.1	
libyaml		0.2.5
libzstd		1.5.5
lua	5.1.4	
lua-libs		5.4.4
lz4-libs		1.9.4
make	3.82	
mpfr		4.1.0
ncurses	5.7	

Package	AL1 Container	AL2023 Container
ncurses-base	5.7	6.2
ncurses-libs	5.7	6.2
npth		1.6
nspr	4.25.0	
nss	3.53.1	
nss-pem	1.0.3	
nss-softokn	3.53.1	
nss-softokn-freebl	3.53.1	
nss-sysinit	3.53.1	
nss-tools	3.53.1	
nss-util	3.53.1	
openldap	2.4.40	
openssl	1.0.2k	
openssl-libs		3.0.8
p11-kit	0.18.5	0.24.1
p11-kit-trust	0.18.5	0.24.1
pcre	8.21	
pcre2		10.40
pcre2-syntax		10.40
pinentry	0.7.6	

Package	AL1 Container	AL2023 Container
pkgconfig	0.27.1	
popt	1.13	1.18
pth	2.0.7	
publicsuffix-list-dafsa		20240212
python27	2.7.18	
python27-chardet	2.0.1	
python27-iniparse	0.3.1	
python27-kitchen	1.1.1	
python27-libs	2.7.18	
python27-pycurl	7.19.0	
python27-pygpme	0.3	
python27-pyliblzma	0.5.3	
python27-pyattr	0.5.0	
python27-urlgrabber	3.10	
python3		3.9.16
python3-dnf		4.14.0
python3-gpg		1.15.1
python3-hawkey		0.69.0
python3-libcomps		0.1.20
python3-libdnf		0.69.0

Package	AL1 Container	AL2023 Container
python3-libs		3.9.16
python3-pip-wheel		21.3.1
python3-rpm		4.16.1.3
python3-setuptools-wheel		59.6.0
readline	6.2	8.1
rpm	4.11.3	4.16.1.3
rpm-build-libs	4.11.3	4.16.1.3
rpm-libs	4.11.3	4.16.1.3
rpm-python27	4.11.3	
rpm-sign-libs		4.16.1.3
sed	4.2.1	4.8
setup	2.8.14	2.13.7
shared-mime-info	1.1	
sqlite	3.7.17	
sqlite-libs		3.40.0
sysctl-defaults	1.0	
system-release	2018.03	2023.6.20241031
tar	1.26	
tzdata	2023c	2024a
xz-libs	5.2.2	5.2.5

Package	AL1 Container	AL2023 Container
yum	3.4.3	4.14.0
yum-metadata-parser	1.1.4	
yum-plugin-ovl	1.1.31	
yum-plugin-priorities	1.1.31	
yum-utils	1.1.31	
zlib	1.2.8	1.2.11

AL2023 system requirements

This section describes the system requirements for using AL2023.

Topics

- [CPU requirements for running AL2023](#)
- [Memory \(RAM\) requirements for running AL2023](#)

CPU requirements for running AL2023

To run any AL2023 code, the processor used needs to meet certain minimum requirements. Attempts to run AL2023 on CPUs that do not meet these requirements might result in illegal instruction errors very early in code execution.

The minimum requirements apply to [AL2023 on Amazon EC2](#), [AL2023 in containers](#), and [AL2023 outside Amazon EC2](#).

ARM CPU Requirements for AL2023

All AL2023 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, AL2023 only supports instance types that use Graviton2 or later processors. AL2023 doesn't support A1 instances.

AL2023 requires an ARMv8.2 compliant processor with the Cryptography Extension (ARMv8.2+crypto). All AL2023 packages for aarch64 are built with the `-march=armv8.2-a+crypto` compiler flag. Although we attempt to print graceful error messages when AL2023 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 AL2023 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 AL2023

All AL2023 x86-64 binaries are built for the x86-64v2 revision of the x86-64 architecture by passing `-march=x86-64-v2` to the compiler.

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

- CMPXCHG16B
- LAHF-SAHF
- POPCNT
- SSE3
- SSE4_1
- SSE4_2
- SSSE3

This roughly maps to x86-64 processors released in 2009 or later. Examples include the Intel Nehalem, AMD Jaguar, Atom Silvermont, along with the VIA Nano and Eden C microarchitectures.

In Amazon EC2, all x86-64 instance types support x86-64v2, including M1, C1, and M2 instance families.

No 32-bit x86 (i686) AL2023 binaries are built. Although AL2023 retains support for running 32-bit userspace binaries, this functionality is deprecated and might be removed in a future major version of Amazon Linux. For more information, see [32bit x86 \(i686\) Packages](#).

Memory (RAM) requirements for running AL2023

The Amazon EC2 .nano family of instance types (t2.nano, t3.nano, t3a.nano, and t4g.nano) have 512 MB RAM which is the minimum requirement for AL2023.

Note

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

AL2023 images have not been tested on systems with less than 512 MB RAM. Running AL2023 based container images in less than 512 MB RAM will be dependent on the containerized workload.

Some workloads, such as `dnf upgrade` between some AL2023 releases can require more than 512 MB RAM. For this reason, the [AL2023.3](#) release introduced enabling `zram` by default for instances with less than 800 MB of RAM. For containerized workloads, this means that some workloads might run fine on AL2023 instances with this amount of memory, but fail when run in a container restricted to this amount of memory usage.

For instance types with less than 800 MB of RAM, AL2023 (as of [AL2023.3](#) or newer) will enable `zram` based swap by default. Examples of Amazon EC2 instance types with less than 800 MB memory include `t4g.nano`, `t3a.nano`, `t3.nano`, `t2.nano`, and `t1.micro`. This means fewer out of memory scenarios for these instance types, because AL2023 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.

AL2023 Graphical Desktop

Amazon Linux 2023 provides an optional, lightweight, cloud-optimized graphical interface based on GNOME as of release 2023.7. This modern desktop environment delivers enhanced productivity features with built-in tools like Firefox for secure browsing, while maintaining Amazon DCV and VNC support for remote access.

Related topics

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

- [Tutorial: Install the GNOME desktop environment on AL2023](#)

Supplementary Packages for Amazon Linux

This section introduces Supplementary Packages for Amazon Linux (SPAL) and outlines its benefits and limitations, along with guidelines for reporting package-related issues.

Topics

- [What is Supplementary Packages for Amazon Linux \(or SPAL\)?](#)
- [Benefits](#)
- [Support of SPAL packages](#)
- [Reporting package-related issues](#)
- [Related topics](#)
- [Tutorial: Configure SPAL repository on AL2023](#)

What is Supplementary Packages for Amazon Linux (or SPAL)?

Supplementary Packages for Amazon Linux (SPAL) is a dedicated package repository that provides access to thousands of additional packages derived from [Extra Packages for Enterprise Linux 9 \(EPEL9\)](#). These packages complement the existing software available in core Amazon Linux 2023.

SPAL simplifies software deployment by providing pre-built packages that are compatible with AL2023, eliminating the need for customers to build packages from source code themselves. This saves time and effort in the software installation process.

Note

SPAL is available in all AWS Commercial Regions, including the AWS GovCloud (US) Regions and China, for AL2023 instances with a release version of `2023.9.20251117` or later.

Benefits

SPAL provides several key advantages to Amazon Linux 2023 users:

- **Expanding AL2023 use cases** — Access to popular additional packages beyond the core AL2023 repository, such as `pandoc`, `GDAL` or `drbd-utils`, enables customers to support multiple business and development needs.

- **Simplifying AL2023 package management** — By providing packages pre-built for AL2023, the process of building additional packages from source is eliminated, saving time and reducing the risk of compilation errors.
- **Streamlining AL2 to AL2023 migration** — SPAL repository enables seamless migration of workloads from AL2 to AL2023, including workloads that depend on EPEL7 packages.

Support of SPAL packages

SPAL packages are not covered by the same level of support as core AL2023 packages, which receive support for the entire life of Amazon Linux 2023.

Important

Prior to using SPAL, customers must carefully evaluate the following considerations:

- SPAL packages **are NOT covered** by AWS Enterprise Support.
- SPAL packages **are provided 'as-is'** from upstream EPEL9.
- SPAL packages **will NOT receive** AWS CVE security tracking.
- SPAL packages receive security patches and bug fixes **exclusively from upstream EPEL9 when available**.

Reporting package-related issues

If you encounter an issue with an SPAL package, we recommend to first check if the same issue occurs with the corresponding package in the upstream EPEL9 repository. For this, please consult the [list of issues](#) on the upstream EPEL repository.

If the issue is not present in EPEL9, create an issue in the [Amazon Linux 2023 GitHub repository](#), as this indicates the problem is specific to the SPAL package build or configuration.

This approach ensures issues are addressed by the appropriate maintainers and contributes to the overall quality of both SPAL and upstream EPEL packages.

Note

The reported issues will be handled on a best-effort basis.

Related topics

For information on configuring SPAL on your system, consult the following documentation page:

- [Tutorial: Configure SPAL repository on AL2023](#)

Tutorial: Configure SPAL repository on AL2023

Supplementary Packages for Amazon Linux (SPAL) is an additional package repository for AL2023 that provides customers access to thousands of open-source packages.

The following tutorial helps you configure the SPAL repository on your AL2023 instance. By installing the repository, you will gain access to all RPM packages available in SPAL. Once installed, you can use your package manager to install and use these packages on your system.

Contents

- [Prerequisites](#)
- [Checking prerequisites](#)
- [Installing SPAL on your system](#)
- [Installing SPAL packages](#)
- [Uninstalling SPAL repository from your system](#)
- [Related topics](#)

Prerequisites

This tutorial assumes that you have already launched an instance using AL2023 release version 2023.9.20251117 or later. For more information, see the [AL2023 on Amazon EC2](#) and [Updating AL2023](#) pages.

Checking prerequisites

- To verify your instance satisfies the prerequisites, you can check the version of system-release installed on your system.

To check the version of the package, you can use the following command.

```
[ec2-user ~]$ rpm -qi system-release
```

The command will display information about the package, including the major version.

```
Name       : system-release
Version    : 2023.9.20251117
...
```

Note

Make sure to have the latest version of `system-release` installed. You can run `sudo dnf upgrade` to update to the latest version.

Installing SPAL on your system

1. Install the `spal-release` package on your system. This adds the `.repo` configuration file and the GPG keys to your system.

```
[ec2-user ~]$ sudo dnf install spal-release
```

Note

During the installation, the support statement will be displayed. The statement explains SPAL's scope of support and limitations. Please take time to review this information carefully.

2. Verify the SPAL repository configuration was successfully added to your system.

```
[ec2-user ~]$ cat /etc/yum.repos.d/amazonlinux-spal.repo
```

You should see the two repositories configured on your system: `amazonlinux-spal` and `amazonlinux-spal-source`

You can also check the list of configured repositories by running `dnf repolist`.

```
[ec2-user ~]$ dnf repolist --all
```

Note

The `--all` flag is required to see both enabled and disabled repositories.

Both SPAL repositories should be available. Note that the **Amazon Linux 2023 SPAL repository - Source packages** repository is disabled by default.

repo id	repo name
amazonlinux-spal	Amazon Linux 2023 SPAL repository
enabled	
amazonlinux-spal-source	Amazon Linux 2023 SPAL repository - Source packages
disabled	

3. (Optional) Enable the source repository.

Note

RPM source (SRPM) repositories are typically disabled by default because they are primarily used by developers for building packages, not by end-users for software installation. DNF automatically enables source repositories when you use commands that require source packages, such as `dnf download --source package`. You do not need to manually enable the source repository for one-off source package operations. Follow this step only if you want to rebuild SRPMs from SPAL on your system.

To permanently enable the **Amazon Linux 2023 SPAL repository - Source packages** repository on your system, run the following command:

```
[ec2-user ~]$ sudo dnf config-manager --enable amazonlinux-spal-source
```

Installing SPAL packages

- Install SPAL packages on your system by running `dnf install` command.

```
[ec2-user ~]$ sudo dnf install package
```

Note

You can use `dnf list` to see a complete list of SPAL packages.

```
[ec2-user ~]$ dnf list --repo=amazonlinux-spal
```

Note

SPAL is a versioned repository. Make sure to have the latest version of `system-release` installed to see the most recent list of packages.

For more information on deterministic updates, you can check [Deterministic upgrades through versioned repositories on AL2023](#)

Uninstalling SPAL repository from your system

1. Remove the SPAL repository configuration using `dnf remove` command.

```
[ec2-user ~]$ sudo dnf remove spal-release
```

2. Verify the repository was removed by running `dnf repolist` command.

```
[ec2-user ~]$ dnf repolist
```

Important

Removing the SPAL repository configuration from your system does not remove any SPAL packages installed on the system.

Related topics

For more information about the Supplementary Packages for Amazon Linux repository, see the following documentation:

- [Supplementary Packages for Amazon Linux](#)

Running applications on AL2023

This section covers methods for running applications on Amazon Linux 2023 (AL2023), including managing when they start (and restart), and controlling resource usage.

Topics

- [Limiting process resource usage in AL2023 using systemd](#)
- [Limiting process resource usage in AL2023 using cgroups](#)

Limiting process resource usage in AL2023 using systemd

On Amazon Linux 2023 (AL2023), 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 the third party [EPEL](#) repository.

For comprehensive information, see the upstream `systemd` documentation for [systemd.resource-control](#), or the man page for `systemd.resource-control` on an AL2023 instance.

The examples below 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 below 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 examples below 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 --property=CPUAccounting=1 stress-ng
--cpu 1 --timeout 10
Running as unit: run-u6.service
Press ^] three times within 1s to disconnect TTY.
stress-ng: info: [339368] setting to a 10 second run per stressor
stress-ng: info: [339368] dispatching hogs: 1 cpu
stress-ng: info: [339368] successful run completed in 10.00s
Finished with result: success
Main processes terminated with: code=exited/status=0
Service runtime: 10.068s
CPU time consumed: 9.060s
```

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, `--wait` means to wait until the service is finished, and the `--property=CPUAccounting=1` option instructs `systemd-run` to record how much CPU time is used running the process. 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 below 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 will receive a SIGXCPU signal and fail.

```
[ec2-user ~]$ systemd-run --user --tty --wait --property=CPUAccounting=1 --  
property=LimitCPU=1 stress-ng --cpu 1 --timeout 10  
Running as unit: run-u12.service  
Press ^] three times within 1s to disconnect TTY.  
stress-ng: info: [340349] setting to a 10 second run per stressor  
stress-ng: info: [340349] dispatching hogs: 1 cpu  
stress-ng: fail: [340349] cpu instance 0 corrupted bogo-ops counter, 1370 vs 0  
stress-ng: fail: [340349] cpu instance 0 hash error in bogo-ops counter and run  
flag, 3250129726 vs 0  
stress-ng: fail: [340349] metrics-check: stressor metrics corrupted, data is  
compromised  
stress-ng: info: [340349] unsuccessful run completed in 1.14s  
Finished with result: exit-code  
Main processes terminated with: code=exited/status=2  
Service runtime: 1.201s  
CPU time consumed: 1.008s
```

More commonly, you may want to restrict the percentage of CPU time that can be consumed by a particular process. In the below 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=CPUAccounting=1 --  
property=CPUQuota=10% stress-ng --cpu 1 --timeout 10  
Running as unit: run-u13.service  
Press ^] three times within 1s to disconnect TTY.  
stress-ng: info: [340664] setting to a 10 second run per stressor  
stress-ng: info: [340664] dispatching hogs: 1 cpu
```

```
stress-ng: info:  [340664] successful run completed in 10.08s
Finished with result: success
Main processes terminated with: code=exited/status=0
Service runtime: 10.140s
CPU time consumed: 1.014s
```

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](#), or the man page for `systemd.resource-control` on an AL2023 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. The [Linux Kernel documentation for cgroup-v2](#) contains extensive details about 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](#), or the man page for `systemd.resource-control` on an AL2023 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)
   Active: active (running) since Fri 2025-01-31 22:36:42 UTC; 1s ago
 Main PID: 356294 (memcached)
    Tasks: 10 (limit: 18907)
   Memory: 1.8M
      CPU: 20ms
   CGroup: /system.slice/memcached.service
          ##356294 /usr/bin/memcached -p 11211 -u memcached -m 64 -c 1024 -l
127.0.0.1,::1

Jan 31 22:35:36 ip-1-2-3-4.us-west-2.compute.internal systemd[1]: Started
memcached.service - memcached daemon.
```

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 4s 63 126 yes 24 2 0
[ec2-user ~]$ sudo systemctl status memcached.service
# memcached.service - memcached daemon
   Loaded: loaded (/usr/lib/systemd/system/memcached.service; enabled; preset:
disabled)
```

```

Active: active (running) since Fri 2025-01-31 22:36:42 UTC; 7min ago
Main PID: 356294 (memcached)
Tasks: 10 (limit: 18907)
Memory: 66.7M
CPU: 203ms
CGroup: /system.slice/memcached.service
        ##356294 /usr/bin/memcached -p 11211 -u memcached -m 64 -c 1024 -l
127.0.0.1,::1

Jan 31 22:36:42 ip-1-2-3-4.us-west-2.compute.internal systemd[1]: Started
memcached.service - memcached daemon.

```

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 below 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: /etc/systemd/system/memcached.service.d
            ##override.conf
   Active: active (running) since Fri 2025-01-31 23:09:13 UTC; 49s ago
 Main PID: 358423 (memcached)
    Tasks: 10 (limit: 18907)

```

```

Memory: 1.8M (max: 32.0M available: 30.1M)
CPU: 25ms
CGroup: /system.slice/memcached.service
        ##358423 /usr/bin/memcached -p 11211 -u memcached -m 64 -c 1024 -l
127.0.0.1,::1

Jan 31 23:09:13 ip-1-2-3-4.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: /etc/systemd/system/memcached.service.d
            ##override.conf
   Active: active (running) since Fri 2025-01-31 23:14:48 UTC; 3s ago
 Main PID: 359492 (memcached)
    Tasks: 10 (limit: 18907)
   Memory: 18.2M (max: 32.0M available: 13.7M)
      CPU: 42ms
   CGroup: /system.slice/memcached.service
           ##359492 /usr/bin/memcached -p 11211 -u memcached -m 64 -c 1024 -l
127.0.0.1,::1

Jan 31 23:14:48 ip-1-2-3-4.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: /etc/systemd/system/memcached.service.d
           ##override.conf
   Active: failed (Result: oom-kill) since Fri 2025-01-31 23:20:28 UTC; 2s ago
 Duration: 2.901s
   Process: 360130 ExecStart=/usr/bin/memcached -p ${PORT} -u ${USER} -m
${CACHE_SIZE} -c ${MAXCONN} $OPTIONS (code=killed, signal=KILL)
   Main PID: 360130 (code=killed, signal=KILL)
      CPU: 94ms

Jan 31 23:20:25 ip-1-2-3-4.us-west-2.compute.internal systemd[1]: Started
memcached.service - memcached daemon.
Jan 31 23:20:28 ip-1-2-3-4.us-west-2.compute.internal systemd[1]:
memcached.service: A process of this unit has been killed by the OOM killer.
Jan 31 23:20:28 ip-1-2-3-4.us-west-2.compute.internal systemd[1]:
memcached.service: Main process exited, code=killed, status=9/KILL
Jan 31 23:20:28 ip-1-2-3-4.us-west-2.compute.internal systemd[1]:
memcached.service: Failed with result 'oom-kill'.
```

Limiting process resource usage in AL2023 using cgroups

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, previously found in [EPEL](#).

The example below 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.

Note

Amazon Linux 2 uses cgroup-v1 rather than cgroup-v2 which is used on AL2023. On AL2, the sysfs paths are different, and there will be `/sys/fs/cgroup/memory/our-example-limits` and `/sys/fs/cgroup/cpu/our-example-limits` directories owned by ec2-user which contain files that can be used to control the limits of the cgroup.

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
```

Note

Amazon Linux 2 uses cgroup-v1 rather than the cgroup-v2 that Amazon Linux 2023 uses. This means that some tunables are different. To limit memory usage on AL2, the below tunable is used instead.

```
[ec2-user ~]$ echo 100000000 > /sys/fs/cgroup/memory/our-example-limits/  
memory.limit_in_bytes
```

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
```

Amazon Linux 2 uses `cgroup-v1` rather than the `cgroup-v2` that Amazon Linux 2023 uses. This means that some tunables are different, including how to limit CPU usage.

5. The below 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](#) contains extensive details about how they work. The documentation of the [cpu](#) and [memory](#) controllers covers the details of how to use each tunable option.

Using AL2023 on AWS

You can set up AL2023 for use with other AWS services. For example, you can choose an AL2023 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](#)
- [AL2023 on Amazon EC2](#)
- [Using AL2023 in containers](#)
- [AL2023 on AWS Elastic Beanstalk](#)
- [Using AL2023 in AWS CloudShell](#)
- [Using AL2023 based Amazon ECS AMIs to host containerized workloads](#)
- [Using Amazon Elastic File System on AL2023](#)
- [Using Amazon EMR built on AL2023](#)
- [Using AL2023 in AWS Lambda](#)

Getting started with AWS

Sign up for an AWS account

If you do not have an AWS account, complete the following steps to create one.

To sign up for an AWS account

1. Open <https://portal.aws.amazon.com/billing/signup>.
2. Follow the online instructions.

Part of the sign-up procedure involves receiving a phone call or text message and entering a verification code on the phone keypad.

When you sign up for an AWS account, an *AWS account root user* is created. The root user has access to all AWS services and resources in the account. As a security best practice, assign administrative access to a user, and use only the root user to perform [tasks that require root user access](#).

AWS sends you a confirmation email after the sign-up process is complete. At any time, you can view your current account activity and manage your account by going to <https://aws.amazon.com/> and choosing **My Account**.

Create a user with administrative access

After you sign up for an AWS account, secure your AWS account root user, enable AWS IAM Identity Center, and create an administrative user so that you don't use the root user for everyday tasks.

Secure your AWS account root user

1. Sign in to the [AWS Management Console](#) as the account owner by choosing **Root user** and entering your AWS account email address. On the next page, enter your password.

For help signing in by using root user, see [Signing in as the root user](#) in the *AWS Sign-In User Guide*.

2. Turn on multi-factor authentication (MFA) for your root user.

For instructions, see [Enable a virtual MFA device for your AWS account root user \(console\)](#) in the *IAM User Guide*.

Create a user with administrative access

1. Enable IAM Identity Center.

For instructions, see [Enabling AWS IAM Identity Center](#) in the *AWS IAM Identity Center User Guide*.

2. In IAM Identity Center, grant administrative access to a user.

For a tutorial about using the IAM Identity Center directory as your identity source, see [Configure user access with the default IAM Identity Center directory](#) in the *AWS IAM Identity Center User Guide*.

Sign in as the user with administrative access

- To sign in with your IAM Identity Center user, use the sign-in URL that was sent to your email address when you created the IAM Identity Center user.

For help signing in using an IAM Identity Center user, see [Signing in to the AWS access portal](#) in the *AWS Sign-In User Guide*.

Assign access to additional users

- In IAM Identity Center, create a permission set that follows the best practice of applying least-privilege permissions.

For instructions, see [Create a permission set](#) in the *AWS IAM Identity Center User Guide*.
- Assign users to a group, and then assign single sign-on access to the group.

For instructions, see [Add groups](#) in the *AWS IAM Identity Center User 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. <ul style="list-style-type: none">For the AWS CLI, see Login for AWS local development in the <i>AWS Command Line Interface User Guide</i>.

Which user needs programmatic access?	To	By
		<ul style="list-style-type: none"> For AWS SDKs, see Login for AWS local development in the <i>AWS SDKs and Tools Reference Guide</i>.
Workforce identity (Users managed in IAM Identity Center)	Use temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	<p>Following the instructions for the interface that you want to use.</p> <ul style="list-style-type: none"> For the AWS CLI, see Configuring the AWS CLI to use AWS IAM Identity Center in the <i>AWS Command Line Interface User Guide</i>. For AWS SDKs, tools, and AWS APIs, see IAM Identity Center authentication in the <i>AWS SDKs and Tools Reference Guide</i>.
IAM	Use temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions in Using temporary credentials with AWS resources in the <i>IAM User Guide</i> .

Which user needs programmatic access?	To	By
IAM	(Not recommended) Use long-term credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	<p>Following the instructions for the interface that you want to use.</p> <ul style="list-style-type: none"> For the AWS CLI, see Authenticating using IAM user credentials in the <i>AWS Command Line Interface User Guide</i>. For AWS SDKs and tools, see Authenticate using long-term credentials in the <i>AWS SDKs and Tools Reference Guide</i>. For AWS APIs, see Managing access keys for IAM users in the <i>IAM User Guide</i>.

AL2023 on Amazon EC2

Use one of the following procedures to launch an Amazon EC2 instance with an AL2023 AMI. You can choose either the standard AMI, or the minimal AMI. For more information about the differences between the standard AMI and the minimal AMI, see [Comparing AL2023 standard \(default\) and minimal AMIs](#).

Topics

- [Launching AL2023 using the Amazon EC2 console](#)
- [Launching AL2023 using the SSM parameter and AWS CLI](#)
- [Launching the latest AL2023 AMI using CloudFormation](#)
- [Launching AL2023 using a specific AMI ID](#)
- [AL2023 AMI deprecation and life cycle](#)

- [Connecting to AL2023 instances](#)
- [Comparing AL2023 standard and minimal AMIs](#)

Launching AL2023 using the Amazon EC2 console

Use the Amazon EC2 console to launch an AL2023 AMI.

Note

For Arm-based instances, AL2023 only supports instance types that use Graviton2 or later processors. AL2023 doesn't support A1 instances.

Use the following steps to launch an Amazon EC2 instance with an AL2023 AMI from the Amazon EC2 console.

To launch an EC2 instance with an AL2023 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 **al2023-ami**.

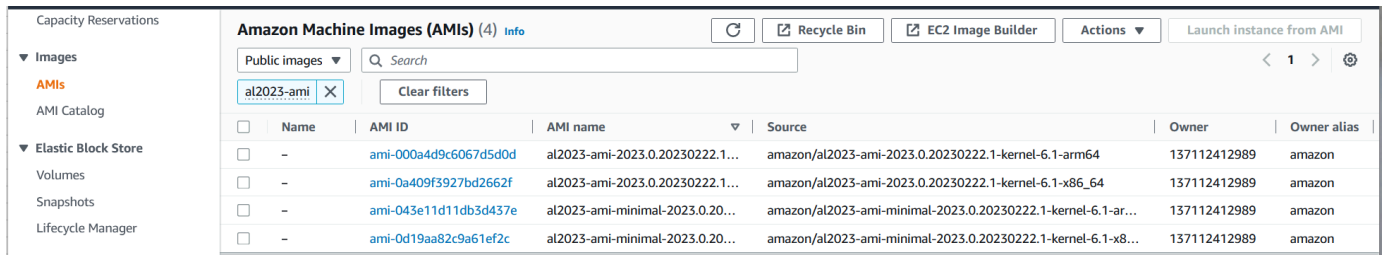
Note

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 AL2023 AMI name can be interpreted by using this format:

```
'al2023-[ami || ami-minimal]-2023.0.[release build date].[build number]-kernel-[version number]-[arm64 || x86_64]'
```

6. The following image shows a partial list of AL2023 AMIs.



The screenshot shows the Amazon Machine Images (AMIs) console. On the left is a navigation menu with options like Capacity Reservations, Images, AMIs, AMI Catalog, Elastic Block Store, Volumes, Snapshots, and Lifecycle Manager. The main panel is titled 'Amazon Machine Images (AMIs) (4) Info'. It includes a search bar with 'al2023-ami' entered, a 'Clear filters' button, and a table of AMIs. The table has columns for Name, AMI ID, AMI name, Source, Owner, and Owner alias. Four AMIs are listed, all owned by 'amazon'.

Name	AMI ID	AMI name	Source	Owner	Owner alias
-	ami-000a4d9c6067d5d0d	al2023-ami-2023.0.20230222.1...	amazon/al2023-ami-2023.0.20230222.1-kernel-6.1-arm64	137112412989	amazon
-	ami-0a409f3927bd2662f	al2023-ami-2023.0.20230222.1...	amazon/al2023-ami-2023.0.20230222.1-kernel-6.1-x86_64	137112412989	amazon
-	ami-043e11d11db3d437e	al2023-ami-minimal-2023.0.20...	amazon/al2023-ami-minimal-2023.0.20230222.1-kernel-6.1-ar...	137112412989	amazon
-	ami-0d19aa82c9a61ef2c	al2023-ami-minimal-2023.0.20...	amazon/al2023-ami-minimal-2023.0.20230222.1-kernel-6.1-x8...	137112412989	amazon

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

Launching AL2023 using the SSM parameter and AWS CLI

In the AWS CLI, you can use an AMI's SSM parameter value to launch a new instance of AL2023. More specifically, use one of the dynamic SSM parameter values from the following list, and add `/aws/service/ami-amazon-linux-latest/` before the SSM parameter value/. You use this to launch the instance in the AWS CLI.

- `al2023-ami-kernel-default-arm64` for arm64 architecture
- `al2023-ami-minimal-kernel-default-arm64` for arm64 architecture (minimal AMI)
- `al2023-ami-kernel-default-x86_64` for x86_64 architecture
- `al2023-ami-minimal-kernel-default-x86_64` for x86_64 architecture (minimal AMI)

Note

Each of the *italic* items is an example parameter. Replace them with your own information.

```
$ aws ec2 run-instances \
  --image-id \
    resolve:ssm:/aws/service/ami-amazon-linux-latest/al2023-ami-kernel-default-x86_64 \
  --instance-type m5.xlarge \
  --region us-east-1 \
  --key-name aws-key-us-east-1 \
  --security-group-ids sg-004a7650
```

The `--image-id` flag specifies the SSM parameter value.

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

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

The `--key-name` flag specifies the AWS Region's key that's used to connect to the instance. If you don't provide a key that exists in the Region where you create the instance, you can't connect to the instance using SSH.

The `--security-group-ids` flag specifies the security group that determines the 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 are placed in a default security group. In a default security group, your instance can only connect with the other instances within your VPC.

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

Launching the latest AL2023 AMI using CloudFormation

To launch an AL2023 AMI using CloudFormation, use one of the following templates.

Note

The x86_64 and Arm64 AMIs each require different instance types. For more information, see [Amazon EC2 Instance Types](#)

JSON template:

```
{
  "Parameters": {
    "LatestAmiId": {
      "Type": "AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>",
      "Default": "/aws/service/ami-amazon-linux-latest/al2023-ami-minimal-kernel-
default-x86_64"
```

```

    }
  },
  "Resources": {
    "MyEC2Instance": {
      "Type": "AWS::EC2::Instance",
      "Properties": {
        "InstanceType": "t2.large",
        "ImageId": {
          "Ref": "LatestAmiId"
        }
      }
    }
  }
}

```

YAML template:

```

Parameters:
  LatestAmiId:
    Type: 'AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>'
    Default: '/aws/service/ami-amazon-linux-latest/al2023-ami-minimal-kernel-default-x86_64'

Resources:
  Instance:
    Type: 'AWS::EC2::Instance'
    Properties:
      InstanceType: 't2.large'
      ImageId: !Ref LatestAmiId

```

Make sure to replace the AMI parameter at the end of the "Default" section, if needed. The following parameter values are available:

- al2023-ami-kernel-6.1-arm64 for arm64 architecture
- al2023-ami-minimal-kernel-6.1-arm64 for arm64 architecture (minimal AMI)
- al2023-ami-kernel-6.1-x86_64 for x86_64 architecture
- al2023-ami-minimal-kernel-6.1-x86_64 for x86_64 architecture (minimal AMI)

The following are dynamic kernel specifications. The default kernel version automatically changes with each major kernel version update.

- `al2023-ami-kernel-default-arm64` for arm64 architecture
- `al2023-ami-minimal-kernel-default-arm64` for arm64 architecture (minimal AMI)
- `al2023-ami-kernel-default-x86_64` for x86_64 architecture
- `al2023-ami-minimal-kernel-default-x86_64` for x86_64 architecture (minimal AMI)

Launching AL2023 using a specific AMI ID

You can launch a specific AL2023 AMI using the AMI ID. You can determine which AL2023 AMI ID is needed by looking at the AMI list in the Amazon EC2 console. Or, you can use AWS Systems Manager. If you're using Systems Manager, make sure to select the AMI alias from those that are listed in the previous section. For more information, see [Query for the latest Amazon Linux AMI IDs using AWS Systems Manager Parameter Store](#).

AL2023 AMI deprecation and life cycle

Each new AL2023 release includes a new AMI. When the AMI is registered, it's marked with a deprecation date. The deprecation date for each AL2023 AMI is 90 days from the time it was released to match the time period that [Kernel Live Patching on AL2023](#) is offered for each individual kernel release.

Note

The 90 day deprecation date refers to an individual AMI and doesn't refer to the AL2023 [Release cadence](#) or product support period.

For more information about AMI deprecation, see [Deprecate an AMI](#) in the *Amazon EC2 User Guide*.

Regularly using an updated AMI to launch an instance ensures that the instance starts with the latest security updates, including an updated kernel. If you launch a previous version of an AMI and apply updates, there is a period of time that the instance doesn't have the latest security updates. To ensure you're using the latest AMI, we recommend that you use SSM parameters.

For more information about using SSM parameters to launch an instance, see:

- [Launching AL2023 using the SSM parameter and AWS CLI](#)
- [Launching the latest AL2023 AMI using CloudFormation](#)

Connecting to AL2023 instances

Use SSH or AWS Systems Manager to connect to your AL2023 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 AL2023 instance, see [Connect to your Linux instance using Session Manager](#) in the *Amazon EC2 User Guide*.

Using Amazon EC2 Instance Connect

The AL2023 AMI, excluding the minimal AMI, comes with the EC2 Instance Connect agent installed by default. To use EC2 Instance Connect with an AL2023 instance launched from the minimal AMI, 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 AL2023 standard and minimal AMIs

You can launch an Amazon EC2 instance with either a standard (default) or minimal AL2023 AMI. For instructions on how to launch an Amazon EC2 instance with the standard or minimal AMI type, see [AL2023 on Amazon EC2](#).

The standard AL2023 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 AL2023 AMI is the basic, streamlined version that contains only the most basic tools and utilities necessary to run the operating system (OS). We recommend the minimal AMI if you want to have the smallest OS footprint possible. The minimal AMI offers slightly reduced disk space utilization and better long-term cost efficiency. The minimal AMI is suitable if you want a smaller OS and don't mind manually installing tools and applications.

The Container image is closer to the AL2023 minimal AMI in package set.

Comparing packages installed on Amazon Linux 2023 Images

A comparison of the RPMs present on the AL2023 AMI, Minimal AMI, and Container images.

Package	AMI	Minimal AMI	Container
acl	2.3.1		
acpid	2.0.32		
alternatives	1.15	1.15	1.15
amazon-chrony-config	4.3	4.3	
amazon-ec2-net-utils	2.5.1	2.5.1	
amazon-linux-repo-cdn			2023.6.20241031
amazon-linux-repo-s3	2023.6.20241031	2023.6.20241031	
amazon-linux-sb-keys	2023.1	2023.1	
amazon-rpm-config	228		
amazon-ssm-agent	3.3.987.0		
amd-ucode-firmware	20210208 (noarch)	20210208 (noarch)	
at	3.1.23		
attr	2.5.1		

Package	AMI	Minimal AMI	Container
audit	3.0.6	3.0.6	
audit-libs	3.0.6	3.0.6	3.0.6
aws-cfn-bootstrap	2.0		
awscli-2	2.15.30	2.15.30	
basesystem	11	11	11
bash	5.2.15	5.2.15	5.2.15
bash-completion	2.11		
bc	1.07.1		
bind-libs	9.18.28		
bind-license	9.18.28		
bind-utils	9.18.28		
binutils	2.39		
boost-filesystem	1.75.0		
boost-system	1.75.0		
boost-thread	1.75.0		
bzip2	1.0.8		
bzip2-libs	1.0.8	1.0.8	1.0.8
ca-certificates	2023.2.68	2023.2.68	2023.2.68
c-ares	1.19.1		

Package	AMI	Minimal AMI	Container
checkpolicy	3.4	3.4	
chkconfig	1.15		
chrony	4.3	4.3	
cloud-init	22.2.2	22.2.2	
cloud-init-cfg-ec2	22.2.2	22.2.2	
cloud-utils-growpart	0.31	0.31	
coreutils	8.32	8.32	
coreutils-common	8.32	8.32	
coreutils-single			8.32
cpio	2.13	2.13	
cracklib	2.9.6	2.9.6	
cracklib-dicts	2.9.6	2.9.6	
crontabs	1.11		
crypto-policies	20220428	20220428	20220428
crypto-policies-scripts	20220428		
cryptsetup	2.6.1		
cryptsetup-libs	2.6.1	2.6.1	

Package	AMI	Minimal AMI	Container
curl-minimal	8.5.0	8.5.0	8.5.0
cyrus-sasl-lib	2.1.27	2.1.27	
cyrus-sasl-plain	2.1.27		
dbus	1.12.28	1.12.28	
dbus-broker	32	32	
dbus-common	1.12.28	1.12.28	
dbus-libs	1.12.28	1.12.28	
device-mapper	1.02.185	1.02.185	
device-mapper-libs	1.02.185	1.02.185	
diffutils	3.8	3.8	
dnf	4.14.0	4.14.0	4.14.0
dnf-data	4.14.0	4.14.0	4.14.0
dnf-plugin-release-notification	1.2	1.2	
dnf-plugins-core	4.3.0	4.3.0	
dnf-plugin-support-info	1.2	1.2	
dnf-utils	4.3.0		
dosfstools	4.2		

Package	AMI	Minimal AMI	Container
dracut	055	055	
dracut-config-ec2	3.0	3.0	
dracut-config-generic	055	055	
dwz	0.14		
dyninst	10.2.1		
e2fsprogs	1.46.5	1.46.5	
e2fsprogs-libs	1.46.5	1.46.5	
ec2-hibinit-agent	1.0.8		
ec2-instance-connect	1.1		
ec2-instance-connect-selinux	1.1		
ec2-utils	2.2.0	2.2.0	
ed	1.14.2		
efi-filesystem	5	5	
efi-srpm-macros	5		
efivar	38	38	
efivar-libs	38	38	

Package	AMI	Minimal AMI	Container
elfutils- debuginfod- client	0.188		
elfutils- default-yama- scope	0.188	0.188	0.188
elfutils-libelf	0.188	0.188	0.188
elfutils-libs	0.188	0.188	0.188
ethtool	5.15		
expat	2.5.0	2.5.0	2.5.0
file	5.39	5.39	
file-libs	5.39	5.39	5.39
filesystem	3.14	3.14	3.14
findutils	4.8.0	4.8.0	
fonts-srpm- macros	2.0.5		
fstrm	0.6.1		
fuse-libs	2.9.9	2.9.9	
gawk	5.1.0	5.1.0	5.1.0
gdbm-libs	1.19	1.19	1.19
gdisk	1.0.8	1.0.8	
gettext	0.21	0.21	

Package	AMI	Minimal AMI	Container
gettext-libs	0.21	0.21	
ghc-srpm-macros	1.5.0		
glib2	2.74.7	2.74.7	2.74.7
glibc	2.34	2.34	2.34
glibc-all-langpacks	2.34	2.34	
glibc-common	2.34	2.34	2.34
glibc-gconv-extra	2.34		
glibc-locale-source	2.34	2.34	
glibc-minimal-langpack			2.34
gmp	6.2.1	6.2.1	6.2.1
gnupg2-minimal	2.3.7	2.3.7	2.3.7
gnutls	3.8.0	3.8.0	
go-srpm-macros	3.2.0		
gpgme	1.15.1	1.15.1	1.15.1
gpm-libs	1.20.7		
grep	3.8	3.8	3.8
groff-base	1.22.4	1.22.4	
grub2-common	2.06	2.06	

Package	AMI	Minimal AMI	Container
grub2-efi-aa64-ec2	2.06 (aarch64)	2.06 (aarch64)	
grub2-efi-x64-ec2	2.06 (x86_64)	2.06 (x86_64)	
grub2-pc-modules	2.06	2.06	
grub2-tools	2.06	2.06	
grub2-tools-minimal	2.06	2.06	
grubby	8.40	8.40	
gssproxy	0.8.4		
gzip	1.12	1.12	
hostname	3.23	3.23	
hunspell	1.7.0		
hunspell-en	0.20140811.1		
hunspell-en-GB	0.20140811.1		
hunspell-en-US	0.20140811.1		
hunspell-filesystem	1.7.0		
hwdata	0.384	0.384	
info	6.7		
inih	49	49	

Package	AMI	Minimal AMI	Container
initscripts	10.09	10.09	
iproute	6.10.0	6.10.0	
iputils	20210202	20210202	
irqbalance	1.9.0	1.9.0	
jansson	2.14	2.14	
jemalloc	5.2.1		
jitterentropy	3.4.1	3.4.1	
jq	1.7.1	1.7.1	
json-c	0.14	0.14	0.14
kbd	2.4.0	2.4.0	
kbd-misc	2.4.0	2.4.0	
kernel	6.1.112	6.1.112	
kernel-libbpf	6.1.112	6.1.112	
kernel-li vepatch-repo- s3	2023.6.20241031	2023.6.20241031	
kernel-srpm- macros	1.0		
kernel-tools	6.1.112		
keyutils	1.6.3		
keyutils-libs	1.6.3	1.6.3	1.6.3

Package	AMI	Minimal AMI	Container
kmod	29	29	
kmod-libs	29	29	
kpatch-runtime	0.9.7		
krb5-libs	1.21.3	1.21.3	1.21.3
less	608	608	
libacl	2.3.1	2.3.1	2.3.1
libaio	0.3.111		
libarchive	3.7.4	3.7.4	3.7.4
libargon2	20171227	20171227	
libassuan	2.5.5	2.5.5	2.5.5
libattr	2.5.1	2.5.1	2.5.1
libbasicobjects	0.1.1		
libblkid	2.37.4	2.37.4	2.37.4
libcap	2.48	2.48	2.48
libcap-ng	0.8.2	0.8.2	0.8.2
libcbor	0.7.0	0.7.0	
libcollection	0.7.0		
libcom_err	1.46.5	1.46.5	1.46.5
libcomps	0.1.20	0.1.20	0.1.20
libconfig	1.7.2		

Package	AMI	Minimal AMI	Container
libcurl-minimal	8.5.0	8.5.0	8.5.0
libdb	5.3.28	5.3.28	
libdhash	0.5.0		
libdnf	0.69.0	0.69.0	0.69.0
libeconf	0.4.0	0.4.0	
libedit	3.1	3.1	
libev	4.33		
libevent	2.1.12		
libfdisk	2.37.4	2.37.4	
libffi	3.4.4	3.4.4	3.4.4
libfido2	1.10.0	1.10.0	
libgcc	11.4.1	11.4.1	11.4.1
libgcrypt	1.10.2	1.10.2	1.10.2
libgomp	11.4.1	11.4.1	11.4.1
libgpg-error	1.42	1.42	1.42
libibverbs	48.0		
libidn2	2.3.2	2.3.2	2.3.2
libini_config	1.3.1		
libkcapi	1.4.0	1.4.0	
libkcapi-hmacalc	1.4.0	1.4.0	

Package	AMI	Minimal AMI	Container
libldb	2.6.2		
libmaxminddb	1.5.2		
libmetalink	0.1.3		
libmnl	1.0.4	1.0.4	
libmodulemd	2.13.0	2.13.0	2.13.0
libmount	2.37.4	2.37.4	2.37.4
libnfsidmap	2.5.4		
libnghttp2	1.59.0	1.59.0	1.59.0
libnl3	3.5.0		
libpath_utils	0.2.1		
libpcap	1.10.1		
libpipeline	1.5.3	1.5.3	
libpkgconf	1.8.0		
libpsl	0.21.1	0.21.1	0.21.1
libpwquality	1.4.4	1.4.4	
libref_array	0.1.5		
librepo	1.14.5	1.14.5	1.14.5
libreport-filesystem	2.15.2	2.15.2	2.15.2
libseccomp	2.5.3	2.5.3	
libselinux	3.4	3.4	3.4

Package	AMI	Minimal AMI	Container
libselinux- utils	3.4	3.4	
libsemanage	3.4	3.4	
libsepol	3.4	3.4	3.4
libsigsegv	2.13	2.13	2.13
libsmartcols	2.37.4	2.37.4	2.37.4
libsolv	0.7.22	0.7.22	0.7.22
libss	1.46.5	1.46.5	
libsss_certmap	2.9.4		
libsss_idmap	2.9.4		
libsss_ns s_idmap	2.9.4		
libsss_sudo	2.9.4		
libstdc++	11.4.1	11.4.1	11.4.1
libstoragemgmt	1.9.4		
libtalloc	2.3.4		
libtasn1	4.19.0	4.19.0	4.19.0
libtdb	1.4.7		
libtevent	0.13.0		
libtextstyle	0.21	0.21	
libtirpc	1.3.3		

Package	AMI	Minimal AMI	Container
libunistring	0.9.10	0.9.10	0.9.10
libuser	0.63	0.63	
libutempter	1.2.1	1.2.1	
libuuid	2.37.4	2.37.4	2.37.4
libuv	1.47.0		
libverto	0.3.2	0.3.2	0.3.2
libverto-libev	0.3.2		
libxcrypt	4.4.33	4.4.33	4.4.33
libxml2	2.10.4	2.10.4	2.10.4
libyaml	0.2.5	0.2.5	0.2.5
libzstd	1.5.5	1.5.5	1.5.5
linux-firmware-whence	20210208 (noarch)	20210208 (noarch)	
lm_sensors-libs	3.6.0		
lmdb-libs	0.9.29		
logrotate	3.20.1	3.20.1	
lsof	4.94.0		
lua-libs	5.4.4	5.4.4	5.4.4
lua-srpm-macros	1		
lz4-libs	1.9.4	1.9.4	1.9.4
man-db	2.9.3	2.9.3	

Package	AMI	Minimal AMI	Container
man-pages	5.10		
microcode_ctl	2.1 (x86_64)	2.1 (x86_64)	
mpfr	4.1.0	4.1.0	4.1.0
nano	5.8		
ncurses	6.2	6.2	
ncurses-base	6.2	6.2	6.2
ncurses-libs	6.2	6.2	6.2
nettle	3.8	3.8	
net-tools	2.0	2.0	
newt	0.52.21		
nfs-utils	2.5.4		
npth	1.6	1.6	1.6
nspr	4.35.0		
nss	3.90.0		
nss-softokn	3.90.0		
nss-softokn-freebl	3.90.0		
nss-sysinit	3.90.0		
nss-util	3.90.0		
ntsysv	1.15		
numactl-libs	2.0.14	2.0.14	

Package	AMI	Minimal AMI	Container
ocaml-srpm-macros	6		
oniguruma	6.9.7.1	6.9.7.1	
openblas-srpm-macros	2		
openldap	2.4.57	2.4.57	
openssh	8.7p1	8.7p1	
openssh-clients	8.7p1	8.7p1	
openssh-server	8.7p1	8.7p1	
openssl	3.0.8	3.0.8	
openssl-lib	3.0.8	3.0.8	3.0.8
openssl-pkcs11	0.4.12	0.4.12	
os-prober	1.77	1.77	
p11-kit	0.24.1	0.24.1	0.24.1
p11-kit-trust	0.24.1	0.24.1	0.24.1
package-notes-srpm-macros	0.4		
pam	1.5.1	1.5.1	
parted	3.4		
passwd	0.80	0.80	
pciutils	3.7.0	3.7.0	

Package	AMI	Minimal AMI	Container
pciutils-libs	3.7.0	3.7.0	
pcre2	10.40	10.40	10.40
pcre2-syntax	10.40	10.40	10.40
perl-Carp	1.50		
perl-Class-Struct	0.66		
perl-constant	1.33		
perl-DynaLoader	1.47		
perl-Encode	3.15		
perl-Errno	1.30		
perl-Exporter	5.74		
perl-Fcntl	1.13		
perl-File-Basename	2.85		
perl-File-Path	2.18		
perl-File-stat	1.09		
perl-File-Temp	0.231.100		
perl-Getopt-Long	2.52		
perl-Getopt-Std	1.12		
perl-HTTP-Tiny	0.078		

Package	AMI	Minimal AMI	Container
perl-if	0.60.800		
perl-interpreter	5.32.1		
perl-IO	1.43		
perl-IPC-Open3	1.21		
perl-libs	5.32.1		
perl-MIME-Base64	3.16		
perl-mro	1.23		
perl-overload	1.31		
perl-overloading	0.02		
perl-parent	0.238		
perl-PathTools	3.78		
perl-Pod-Escapes	1.07		
perl-podlators	4.14		
perl-Pod-Perldoc	3.28.01		
perl-Pod-Simple	3.42		
perl-Pod-Usage	2.01		
perl-POSIX	1.94		

Package	AMI	Minimal AMI	Container
perl-Scalar-List-Utils	1.56		
perl-SelectSaver	1.02		
perl-Socket	2.032		
perl-srpm-macros	1		
perl-Storable	3.21		
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-Local	1.300		
perl-vars	1.05		
pkgconf	1.8.0		
pkgconf-m4	1.8.0		
pkgconf-pkg-config	1.8.0		

Package	AMI	Minimal AMI	Container
polycoreutils	3.4	3.4	
polycoreutils-python-utils	3.4		
popt	1.18	1.18	1.18
procps-ng	3.3.17	3.3.17	
protobuf-c	1.4.1		
psacct	6.6.4		
psmisc	23.4	23.4	
publicsuffix-list-dafsa	20240212	20240212	20240212
python3	3.9.16	3.9.16	3.9.16
python3-attrs	20.3.0	20.3.0	
python3-audit	3.0.6	3.0.6	
python3-awscli	0.19.19	0.19.19	
python3-babel	2.9.1	2.9.1	
python3-cffi	1.14.5	1.14.5	
python3-chardet	4.0.0	4.0.0	
python3-colorama	0.4.4	0.4.4	
python3-configobj	5.0.6	5.0.6	

Package	AMI	Minimal AMI	Container
python3-cryptography	36.0.1	36.0.1	
python3-daemon	2.3.0		
python3-dateutil	2.8.1	2.8.1	
python3-dbus	1.2.18	1.2.18	
python3-distro	1.5.0	1.5.0	
python3-dnf	4.14.0	4.14.0	4.14.0
python3-dnf-plugins-core	4.3.0	4.3.0	
python3-docutils	0.16	0.16	
python3-gpg	1.15.1	1.15.1	1.15.1
python3-hawkey	0.69.0	0.69.0	0.69.0
python3-idna	2.10	2.10	
python3-jinja2	2.11.3	2.11.3	
python3-jmespath	0.10.0	0.10.0	
python3-jsonpatch	1.21	1.21	
python3-jsonpointer	2.0	2.0	

Package	AMI	Minimal AMI	Container
python3-j sonschema	3.2.0	3.2.0	
python3-l ibcomps	0.1.20	0.1.20	0.1.20
python3-libdnf	0.69.0	0.69.0	0.69.0
python3-libs	3.9.16	3.9.16	3.9.16
python3-l ibselinux	3.4	3.4	
python3-l ibsemanage	3.4	3.4	
python3-l ibstoragemgmt	1.9.4		
python3-l ockfile	0.12.2		
python3-m arkupsafe	1.1.1	1.1.1	
python3-n etifaces	0.10.6	0.10.6	
python3-o authlib	3.0.2	3.0.2	
python3-pip- wheel	21.3.1	21.3.1	21.3.1
python3-ply	3.11	3.11	
python3-p olicycoreutils	3.4	3.4	

Package	AMI	Minimal AMI	Container
python3-p rettytable	0.7.2	0.7.2	
python3-prompt- toolkit	3.0.24	3.0.24	
python3-p ycparser	2.20	2.20	
python3-p yrsistent	0.17.3	0.17.3	
python3-p yserial	3.4	3.4	
python3-pysocks	1.7.1	1.7.1	
python3-pytz	2022.7.1	2022.7.1	
python3-pyyaml	5.4.1	5.4.1	
python3-r equests	2.25.1	2.25.1	
python3-rpm	4.16.1.3	4.16.1.3	4.16.1.3
python3-ruamel- yaml	0.16.6	0.16.6	
python3-ruamel- yaml-clib	0.1.2	0.1.2	
python3-setools	4.4.1	4.4.1	
python3-s etuptools	59.6.0	59.6.0	

Package	AMI	Minimal AMI	Container
python3-s etuptools- wheel	59.6.0	59.6.0	59.6.0
python3-six	1.15.0	1.15.0	
python3-systemd	235	235	
python3-urllib3	1.25.10	1.25.10	
python3-wcwidth	0.2.5	0.2.5	
python-chevron	0.13.1		
python-srpm- macros	3.9		
quota	4.06		
quota-nls	4.06		
readline	8.1	8.1	8.1
rng-tools	6.14	6.14	
rootfiles	8.1	8.1	
rpcbind	1.2.6		
rpm	4.16.1.3	4.16.1.3	4.16.1.3
rpm-build-libs	4.16.1.3	4.16.1.3	4.16.1.3
rpm-libs	4.16.1.3	4.16.1.3	4.16.1.3
rpm-plugin- selinux	4.16.1.3	4.16.1.3	

Package	AMI	Minimal AMI	Container
rpm-plugin-systemd-inhibit	4.16.1.3	4.16.1.3	
rpm-sign-libs	4.16.1.3	4.16.1.3	4.16.1.3
rsync	3.2.6		
rust-srpm-macros	21		
sbsigntools	0.9.4	0.9.4	
screen	4.8.0		
sed	4.8	4.8	4.8
selinux-policy	38.1.45	38.1.45	
selinux-policy-targeted	38.1.45	38.1.45	
setup	2.13.7	2.13.7	2.13.7
shadow-utils	4.9	4.9	
slang	2.3.2		
sqlite-libs	3.40.0	3.40.0	3.40.0
sssd-client	2.9.4		
sssd-common	2.9.4		
sssd-kcm	2.9.4		
sssd-nfs-idmap	2.9.4		
strace	6.8		

Package	AMI	Minimal AMI	Container
sudo	1.9.15	1.9.15	
sysctl-defaults	1.0	1.0	
sysstat	12.5.6		
systemd	252.23	252.23	
systemd-libs	252.23	252.23	
systemd-networkd	252.23	252.23	
systemd-pam	252.23	252.23	
systemd-resolved	252.23	252.23	
systemd-udev	252.23	252.23	
system-release	2023.6.20241031	2023.6.20241031	2023.6.20241031
systemtap-runtime	4.8		
tar	1.34	1.34	
tbb	2020.3		
tcpdump	4.99.1		
tcsh	6.24.07		
time	1.9		
traceroute	2.1.3		
tzdata	2024a	2024a	2024a

Package	AMI	Minimal AMI	Container
unzip	6.0		
update-motd	2.2	2.2	
userspace-rcu	0.12.1	0.12.1	
util-linux	2.37.4	2.37.4	
util-linux-core	2.37.4	2.37.4	
vim-common	9.0.2153		
vim-data	9.0.2153	9.0.2153	
vim-enhanced	9.0.2153		
vim-filesystem	9.0.2153		
vim-minimal	9.0.2153	9.0.2153	
wget	1.21.3		
which	2.21	2.21	
words	3.0		
xfsdump	3.1.11		
xfsplogs	5.18.0	5.18.0	
xxd	9.0.2153		
xxhash-libs	0.8.0		
xz	5.2.5	5.2.5	
xz-libs	5.2.5	5.2.5	5.2.5
yum	4.14.0	4.14.0	4.14.0

Package	AMI	Minimal AMI	Container
zip	3.0		
zlib	1.2.11	1.2.11	1.2.11
zram-generator	1.1.2	1.1.2	
zram-generator-defaults	1.1.2	1.1.2	
zstd	1.5.5	1.5.5	

Using AL2023 in containers

Note

For more information about how to use AL2023 to host containerized workloads on Amazon ECS, see [AL2023 for Amazon ECS container hosts](#).

There are several ways that AL2023 can be used inside containers depending on the use case. The [AL2023 Base Container Image](#) is most similar to an Amazon Linux 2 container image and the AL2023 minimal AMI.

For advanced users, we offer a minimal container image, introduced in the AL2023.2 release, along with documentation that describes how to build [bare-bones containers](#).

AL2023 can also be used to host containerized workloads, either of AL2023 based container images, or containers based on other Linux distributions. You can use [AL2023 for Amazon ECS container hosts](#), or use the provided container runtime packages directly. The `docker`, `containerd`, and `nerdctl` packages are available to be installed and used on AL2023.

Topics

- [Using the AL2023 base container image](#)
- [AL2023 Minimal container image](#)
- [Building bare-bones AL2023 container images](#)

- [Comparing packages installed on Amazon Linux 2023 Container Images](#)
- [Comparing packages installed on Amazon Linux 2023 Minimal AMI and Container Images](#)

Using the AL2023 base container image

The AL2023 container image is built from the same software components that are included in the AL2023 AMI. It's available for use in any environment as a base image for Docker workloads. If you're using the Amazon Linux AMI for applications in [Amazon Elastic Compute Cloud](#) (Amazon EC2), you can containerize your applications with the Amazon Linux container image.

Use the Amazon Linux container image in your local development environment and then push your application to AWS using [Amazon Elastic Container Service](#) (Amazon ECS). For more information, see [Using Amazon ECR images with Amazon ECS](#) in the *Amazon Elastic Container Registry User Guide*.

The Amazon Linux container image is available on Amazon ECR Public. You can provide feedback for AL2023 through your designated AWS representative or by filing an issue in the [amazon-linux-2023 repo](#) on GitHub.

To pull the Amazon Linux container image from Amazon ECR Public

1. Authenticate your Docker client to the Amazon Linux Public registry. Authentication tokens are valid for 12 hours. For more information, see [Private registry authentication](#) in the *Amazon Elastic Container Registry User Guide*.

Note

The **get-login-password** command is supported using the latest version of AWS CLI version 2. For more information, see [Installing the AWS Command Line Interface](#) in the *AWS Command Line Interface User Guide*.

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

The output is as follows.

```
Login succeeded
```

2. Pull the Amazon Linux container image by running the **docker pull** command. To view the Amazon Linux container image on the Amazon ECR Public Gallery, see [Amazon ECR Public Gallery - amazonlinux](#).

Note

When you pull the AL2023 Docker container image, you can use the tags in one of the following formats:

- To get the latest version of the AL2023 container image, use the `:2023` tag.
- To get a specific version of AL2023, you can use the following format:
 - `:2023.[0-7 release quarter].[release date].[build number]`

The following examples use the tag `:2023` and pull the most recent available container image of AL2023.

```
$ docker pull public.ecr.aws/amazonlinux/amazonlinux:2023
```

3. (Optional) Run the container locally.

```
$ docker run -it --security-opt seccomp=unconfined public.ecr.aws/amazonlinux/amazonlinux:2023 /bin/bash
```

To pull the AL2023 container image from Docker Hub

1. Pull the AL2023 container image using the **docker pull** command.

```
$ docker pull amazonlinux:2023
```

2. (Optional) Run the container locally.

```
$ docker run -it amazonlinux:2023 /bin/bash
```

Note

The container image of AL2023 uses only the `dnf` package manager to install software packages. This means that there's no `amazon-linux-extras` or equivalent command to use for additional software.

AL2023 Minimal container image

Note

The standard AL2023 container images are suitable for most use cases, and adapting to the minimal container image is likely to be more work than adapting to the AL2023 base container image.

The AL2023 minimal container image, introduced in AL2023.2, differs from the base container image because it contains only the bare minimum packages needed to install other packages. The minimal container image is designed to be a minimal set of packages, not a convenient set of packages .

The AL2023 minimal container image is built from software components already available in AL2023. The key difference in the minimal container image is using `microdnf` to provide the `dnf` package manager rather than the fully featured Python based `dnf`. This enables the minimal container image to be smaller with the trade-off of not having the full feature set of the `dnf` package manager which is included in the AL2023 AMIs and base container image.

The AL2023 minimal container image forms the base of the provided `.al2023` AWS Lambda runtime environment.

For a detailed list of packages included in the minimal container image, see [Comparing packages installed on Amazon Linux 2023 Container Images](#).

Minimal Container image size

Because the AL2023 minimal container image contains fewer packages than the AL2023 base container image, it is also significantly smaller. The following table compares the container image options of current and past releases of Amazon Linux.

Note

Image Size is as-shown on [Amazon Linux on Amazon ECR Public Gallery](#).

Image	Version	Image Size	Note
Amazon Linux 1 (AL1)	2018.03.0.20230918.0	62.3MB	x86-64 only
Amazon Linux 2	2.0.20230926.0	64.2MB	aarch64 is 1.6MB larger than x86-64
Amazon Linux 2023 base container image	2023.2.20231002.0	52.4MB	
Amazon Linux 2023 minimal container image	2023.2.20231002.0-minimal	35.2MB	

Using the AL2023 Minimal Container image

The AL2023 minimal container image is available on ECR and the `2023-minimal` tag will always point to the latest AL2023 based minimal container image, while the `minimal` tag may be updated to a newer version of Amazon Linux than AL2023.

You can pull these tags using docker with the following example:

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

```
$ docker pull public.ecr.aws/amazonlinux/amazonlinux:2023-minimal
```

The following example shows a Dockerfile that takes the minimal container image and installs GCC on top of it :

```
FROM public.ecr.aws/amazonlinux/amazonlinux:2023-minimal
```

```
RUN dnf install -y gcc && dnf clean all
```

Building bare-bones AL2023 container images

The AL2023 container image is built from the same software components that are included in the AL2023 AMI. It includes a software that enables the base container layer to behave similarly to running on an Amazon EC2 instance, such as the package manager `dnf`. This section explains how you can construct a container from scratch that includes only the bare minimum dependencies needed for an application.

Note

The standard AL2023 container images are suitable for most use cases. Using the standard container image makes it easy to build on top of your image. A bare-bones container image makes it more difficult to build on top of your image.

To create a container with bare minimum dependencies for an application

1. Determine your runtime dependencies. This will vary based on your application.
2. Construct a Dockerfile / Containerfile that builds FROM `scratch`. The following example of a Dockerfile can be used to build a container that contains only the bash shell and its dependencies.

```
FROM public.ecr.aws/amazonlinux/amazonlinux:2023 as build
RUN mkdir /sysroot
RUN dnf --releasever=$(rpm -q system-release --qf '%{VERSION}') \
  --installroot /sysroot \
  -y \
  --setopt=install_weak_deps=False \
  install bash

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

- This Dockerfile works by:

1. Starting a AL2023 container named `build`. This container will be used to bootstrap the barebones container, this container is not deployed itself, but generates the container to be deployed.
2. Creating the `/sysroot` directory. This directory will be where the build container will install the dependencies needed for the barebones container. In a subsequent step, the `/sysroot` path will be packaged up to be the root directory of our barebones image.

Using the `--installroot` option to `dnf` in this manner is how we create the other AL2023 images. It's a feature of `dnf` that allows installers and image creation tooling to work.

3. Invoking `dnf` to install packages into `/sysroot`.

The `rpm -q system-release --qf '%{VERSION}'` command queries (`-q`) the `system-release` package, setting the query format (`--qf`) to print out the version of the package being queried (the `%{VERSION}` variable is the `rpm` variable for the version of the RPM).

By setting the `--releasever` argument of `dnf` to the version of `system-release` in the build container, this `Dockerfile` can be used to rebuild the barebones container whenever an updated container base image of Amazon Linux is released.

It is possible to set the `--releasever` to any Amazon Linux 2023 version, such as `2023.9.20251117`. Doing this would mean that the build container would run as the latest AL2023 version, but build the barebones container from `2023.9.20251117` regardless of what was the current AL2023 release.

The `--setopt=install_weak_deps=False` configuration option tells `dnf` to only install dependencies that are *required* rather than recommended or suggested.

4. Copying the installed system into the root of a blank (`FROM scratch`) container.
 5. Setting the `ENTRYPOINT` to be the desired binary, in this case `/bin/bash`.
3. Create an empty directory and add the content of the example in Step 2 to a file named `Dockerfile`.

```
$ mkdir al2023-barebones-bash-example
$ cd al2023-barebones-bash-example
$ cat > Dockerfile <<EOF
FROM public.ecr.aws/amazonlinux/amazonlinux:2023 as build
```

```

RUN mkdir /sysroot
RUN dnf --releasever=$(rpm -q system-release --qf '%{VERSION}') \
  --installroot /sysroot \
  -y \
  --setopt=install_weak_deps=False \
  install bash && dnf --installroot /sysroot clean all

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

```

4. Build the container by running the following command.

```
$ docker build -t al2023-barebones-bash-example
```

5. Run the container using the following command to see how minimal a bash-only container is.

```

$ docker run -it --rm al2023-barebones-bash-example
bash-5.2# rpm
bash: rpm: command not found
bash-5.2# du -sh /usr/
bash: du: command not found
bash-5.2# ls
bash: ls: command not found
bash-5.2# echo /bin/*
/bin/alias /bin/bash /bin/bashbug /bin/bashbug-64 /bin/bg /bin/catchsegv /bin/cd /
bin/command /bin/fc /bin/fg /bin/gencat /bin/getconf /bin/getent /bin/getopts /
bin/hash /bin/iconv /bin/jobs /bin/ld.so /bin/ldd /bin/locale /bin/localedef /
bin/pldd /bin/read /bin/sh /bin/sotruss /bin/sprof /bin/type /bin/tzselect /bin/
ulimit /bin/umask /bin/unalias /bin/wait /bin/zdump

```

For a more practical example, the following procedure builds a container for a C application that displays Hello World!.

1. Create an empty directory and add the C source code and Dockerfile.

```

$ mkdir al2023-barebones-c-hello-world-example
$ cd al2023-barebones-c-hello-world-example
$ cat > hello-world.c <<EOF

```

```
#include <stdio.h>
int main(void)
{
    printf("Hello World!\n");
    return 0;
}
EOF

$ cat > Dockerfile <<EOF
FROM public.ecr.aws/amazonlinux/amazonlinux:2023 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 \
    -y \
    --setopt=install_weak_deps=False \
    install glibc && dnf --installroot /sysroot clean all

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

2. Build the container using the following command.

```
$ docker build -t al2023-barebones-c-hello-world-example .
```

3. Run the container using the following command.

```
$ docker run -it --rm al2023-barebones-c-hello-world-example
Hello World!
```

Comparing packages installed on Amazon Linux 2023 Container Images

A comparison of the RPMs present on the AL2023 base container image compared with the RPMs present on the AL2023 minimal container image.

Package	Container	Minimal Container
alternatives	1.15	1.15
amazon-linux-repo-cdn	2023.6.20241031	2023.6.20241031
audit-libs	3.0.6	3.0.6
basesystem	11	11
bash	5.2.15	5.2.15
bzip2-libs	1.0.8	1.0.8
ca-certificates	2023.2.68	2023.2.68
coreutils-single	8.32	8.32
crypto-policies	20220428	20220428
curl-minimal	8.5.0	8.5.0
dnf	4.14.0	
dnf-data	4.14.0	4.14.0
elfutils-default-yama-scope	0.188	
elfutils-libelf	0.188	
elfutils-libs	0.188	
expat	2.5.0	
file-libs	5.39	5.39
filesystem	3.14	3.14
gawk	5.1.0	5.1.0

Package	Container	Minimal Container
gdbm-libs	1.19	
glib2	2.74.7	2.74.7
glibc	2.34	2.34
glibc-common	2.34	2.34
glibc-minimal-lang pack	2.34	2.34
gmp	6.2.1	6.2.1
gnupg2-minimal	2.3.7	2.3.7
gobject-introspect ion		1.73.0
gpgme	1.15.1	1.15.1
grep	3.8	3.8
json-c	0.14	0.14
keyutils-libs	1.6.3	1.6.3
krb5-libs	1.21.3	1.21.3
libacl	2.3.1	2.3.1
libarchive	3.7.4	3.7.4
libassuan	2.5.5	2.5.5
libattr	2.5.1	2.5.1
libblkid	2.37.4	2.37.4
libcap	2.48	2.48

Package	Container	Minimal Container
libcap-ng	0.8.2	0.8.2
libcom_err	1.46.5	1.46.5
libcomps	0.1.20	
libcurl-minimal	8.5.0	8.5.0
libdnf	0.69.0	0.69.0
libffi	3.4.4	3.4.4
libgcc	11.4.1	11.4.1
libgcrypt	1.10.2	1.10.2
libgomp	11.4.1	
libgpg-error	1.42	1.42
libidn2	2.3.2	2.3.2
libmodulemd	2.13.0	2.13.0
libmount	2.37.4	2.37.4
libnghttp2	1.59.0	1.59.0
libpeas		1.32.0
libpsl	0.21.1	0.21.1
librepo	1.14.5	1.14.5
libreport-filessystem	2.15.2	2.15.2
libselinux	3.4	3.4
libsepol	3.4	3.4

Package	Container	Minimal Container
libsigsegv	2.13	2.13
libsmartcols	2.37.4	2.37.4
libsolv	0.7.22	0.7.22
libstdc++	11.4.1	11.4.1
libtasn1	4.19.0	4.19.0
libunistring	0.9.10	0.9.10
libuuid	2.37.4	2.37.4
libverto	0.3.2	0.3.2
libxcrypt	4.4.33	
libxml2	2.10.4	2.10.4
libyaml	0.2.5	0.2.5
libzstd	1.5.5	1.5.5
lua-libs	5.4.4	5.4.4
lz4-libs	1.9.4	1.9.4
microdnf		3.10.0
microdnf-dnf		3.10.0
mpfr	4.1.0	4.1.0
ncurses-base	6.2	6.2
ncurses-libs	6.2	6.2
npth	1.6	1.6

Package	Container	Minimal Container
openssl-lib	3.0.8	3.0.8
p11-kit	0.24.1	0.24.1
p11-kit-trust	0.24.1	0.24.1
pcre2	10.40	10.40
pcre2-syntax	10.40	10.40
popt	1.18	1.18
publicsuffix-list-dafsa	20240212	20240212
python3	3.9.16	
python3-dnf	4.14.0	
python3-gpg	1.15.1	
python3-hawkey	0.69.0	
python3-libcomps	0.1.20	
python3-libdnf	0.69.0	
python3-libs	3.9.16	
python3-pip-wheel	21.3.1	
python3-rpm	4.16.1.3	
python3-setuptools-wheel	59.6.0	
readline	8.1	8.1
rpm	4.16.1.3	4.16.1.3

Package	Container	Minimal Container
rpm-build-libs	4.16.1.3	
rpm-libs	4.16.1.3	4.16.1.3
rpm-sign-libs	4.16.1.3	
sed	4.8	4.8
setup	2.13.7	2.13.7
sqlite-libs	3.40.0	3.40.0
system-release	2023.6.20241031	2023.6.20241031
tzdata	2024a	
xz-libs	5.2.5	5.2.5
yum	4.14.0	
zlib	1.2.11	1.2.11

Comparing packages installed on Amazon Linux 2023 Minimal AMI and Container Images

A comparison of the RPMs present on the AL2023 Minimal AMI to the RPMs present on the AL2023 base and minimal container images.

Package	Minimal AMI	Container	Minimal Container
alternatives	1.15	1.15	1.15
amazon-chrony-config	4.3		
amazon-ec2-net-utils	2.5.1		

Package	Minimal AMI	Container	Minimal Container
amazon-linux-repo-cdn		2023.6.20241031	2023.6.20241031
amazon-linux-repo-s3	2023.6.20241031		
amazon-linux-sb-keys	2023.1		
amd-ucode-firmware	20210208 (noarch)		
audit	3.0.6		
audit-libs	3.0.6	3.0.6	3.0.6
awscli-2	2.15.30		
basesystem	11	11	11
bash	5.2.15	5.2.15	5.2.15
bzip2-libs	1.0.8	1.0.8	1.0.8
ca-certificates	2023.2.68	2023.2.68	2023.2.68
checkpolicy	3.4		
chrony	4.3		
cloud-init	22.2.2		
cloud-init-cfg-ec2	22.2.2		
cloud-utils-growpart	0.31		
coreutils	8.32		

Package	Minimal AMI	Container	Minimal Container
coreutils-common	8.32		
coreutils-single		8.32	8.32
cpio	2.13		
cracklib	2.9.6		
cracklib-dicts	2.9.6		
crypto-policies	20220428	20220428	20220428
cryptsetup-libs	2.6.1		
curl-minimal	8.5.0	8.5.0	8.5.0
cyrus-sasl-lib	2.1.27		
dbus	1.12.28		
dbus-broker	32		
dbus-common	1.12.28		
dbus-libs	1.12.28		
device-mapper	1.02.185		
device-mapper-libs	1.02.185		
diffutils	3.8		
dnf	4.14.0	4.14.0	
dnf-data	4.14.0	4.14.0	4.14.0

Package	Minimal AMI	Container	Minimal Container
dnf-plugin-release-notification	1.2		
dnf-plugins-core	4.3.0		
dnf-plugin-support-info	1.2		
dracut	055		
dracut-config-ec2	3.0		
dracut-config-generic	055		
e2fsprogs	1.46.5		
e2fsprogs-libs	1.46.5		
ec2-utils	2.2.0		
efi-filesystem	5		
efivar	38		
efivar-libs	38		
elfutils-default-yama-scope	0.188	0.188	
elfutils-libelf	0.188	0.188	
elfutils-libs	0.188	0.188	

Package	Minimal AMI	Container	Minimal Container
expat	2.5.0	2.5.0	
file	5.39		
file-libs	5.39	5.39	5.39
filesystem	3.14	3.14	3.14
findutils	4.8.0		
fuse-libs	2.9.9		
gawk	5.1.0	5.1.0	5.1.0
gdbm-libs	1.19	1.19	
gdisk	1.0.8		
gettext	0.21		
gettext-libs	0.21		
glib2	2.74.7	2.74.7	2.74.7
glibc	2.34	2.34	2.34
glibc-all-langpacks	2.34		
glibc-common	2.34	2.34	2.34
glibc-locale-source	2.34		
glibc-minimal-langpack		2.34	2.34
gmp	6.2.1	6.2.1	6.2.1

Package	Minimal AMI	Container	Minimal Container
gnupg2-minimal	2.3.7	2.3.7	2.3.7
gnutls	3.8.0		
gobject-introspection			1.73.0
gpgme	1.15.1	1.15.1	1.15.1
grep	3.8	3.8	3.8
groff-base	1.22.4		
grub2-common	2.06		
grub2-efi-aa64-ec2	2.06 (aarch64)		
grub2-efi-x64-ec2	2.06 (x86_64)		
grub2-pc-modules	2.06		
grub2-tools	2.06		
grub2-tools-minimal	2.06		
grubby	8.40		
gzip	1.12		
hostname	3.23		
hwdata	0.384		
inih	49		

Package	Minimal AMI	Container	Minimal Container
initscripts	10.09		
iproute	6.10.0		
iputils	20210202		
irqbalance	1.9.0		
jansson	2.14		
jitterentropy	3.4.1		
jq	1.7.1		
json-c	0.14	0.14	0.14
kbd	2.4.0		
kbd-misc	2.4.0		
kernel	6.1.112		
kernel-libbpf	6.1.112		
kernel-libvepatch-repos3	2023.6.20241031		
keyutils-libs	1.6.3	1.6.3	1.6.3
kmod	29		
kmod-libs	29		
krb5-libs	1.21.3	1.21.3	1.21.3
less	608		
libacl	2.3.1	2.3.1	2.3.1

Package	Minimal AMI	Container	Minimal Container
libarchive	3.7.4	3.7.4	3.7.4
libargon2	20171227		
libassuan	2.5.5	2.5.5	2.5.5
libattr	2.5.1	2.5.1	2.5.1
libblkid	2.37.4	2.37.4	2.37.4
libcap	2.48	2.48	2.48
libcap-ng	0.8.2	0.8.2	0.8.2
libcbor	0.7.0		
libcom_err	1.46.5	1.46.5	1.46.5
libcomps	0.1.20	0.1.20	
libcurl-minimal	8.5.0	8.5.0	8.5.0
libdb	5.3.28		
libdnf	0.69.0	0.69.0	0.69.0
libeconf	0.4.0		
libedit	3.1		
libfdisk	2.37.4		
libffi	3.4.4	3.4.4	3.4.4
libfido2	1.10.0		
libgcc	11.4.1	11.4.1	11.4.1
libgcrypt	1.10.2	1.10.2	1.10.2

Package	Minimal AMI	Container	Minimal Container
libgomp	11.4.1	11.4.1	
libgpg-error	1.42	1.42	1.42
libidn2	2.3.2	2.3.2	2.3.2
libkcapi	1.4.0		
libkcapi-hmacalc	1.4.0		
libmnl	1.0.4		
libmodulemd	2.13.0	2.13.0	2.13.0
libmount	2.37.4	2.37.4	2.37.4
libnghttp2	1.59.0	1.59.0	1.59.0
libpeas			1.32.0
libpipeline	1.5.3		
libpsl	0.21.1	0.21.1	0.21.1
libpwquality	1.4.4		
librepo	1.14.5	1.14.5	1.14.5
libreport-filesystem	2.15.2	2.15.2	2.15.2
libseccomp	2.5.3		
libselinux	3.4	3.4	3.4
libselinux-utils	3.4		

Package	Minimal AMI	Container	Minimal Container
libsemanage	3.4		
libsepol	3.4	3.4	3.4
libsigsegv	2.13	2.13	2.13
libsmartcols	2.37.4	2.37.4	2.37.4
libsolv	0.7.22	0.7.22	0.7.22
libss	1.46.5		
libstdc++	11.4.1	11.4.1	11.4.1
libtasn1	4.19.0	4.19.0	4.19.0
libtextstyle	0.21		
libunistring	0.9.10	0.9.10	0.9.10
libuser	0.63		
libutempter	1.2.1		
libuuid	2.37.4	2.37.4	2.37.4
libverto	0.3.2	0.3.2	0.3.2
libxcrypt	4.4.33	4.4.33	
libxml2	2.10.4	2.10.4	2.10.4
libyaml	0.2.5	0.2.5	0.2.5
libzstd	1.5.5	1.5.5	1.5.5
linux-firmware-whence	20210208 (noarch)		
logrotate	3.20.1		

Package	Minimal AMI	Container	Minimal Container
lua-libs	5.4.4	5.4.4	5.4.4
lz4-libs	1.9.4	1.9.4	1.9.4
man-db	2.9.3		
microcode_ctl	2.1 (x86_64)		
microdnf			3.10.0
microdnf-dnf			3.10.0
mpfr	4.1.0	4.1.0	4.1.0
ncurses	6.2		
ncurses-base	6.2	6.2	6.2
ncurses-libs	6.2	6.2	6.2
nettle	3.8		
net-tools	2.0		
npth	1.6	1.6	1.6
numactl-libs	2.0.14		
oniguruma	6.9.7.1		
openldap	2.4.57		
openssh	8.7p1		
openssh-clients	8.7p1		
openssh-server	8.7p1		
openssl	3.0.8		

Package	Minimal AMI	Container	Minimal Container
openssl-lib	3.0.8	3.0.8	3.0.8
openssl-pkcs11	0.4.12		
os-prober	1.77		
p11-kit	0.24.1	0.24.1	0.24.1
p11-kit-trust	0.24.1	0.24.1	0.24.1
pam	1.5.1		
passwd	0.80		
pciutils	3.7.0		
pciutils-lib	3.7.0		
pcre2	10.40	10.40	10.40
pcre2-syntax	10.40	10.40	10.40
policycoreutils	3.4		
popt	1.18	1.18	1.18
procps-ng	3.3.17		
psmisc	23.4		
publicsuffix-list-dafsa	20240212	20240212	20240212
python3	3.9.16	3.9.16	
python3-attrs	20.3.0		
python3-audit	3.0.6		
python3-awscrt	0.19.19		

Package	Minimal AMI	Container	Minimal Container
python3-babel	2.9.1		
python3-cffi	1.14.5		
python3-chardet	4.0.0		
python3-colorama	0.4.4		
python3-configobj	5.0.6		
python3-cryptography	36.0.1		
python3-dateutil	2.8.1		
python3-dbus	1.2.18		
python3-distro	1.5.0		
python3-dnf	4.14.0	4.14.0	
python3-dnf-plugins-core	4.3.0		
python3-docutils	0.16		
python3-gpg	1.15.1	1.15.1	
python3-hawkey	0.69.0	0.69.0	
python3-idna	2.10		
python3-jinja2	2.11.3		

Package	Minimal AMI	Container	Minimal Container
python3-jmespath	0.10.0		
python3-jsonpatch	1.21		
python3-jsonpointer	2.0		
python3-jsonschema	3.2.0		
python3-libibcomps	0.1.20	0.1.20	
python3-libdnf	0.69.0	0.69.0	
python3-liblibs	3.9.16	3.9.16	
python3-libselinux	3.4		
python3-libsemanage	3.4		
python3-markupsafe	1.1.1		
python3-netifaces	0.10.6		
python3-oauthlib	3.0.2		
python3-pip-wheel	21.3.1	21.3.1	
python3-ply	3.11		

Package	Minimal AMI	Container	Minimal Container
python3-policycoreutils	3.4		
python3-prettytable	0.7.2		
python3-prompt-toolkit	3.0.24		
python3-pyparser	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		
python3-requests	2.25.1		
python3-rpm	4.16.1.3	4.16.1.3	
python3-ruamel-yaml	0.16.6		
python3-ruamel-yaml-clib	0.1.2		
python3-setuptools	4.4.1		

Package	Minimal AMI	Container	Minimal Container
python3-s etuptools	59.6.0		
python3-s etuptools- wheel	59.6.0	59.6.0	
python3-six	1.15.0		
python3-systemd	235		
python3-urllib3	1.25.10		
python3-wcwidth	0.2.5		
readline	8.1	8.1	8.1
rng-tools	6.14		
rootfiles	8.1		
rpm	4.16.1.3	4.16.1.3	4.16.1.3
rpm-build-libs	4.16.1.3	4.16.1.3	
rpm-libs	4.16.1.3	4.16.1.3	4.16.1.3
rpm-plugin- selinux	4.16.1.3		
rpm-plugin- systemd-inhibit	4.16.1.3		
rpm-sign-libs	4.16.1.3	4.16.1.3	
sbsigntools	0.9.4		
sed	4.8	4.8	4.8

Package	Minimal AMI	Container	Minimal Container
selinux-policy	38.1.45		
selinux-policy-targeted	38.1.45		
setup	2.13.7	2.13.7	2.13.7
shadow-utils	4.9		
sqlite-libs	3.40.0	3.40.0	3.40.0
sudo	1.9.15		
sysctl-defaults	1.0		
systemd	252.23		
systemd-libs	252.23		
systemd-networkd	252.23		
systemd-pam	252.23		
systemd-resolved	252.23		
systemd-udev	252.23		
system-release	2023.6.20241031	2023.6.20241031	2023.6.20241031
tar	1.34		
tzdata	2024a	2024a	
update-motd	2.2		
userspace-rcu	0.12.1		

Package	Minimal AMI	Container	Minimal Container
util-linux	2.37.4		
util-linux-core	2.37.4		
vim-data	9.0.2153		
vim-minimal	9.0.2153		
which	2.21		
xfspgrog	5.18.0		
xz	5.2.5		
xz-libs	5.2.5	5.2.5	5.2.5
yum	4.14.0	4.14.0	
zlib	1.2.11	1.2.11	1.2.11
zram-generator	1.1.2		
zram-generator-defaults	1.1.2		
zstd	1.5.5		

AL2023 on AWS Elastic Beanstalk

AWS Elastic Beanstalk is a service for deploying and scaling web applications and services. Upload your code and Elastic Beanstalk automatically handles the deployment — from capacity provisioning, load balancing, and auto scaling to application health monitoring. For more information, see [AWS Elastic Beanstalk](#).

To use Elastic Beanstalk, you create an application, upload an application version in the form of an application source bundle (for example, a Java .war file) to Elastic Beanstalk, and then provide some information about the application. Elastic Beanstalk automatically launches an environment

and creates and configures the AWS resources needed to run your code. For more information, see the [AWS Elastic Beanstalk Developer Guide](#).

Elastic Beanstalk Linux platforms use Amazon EC2 instances, and these instances run Amazon Linux. As of August 4, 2023, Elastic Beanstalk offers the following platform branches based on Amazon Linux 2023: Docker, Tomcat, Java SE, Node.js, PHP, and Python. Elastic Beanstalk is working on releasing support for AL2023 to more Elastic Beanstalk platforms.

The full list of Elastic Beanstalk platform support and current platforms built on top of AL2023 can be found in the [Elastic Beanstalk Linux platforms](#) section of the [Elastic Beanstalk Developer Guide](#).

You can find the Release Notes for new Elastic Beanstalk platforms and versions of existing platforms in the [Elastic Beanstalk Release Notes](#).

Using AL2023 in AWS CloudShell

AWS CloudShell is a browser-based, pre-authenticated shell that you can launch directly from the AWS Management Console. You can navigate to CloudShell from the AWS Management Console a few different ways. For more information, see [How to get started with AWS CloudShell?](#)

AWS CloudShell, which is currently based on Amazon Linux 2, will migrate to AL2023. The migration to AL2023 will begin to roll out in all AWS Regions starting on December 4, 2023. For more information about CloudShell migrating to AL2023, see [AWS CloudShell migrating from Amazon Linux 2 to Amazon Linux 2023](#).

Using AL2023 based Amazon ECS AMIs to host containerized workloads

Note

For more information on how to use AL2023 inside a container, see [AL2023 in containers](#).

Amazon Elastic Container Service (Amazon ECS) is a fully managed container orchestration service that helps you easily deploy, manage, and scale containerized applications. As a fully managed service, Amazon ECS comes with AWS configuration and operational best practices built-in. It's integrated with both AWS and third-party tools, such as Amazon Elastic Container Registry

(Amazon ECR) and Docker. This integration makes it easier for teams to focus on building the applications, not the environment. You can run and scale your container workloads across AWS Regions in the cloud, without the complexity of managing a control plane.

You can host containerized workloads on AL2023 using the AL2023 based Amazon ECS-optimized AMI. For more information, see the [Amazon ECS-optimized AMI](#)

Changes in AL2023 for Amazon ECS compared to AL2

As with AL2, AL2023 provides the base packages required to run as an Amazon ECS Linux instance. In AL2 the `containerd`, `docker`, and `ecs-init` packages were available through `amazon-linux-extras`, whereas AL2023 includes these packages in the core repositories.

With the deterministic upgrades through versioned repositories feature, every AL2023 AMI by default is locked to a specific repository version. This is also true for the AL2023 Amazon ECS optimized AMI. All updates to your environment can be carefully managed and tested prior to deployment, as well as providing an easy way to revert to the content of a prior AMI in the event of an issue. For more information on this AL2023 feature, see [Deterministic upgrades through versioned repositories on AL2023](#).

AL2023 switches to cgroup v2 over the cgroup v1 interface supported in AL2. For more information, see [Unified Control Group hierarchy \(cgroup v2\)](#).

Note

AL2023 versions prior to [2023.2.20230920](#) (the first AL2023.2 release) contained a bug in `systemd` for Out-of-Memory (OOM) handling inside a cgroup. All processes in the cgroup were always killed instead of the OOM-Killer choosing one process at a time, which is the intended behavior.

This was a regression when compared to AL2 behavior, and is fixed as of the 2023.2.20230920 release of AL2023.

The code to build the Amazon ECS-optimized AMI is available on the [amazon-ecs-ami GitHub project](#). The [release notes](#) describe which AL2023 version maps to which Amazon ECS AMI version.

Customizing the AL2023 based Amazon ECS-optimized AMI

Important

We recommend that you use the Amazon ECS optimized AL2023 AMI. For more information, see [Amazon ECS-optimized AMI](#) in the *Amazon Elastic Container Service Developer Guide*.

You can use the same build scripts that Amazon ECS uses to create custom AMIs. For more information, see [Amazon ECS-optimized Linux AMI build script](#).

Using Amazon Elastic File System on AL2023

Amazon Elastic File System (Amazon EFS) provides serverless, fully elastic file storage so that you can share file data without provisioning or managing storage capacity and performance. Amazon EFS is built to scale 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. The service manages all the file storage infrastructure for you, meaning that you can avoid the complexity of deploying, patching, and maintaining complex file system configurations.

Amazon EFS supports the Network File System version 4 (NFSv4.1 and NFSv4.0) protocol, so the applications and tools that you use today 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 EFS file system can provide a common data source for workloads and applications that are running on more than one compute instance or server.

Installing `amazon-efs-utils` on AL2023

The `amazon-efs-utils` package is available in the AL2023 repositories to be installed and used to access Amazon EFS file systems.

Install the `amazon-efs-utils` package on AL2023

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

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

Mounting an Amazon EFS file system on AL2023

After `amazon-efs-utils` is installed, you can mount an Amazon EFS file system on your AL2023 instance.

Mount an Amazon EFS file system on AL2023

- 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 so that data in transit is encrypted using TLS, or by using 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](#).

Using Amazon EMR built on AL2023

Amazon EMR is a web service that makes it easy to process vast amounts of data efficiently using Apache Hadoop and services offered by AWS.

AL2023 based Amazon EMR releases

Amazon EMR release 7.0.0 was the first release built on AL2023. With this release, AL2023 is the base operating system for Amazon EMR, bringing all the advantages of AL2023 to Amazon EMR. For more information, see the [Amazon EMR 7.0.0 release notes](#).

AL2023 based Amazon EMR on EKS

Amazon EMR on EKS 6.13 was the first release introducing AL2023 as an option. With this release, you can launch Spark with AL2023 as the operating system, together with Java 17 runtime. For more information, see the [Amazon EMR on EKS 6.13 release notes](#), and all [Amazon EMR on EKS release notes](#).

Using AL2023 in AWS Lambda

With AWS Lambda, you can run code without provisioning or managing servers. You pay only for the compute time that you consume—there's no charge when your code isn't running. You can run code for virtually any type of application or backend service—all with zero administration. Just upload your code and Lambda takes care of everything required to run and scale your code with high availability.

AL2023 provided .a12023 managed runtime and container image

The provided .a12023 base runtime is based on the [AL2023 minimal container image](#), and provides an AL2023 based Lambda managed runtime and [container base image](#). Because the provided .a12023 runtime is based on the AL2023 minimal container image, it is substantially smaller at less than 40 MB than the provided .a12 runtime at around 109 MB.

For more information, see [Lambda runtimes](#) and [Working with Lambda container images](#).

AL2023 based Lambda runtimes

Future releases of managed language runtimes, such as Node.js 20, Python 3.12, Java 21, and .NET 8, are based on AL2023 and will use provided .a12023 as the base image as described in the [announcement of AL2023 based runtimes](#).

AL2023 based Lambda functions

- [AL2023 Lambda functions written in Go](#)
- [AL2023 Lambda functions written in Rust](#)

For more information, see [Lambda runtimes](#) in the *AWS Lambda Developer Guide*.

Tutorials

The following tutorials show you how to perform common tasks using Amazon EC2 instances running Amazon Linux 2023 (AL2023). For video tutorials, see [AWS Instructional videos and labs](#).

For AL2 instructions, see [Tutorials for Amazon EC2 instances running Linux](#) in the *Amazon EC2 User Guide*.

Tutorials

- [Tutorial: Install a LAMP server on AL2023](#)
- [Tutorial: Configure SSL/TLS on AL2023](#)
- [Tutorial: Host a WordPress blog on AL2023](#)
- [Tutorial: Redis 6 to Valkey Transition on AL2023](#)
- [Tutorial: Install the GNOME desktop environment on AL2023](#)
- [Tutorial: Configure TigerVNC server on AL2023](#)
- [Using Multi-Gen LRU \(MGLRU\) on AL2023 kernels](#)

Tutorial: Install a LAMP server on AL2023

The following procedures help you install an Apache web server with PHP and [MariaDB](#) (a community-developed fork of MySQL) support on your AL2023 instance (sometimes called a LAMP web server or LAMP stack). You can use this server to host a static website or deploy a dynamic PHP application that reads and writes information to a database.

Important

These procedures are intended for use with AL2023. If you are trying to set up a LAMP web server on a different distribution, such as Ubuntu or Red Hat Enterprise Linux, this tutorial will not work. For Ubuntu, see the following Ubuntu community documentation: [ApacheMySQLPHP](#). For other distributions, see their specific documentation.

Tasks

- [Step 1: Prepare the LAMP server](#)
- [Step 2: Test your LAMP server](#)

- [Step 3: Secure the database server](#)
- [Step 4: \(Optional\) Install phpMyAdmin](#)
- [Troubleshoot](#)
- [Related topics](#)

Step 1: Prepare the LAMP server

Prerequisites

- This tutorial assumes that you have already launched a new instance using AL2023, with a public DNS name that is reachable from the internet. For more information, see [AL2023 on Amazon EC2](#). You must also have configured your security group to allow SSH (port 22), HTTP (port 80), and HTTPS (port 443) connections. For more information about these prerequisites, see [Authorize inbound traffic for your Linux instances](#) in the *Amazon EC2 User Guide*.
- The following procedure installs the latest PHP version available on AL2023, currently 8.1. If you plan to use PHP applications other than those described in this tutorial, you should check their compatibility with 8.1.

To prepare the LAMP server

1. Connect to your instance. For more information, see [Connecting to AL2023 instances](#).
2. To ensure that all of your software packages are up to date, perform a quick software update on your instance. This process might take a few minutes, but it is important to make sure that you have the latest security updates and bug fixes.

The `-y` option installs the updates without asking for confirmation. If you would like to examine the updates before installing, you can omit this option.

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

3. Install the latest versions of Apache web server and PHP packages for AL2023.

```
[ec2-user ~]$ sudo dnf install -y httpd wget php-fpm php-mysql php-json php php-devel
```

4. Install the MariaDB software packages. Use the **dnf install** command to install multiple software packages and all related dependencies at the same time.

```
[ec2-user ~]$ sudo dnf install mariadb105-server
```

You can view the current versions of these packages using the following command:

```
[ec2-user ~]$ sudo dnf info package_name
```

Example:

```
[root@ip-172-31-25-170 ec2-user]# dnf info mariadb105
Last metadata expiration check: 0:00:16 ago on Tue Feb 14 21:35:13 2023.
Installed Packages
Name           : mariadb105
Epoch         : 3
Version        : 10.5.16
Release        : 1.amzn2023.0.6
Architecture   : x86_64
Size           : 18 M
Source         : mariadb105-10.5.16-1.amzn2023.0.6.src.rpm
Repository     : @System
From repo      : amazonlinux
Summary        : A very fast and robust SQL database server
URL            : http://mariadb.org
License        : GPLv2 and LGPLv2
Description    : MariaDB is a community developed fork from MySQL - a multi-user,
                  multi-threaded
                  : SQL database server. It is a client/server implementation consisting
of
                  : a server daemon (mariabdb) and many different client programs and
libraries.
                  : The base package contains the standard MariaDB/MySQL client programs
and
                  : utilities.
```

5. Start the Apache web server.

```
[ec2-user ~]$ sudo systemctl start httpd
```

6. Use the **systemctl** command to configure the Apache web server to start at each system boot.

```
[ec2-user ~]$ sudo systemctl enable httpd
```

You can verify that **httpd** is on by running the following command:

```
[ec2-user ~]$ sudo systemctl is-enabled httpd
```

7. Add a security rule to allow inbound HTTP (port 80) connections to your instance if you have not already done so. By default, a **launch-wizard-*N*** security group was created for your instance during launch. If you did not add additional security group rules, this group contains only a single rule to allow SSH connections.
 - a. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
 - b. In the left navigator, choose **Instances**, and select your instance.
 - c. On the **Security** tab, view the inbound rules. You should see the following rule:

Port range	Protocol	Source
22	tcp	0.0.0.0/0

 **Warning**

Using 0.0.0.0/0 allows all IPv4 addresses to access your instance using SSH. This is acceptable for a short time in a test environment, but it's unsafe for production environments. In production, you authorize only a specific IP address or range of addresses to access your instance.

- d. If there is no inbound rule to allow HTTP (port 80) connections, you must the add rule now. Choose the link for the security group. Using the procedures in see [Authorize inbound traffic for your Linux instances](#), add a new inbound security rule with the following values:
 - **Type:** HTTP
 - **Protocol:** TCP
 - **Port Range:** 80
 - **Source:** Custom
8. Test your web server. In a web browser, type the public DNS address (or the public IP address) of your instance. If there is no content in `/var/www/html`, you should see the Apache test page, which will display the message "**It works!**".

You can get the public DNS for your instance using the Amazon EC2 console (check the **Public IPv4 DNS** column; if this column is hidden, choose **Preferences** (the gear-shaped icon) and toggle on **Public IPv4 DNS**).

Verify that the security group for the instance contains a rule to allow HTTP traffic on port 80. For more information, see [Add rules to security group](#).

Important

If you are not using Amazon Linux, you might also need to configure the firewall on your instance to allow these connections. For more information about how to configure the firewall, see the documentation for your specific distribution.

Apache **httpd** serves files that are kept in a directory called the Apache document root. The Amazon Linux Apache document root is `/var/www/html`, which by default is owned by root.

To allow the `ec2-user` account to manipulate files in this directory, you must modify the ownership and permissions of the directory. There are many ways to accomplish this task. In this tutorial, you add `ec2-user` to the `apache` group to give the `apache` group ownership of the `/var/www` directory and assign write permissions to the group.

To set file permissions

1. Add your user (in this case, `ec2-user`) to the `apache` group.

```
[ec2-user ~]$ sudo usermod -a -G apache ec2-user
```

2. Log out and then log back in again to pick up the new group, and then verify your membership.

- a. Log out (use the **exit** command or close the terminal window):

```
[ec2-user ~]$ exit
```

- b. To verify your membership in the `apache` group, reconnect to your instance, and then run the following command:

```
[ec2-user ~]$ groups
```

```
ec2-user adm wheel apache systemd-journal
```

3. Change the group ownership of `/var/www` and its contents to the `apache` group.

```
[ec2-user ~]$ sudo chown -R ec2-user:apache /var/www
```

4. To add group write permissions and to set the group ID on future subdirectories, change the directory permissions of `/var/www` and its subdirectories.

```
[ec2-user ~]$ sudo chmod 2775 /var/www && find /var/www -type d -exec sudo chmod 2775 {} \;
```

5. To add group write permissions, recursively change the file permissions of `/var/www` and its subdirectories:

```
[ec2-user ~]$ find /var/www -type f -exec sudo chmod 0664 {} \;
```

Now, `ec2-user` (and any future members of the `apache` group) can add, delete, and edit files in the Apache document root, enabling you to add content, such as a static website or a PHP application.

To secure your web server (Optional)

A web server running the HTTP protocol provides no transport security for the data that it sends or receives. When you connect to an HTTP server using a web browser, the URLs that you visit, the content of webpages that you receive, and the contents (including passwords) of any HTML forms that you submit are all visible to eavesdroppers anywhere along the network pathway. The best practice for securing your web server is to install support for HTTPS (HTTP Secure), which protects your data with SSL/TLS encryption.

For information about enabling HTTPS on your server, see [Tutorial: Configure SSL/TLS on AL2023](#).

Step 2: Test your LAMP server

If your server is installed and running, and your file permissions are set correctly, your `ec2-user` account should be able to create a PHP file in the `/var/www/html` directory that is available from the internet.

To test your LAMP server

1. Create a PHP file in the Apache document root.

```
[ec2-user ~]$ echo "<?php phpinfo(); ?>" > /var/www/html/phpinfo.php
```

If you get a "Permission denied" error when trying to run this command, try logging out and logging back in again to pick up the proper group permissions that you configured in [To set file permissions](#).

2. In a web browser, type the URL of the file that you just created. This URL is the public DNS address of your instance followed by a forward slash and the file name. For example:

```
http://my.public.dns.amazonaws.com/phpinfo.php
```

You should see the PHP information page:

PHP Version 8.1.7



System	Linux ip-172-31-16-77.ec2.internal 5.15.57-28.127.amzn2022.aarch64 #1 SMP Thu Aug 4 17:06:57 UTC 2022 aarch64
Build Date	Jun 7 2022 18:21:38
Build System	Linux
Build Provider	Amazon Linux
Compiler	gcc (GCC) 11.3.1 20220421 (Red Hat 11.3.1-2)
Architecture	aarch64
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc
Loaded Configuration File	/etc/php.ini
Scan this dir for additional .ini files	/etc/php.d
Additional .ini files parsed	/etc/php.d/10-opcache.ini, /etc/php.d/20-bz2.ini, /etc/php.d/20-calendar.ini, /etc/php.d/20-ctype.ini, /etc/php.d/20-curl.ini, /etc/php.d/20-dom.ini, /etc/php.d/20-exif.ini, /etc/php.d/20-fileinfo.ini, /etc/php.d/20-ftp.ini, /etc/php.d/20-gd.ini, /etc/php.d/20-gettext.ini, /etc/php.d/20-iconv.ini, /etc/php.d/20-mbstring.ini, /etc/php.d/20-mysqlnd.ini, /etc/php.d/20-pdo.ini, /etc/php.d/20-phar.ini, /etc/php.d/20-simplexml.ini, /etc/php.d/20-sockets.ini, /etc/php.d/20-sqlite3.ini, /etc/php.d/20-tokenizer.ini, /etc/php.d/20-xml.ini, /etc/php.d/20-xmlwriter.ini, /etc/php.d/20-xsl.ini, /etc/php.d/30-mysqli.ini, /etc/php.d/30-pdo_mysql.ini, /etc/php.d/30-pdo_sqlite.ini, /etc/php.d/30-xmldrader.ini
PHP API	20210902
PHP Extension	20210902
Zend Extension	420210902
Zend Extension Build	API420210902,NTS
PHP Extension Build	API20210902,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
IPv6 Support	enabled
DTrace Support	available, disabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp, compress.bzip2, phar
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, tls, tlsv1.0, tlsv1.1, tlsv1.2, tlsv1.3
Registered Stream Filters	zlib.*, string.rot13, string.toupper, string.tolower, convert.*, consumed, dechunk, bzip2.*, convert.iconv.*

This program makes use of the Zend Scripting Language Engine:
 Zend Engine v4.1.7, Copyright (c) Zend Technologies
 with Zend OPcache v8.1.7, Copyright (c), by Zend Technologies

If you do not see this page, verify that the `/var/www/html/phpinfo.php` file was created properly in the previous step. You can also verify that all of the required packages were installed with the following command.

```
[ec2-user ~]$ sudo dnf list installed httpd mariadb-server php-mysqlnd
```

If any of the required packages are not listed in your output, install them with the **sudo yum install *package*** command.

3. Delete the `phpinfo.php` file. Although this can be useful information, it should not be broadcast to the internet for security reasons.

```
[ec2-user ~]$ rm /var/www/html/phpinfo.php
```

You should now have a fully functional LAMP web server. If you add content to the Apache document root at `/var/www/html`, you should be able to view that content at the public DNS address for your instance.

Step 3: Secure the database server

The default installation of the MariaDB server has several features that are great for testing and development, but they should be disabled or removed for production servers. The **`mysql_secure_installation`** command walks you through the process of setting a root password and removing the insecure features from your installation. Even if you are not planning on using the MariaDB server, we recommend performing this procedure.

To secure the MariaDB server

1. Start the MariaDB server.

```
[ec2-user ~]$ sudo systemctl start mariadb
```

2. Run **`mysql_secure_installation`**.

```
[ec2-user ~]$ sudo mysql_secure_installation
```

- a. When prompted, type a password for the root account.
 - i. Type the current root password. By default, the root account does not have a password set. Press Enter.
 - ii. Type **Y** to set a password, and type a secure password twice. For more information about creating a secure password, see <https://identitysafe.norton.com/password-generator/>. Make sure to store this password in a safe place.

Setting a root password for MariaDB is only the most basic measure for securing your database. When you build or install a database-driven application, you typically

create a database service user for that application and avoid using the root account for anything but database administration.

- b. Type **Y** to remove the anonymous user accounts.
 - c. Type **Y** to disable the remote root login.
 - d. Type **Y** to remove the test database.
 - e. Type **Y** to reload the privilege tables and save your changes.
3. (Optional) If you do not plan to use the MariaDB server right away, stop it. You can restart it when you need it again.

```
[ec2-user ~]$ sudo systemctl stop mariadb
```

4. (Optional) If you want the MariaDB server to start at every boot, type the following command.

```
[ec2-user ~]$ sudo systemctl enable mariadb
```

Step 4: (Optional) Install phpMyAdmin

[phpMyAdmin](#) is a web-based database management tool that you can use to view and edit the MySQL databases on your EC2 instance. Follow the steps below to install and configure phpMyAdmin on your Amazon Linux instance.

Important

We do not recommend using phpMyAdmin to access a LAMP server unless you have enabled SSL/TLS in Apache; otherwise, your database administrator password and other data are transmitted insecurely across the internet. For security recommendations from the developers, see [Securing your phpMyAdmin installation](#). For general information about securing a web server on an EC2 instance, see [Tutorial: Configure SSL/TLS on AL2023](#).

To install phpMyAdmin

1. Install the required dependencies.

```
[ec2-user ~]$ sudo dnf install php-mbstring php-xml -y
```

2. Restart Apache.

```
[ec2-user ~]$ sudo systemctl restart httpd
```

- Restart php-fpm.

```
[ec2-user ~]$ sudo systemctl restart php-fpm
```

- Navigate to the Apache document root at `/var/www/html`.

```
[ec2-user ~]$ cd /var/www/html
```

- Select a source package for the latest phpMyAdmin release from <https://www.phpmyadmin.net/downloads>. To download the file directly to your instance, copy the link and paste it into a **wget** command, as in this example:

```
[ec2-user html]$ wget https://www.phpmyadmin.net/downloads/phpMyAdmin-latest-all-languages.tar.gz
```

- Create a phpMyAdmin folder and extract the package into it with the following command.

```
[ec2-user html]$ mkdir phpMyAdmin && tar -xvzf phpMyAdmin-latest-all-languages.tar.gz -C phpMyAdmin --strip-components 1
```

- Delete the *phpMyAdmin-latest-all-languages.tar.gz* tarball.

```
[ec2-user html]$ rm phpMyAdmin-latest-all-languages.tar.gz
```

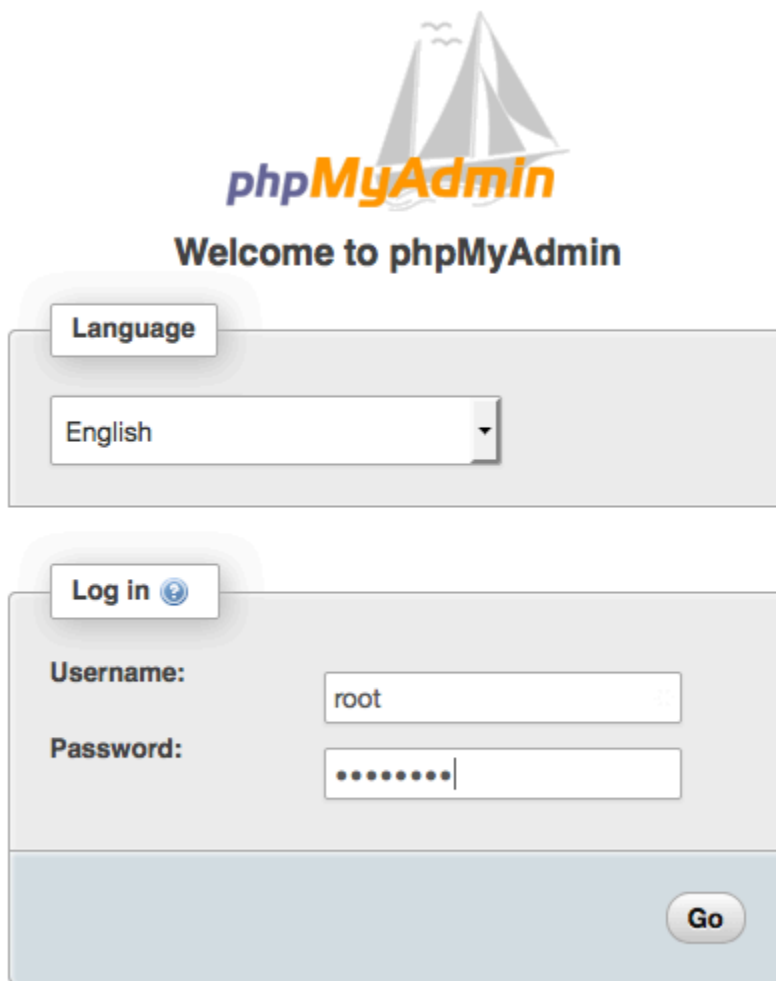
- (Optional) If the MySQL server is not running, start it now.

```
[ec2-user ~]$ sudo systemctl start mariadb
```

- In a web browser, type the URL of your phpMyAdmin installation. This URL is the public DNS address (or the public IP address) of your instance followed by a forward slash and the name of your installation directory. For example:

```
http://my.public.dns.amazonaws.com/phpMyAdmin
```

You should see the phpMyAdmin login page:



phpMyAdmin

Welcome to phpMyAdmin

Language

English

Log in ?

Username: root

Password:

Go

10. Log in to your phpMyAdmin installation with the `root` user name and the MySQL root password you created earlier.

Your installation must still be configured before you put it into service. We suggest that you begin by manually creating the configuration file, as follows:

- a. To start with a minimal configuration file, use your favorite text editor to create a new file, and then copy the contents of `config.sample.inc.php` into it.
- b. Save the file as `config.inc.php` in the phpMyAdmin directory that contains `index.php`.
- c. Refer to post-file creation instructions in the [Using the Setup script](#) section of the phpMyAdmin installation instructions for any additional setup.

For information about using phpMyAdmin, see the [phpMyAdmin User Guide](#).

Troubleshoot

This section offers suggestions for resolving common problems you might encounter while setting up a new LAMP server.

I can't connect to my server using a web browser

Perform the following checks to see if your Apache web server is running and accessible.

- **Is the web server running?**

You can verify that **httpd** is on by running the following command:

```
[ec2-user ~]$ sudo systemctl is-enabled httpd
```

If the **httpd** process is not running, repeat the steps described in [To prepare the LAMP server](#).

- **Is the firewall correctly configured?**

Verify that the security group for the instance contains a rule to allow HTTP traffic on port 80.

For more information, see [Add rules to security group](#).

I can't connect to my server using HTTPS

Perform the following checks to see if your Apache web server is configured to support HTTPS.

- **Is the web server correctly configured?**

After you install Apache, the server is configured for HTTP traffic. To support HTTPS, enable TLS on the server and install an SSL certificate. For information, see [Tutorial: Configure SSL/TLS on AL2023](#).

- **Is the firewall correctly configured?**

Verify that the security group for the instance contains a rule to allow HTTPS traffic on port 443.

For more information, see [Authorize inbound traffic for your Linux instances](#).

Related topics

For more information about transferring files to your instance or installing a WordPress blog on your web server, see the following documentation:

- [Transfer files to your Linux instance using WinSCP](#) in the *Amazon EC2 User Guide*.
- [Transfer files to Linux instances using an SCP client](#) in the *Amazon EC2 User Guide*.
- [Tutorial: Host a WordPress blog on AL2023](#)

For more information about the commands and software used in this tutorial, see the following webpages:

- Apache web server: <http://httpd.apache.org/>
- MariaDB database server: <https://mariadb.org/>
- PHP programming language: <http://php.net/>

For more information about registering a domain name for your web server, or transferring an existing domain name to this host, see [Creating and Migrating Domains and Subdomains to Amazon Route 53](#) in the *Amazon Route 53 Developer Guide*.

Tutorial: Configure SSL/TLS on AL2023

Secure Sockets Layer/Transport Layer Security (SSL/TLS) creates an encrypted channel between a web server and web client that protects data in transit from being eavesdropped on. This tutorial explains how to add support manually for SSL/TLS on an EC2 instance with AL2023 and Apache web server. This tutorial assumes that you are not using a load balancer. If you are using Elastic Load Balancing, you can choose to configure SSL offload on the load balancer, using a certificate from [AWS Certificate Manager](#) instead.

For historical reasons, web encryption is often referred to simply as SSL. While web browsers still support SSL, its successor protocol TLS is less vulnerable to attack. AL2023 disables server-side support for all versions of SSL by default. [Security standards bodies](#) consider TLS 1.0 to be unsafe. TLS 1.0 and TLS 1.1 were formally [deprecated](#) in March 2021. This tutorial contains guidance based exclusively on enabling TLS 1.2. TLS 1.3 was finalized in 2018 and is available in AL2 as long as the underlying TLS library (OpenSSL in this tutorial) is supported and enabled. [Clients must support TLS 1.2 or later by June 28, 2023](#). For more information about the updated encryption standards, see [RFC 7568](#) and [RFC 8446](#).

This tutorial refers to modern web encryption simply as TLS.

Important

These procedures are intended for use with AL2023. If you are trying to set up an EC2 instance running a different distribution, or an instance running an old version of Amazon Linux, some procedures in this tutorial might not work. For Ubuntu, see the following Ubuntu community documentation: [Open SSL on Ubuntu](#). For Red Hat Enterprise Linux, see the following: [Setting up the Apache HTTP Web Server](#). For other distributions, see their specific documentation.

Note

Alternatively, you can use AWS Certificate Manager (ACM) for AWS Nitro enclaves, which is an enclave application that allows you to use public and private SSL/TLS certificates with your web applications and servers running on Amazon EC2 instances with AWS Nitro Enclaves. Nitro Enclaves is an Amazon EC2 capability that enables creation of isolated compute environments to protect and securely process highly sensitive data, such as SSL/TLS certificates and private keys.

ACM for Nitro Enclaves works with **nginx** running on your Amazon EC2 Linux instance to create private keys, to distribute certificates and private keys, and to manage certificate renewals.

To use ACM for Nitro Enclaves, you must use an enclave-enabled Linux instance.

For more information, see [What is AWS Nitro Enclaves?](#) and [AWS Certificate Manager for Nitro Enclaves](#) in the *AWS Nitro Enclaves User Guide*.

Contents

- [Prerequisites](#)
- [Step 1: Enable TLS on the server](#)
- [Step 2: Obtain a CA-signed certificate](#)
- [Step 3: Test and harden the security configuration](#)
- [Troubleshoot](#)

Prerequisites

Before you begin this tutorial, complete the following steps:

- Launch an EBS-backed AL2023 instance. For more information, see [AL2023 on Amazon EC2](#).
- Configure your security groups to allow your instance to accept connections on the following TCP ports:
 - SSH (port 22)
 - HTTP (port 80)
 - HTTPS (port 443)

For more information, see [Authorize inbound traffic for your Linux instances](#) in the *Amazon EC2 User Guide*.

- Install the Apache web server. For step-by-step instructions, see [Tutorial: Install a LAMP server on AL2023](#). Only the httpd package and its dependencies are needed, so you can ignore the instructions involving PHP and MariaDB.
- To identify and authenticate websites, the TLS public key infrastructure (PKI) relies on the Domain Name System (DNS). To use your EC2 instance to host a public website, you need to register a domain name for your web server or transfer an existing domain name to your Amazon EC2 host. Numerous third-party domain registration and DNS hosting services are available for this, or you can use [Amazon Route 53](#).

Step 1: Enable TLS on the server

This procedure takes you through the process of setting up TLS on AL2023 with a self-signed digital certificate.

Note

A self-signed certificate is acceptable for testing but not production. If you expose your self-signed certificate to the internet, visitors to your site are greeted by security warnings.

To enable TLS on a server

1. Connect to your instance and confirm that Apache is running. For more information, see [Connecting to AL2023 instances](#).

```
[ec2-user ~]$ sudo systemctl is-enabled httpd
```

If the returned value is not "enabled," start Apache and set it to start each time the system boots.

```
[ec2-user ~]$ sudo systemctl start httpd && sudo systemctl enable httpd
```

2. To ensure that all of your software packages are up to date, perform a quick software update on your instance. This process may take a few minutes, but it is important to make sure that you have the latest security updates and bug fixes.

 **Note**

The `-y` option installs the updates without asking for confirmation. If you would like to examine the updates before installing, you can omit this option.

```
[ec2-user ~]$ sudo dnf install openssl mod_ssl
```

3. After you enter the following command, you will be taken to a prompt where you can enter information about your site.

```
[ec2-user ~]$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /  
etc/pki/tls/private/apache-selfsigned.key -out /etc/pki/tls/certs/apache-  
selfsigned.crt
```

This generates a new file `apache-selfsigned.crt` in the `/etc/pki/tls/certs/` directory. The specified file name matches the default that is assigned in the **SSLCertificateFile** directive in `/etc/httpd/conf.d/ssl.conf`.

Your instance now has the following files that you use to configure your secure server and create a certificate for testing:

- `/etc/httpd/conf.d/ssl.conf`

The configuration file for `mod_ssl`. It contains *directives* telling Apache where to find encryption keys and certificates, the TLS protocol versions to allow, and the encryption ciphers to accept. This will be your local certificate file:

- `/etc/pki/tls/certs/apache-selfsigned.crt`

This file contains both a self-signed certificate and the certificate's private key. Apache requires the certificate and key to be in PEM format, which consists of Base64-encoded ASCII characters framed by "BEGIN" and "END" lines, as in the following abbreviated example.

```
-----BEGIN PRIVATE KEY-----
MIIEvgIBADANBgkqhkiG9w0BAQEFAASCBAgEAAoIBAQD2KKx/8Zk94m1q
3gQMZF9ZN66Ls19+3tHAgQ5Fpo9KJDhzLj00CI8u1PTcGmAah5kEitCEc0wzmNeo
BCl0wYR6G0rGaKtK9Dn7CuIjvubtUysVyQoMVPQ971deakHWeRMiEJFXg6kZZ0vr
GvwnKoMh3DlK44D9dX7IDua2PlYx5+eroA+1Lqf32ZSaA00bBIMIYTHigwbHmZoT
...
56tE7THvH7v0Ef4/iU0sIrEzaMaJ0mqkmY1A70qQGKKBgBF3H1qNRNHuyMcP0DFs
27hDzPDinrquSEvoZlggkDMlh2irTiipJ/GhkvtPoQlv0fK/VXw8vSgeaBuhwJvS
LXU9HvYq0U604FgD3nAyB9hI0BE13r1HjUvbjT7moH+RhNz6eqqdsccS09VtRAO
4QQvAq0a8UheYeoXLdWcHaLP
-----END PRIVATE KEY-----

-----BEGIN CERTIFICATE-----
MIIEazCCA10gAwIBAgICWxQwDQYJKoZIhvcNAQELBQAwbGExCzAJBgNVBAYTAi0t
MRIwEAYDVQQIDAlTb21lU3RhdGUxETAPBgNVBACMFNvbWVkaXR5MRkwFwYDVQQK
DBBTb21lT3JnYW5pemF0aW9uMR8wHQYDVQQLDBZTb21lT3JnYW5pemF0aW9uYXVx
bm10MRkwFwYDVQQDDDBBpcC0xNzItMzEtMjAtMjMMSQwIgYJKoZIhvcNAQkBFhVy
...
z5rRUE/XzxRLBZ0oWZpNWTXJkQ3uFYH6s/sBwtHpKKZMz0vDedREjNKAvk4ws6F0
CuIjvubtUysVyQoMVPQ971deakHWeRMiEJFXg6kZZ0vrGvwnKoMh3DlK44D9d1U3
WanXWehT6FiSZvB4sTEXXJN2jdw8g+sHGnZ8zC0sc1knYhHrCVD2vnBlZJKSZvak
3ZazhBxtQSukFM0nWPP2a0DMMFGYUHOd0BQE8sBJxg==
-----END CERTIFICATE-----
```

The file names and extensions are a convenience and have no effect on function. For example, you can call a certificate `cert.crt`, `cert.pem`, or any other file name, so long as the related directive in the `ssl.conf` file uses the same name.

Note

When you replace the default TLS files with your own customized files, be sure that they are in PEM format.

4. Restart Apache.

```
[ec2-user ~]$ sudo systemctl restart httpd
```

Note

Make sure that TCP port 443 is accessible on your EC2 instance, as previously described.

5. Your Apache web server should now support HTTPS (secure HTTP) over port 443. Test it by entering the IP address or fully qualified domain name of your EC2 instance into a browser URL bar with the prefix **https://**.

Because you are connecting to a site with a self-signed, untrusted host certificate, your browser may display a series of security warnings. Override the warnings and proceed to the site.

If the default Apache test page opens, it means that you have successfully configured TLS on your server. All data passing between the browser and server is now encrypted.

Note

To prevent site visitors from encountering warning screens, you must obtain a trusted, CA-signed certificate that not only encrypts, but also publicly authenticates you as the owner of the site.

Step 2: Obtain a CA-signed certificate

You can use the following process to obtain a CA-signed certificate:

- Generate a certificate signing request (CSR) from a private key
- Submit the CSR to a certificate authority (CA)
- Obtain a signed host certificate
- Configure Apache to use the certificate

A self-signed TLS X.509 host certificate is cryptologically identical to a CA-signed certificate. The difference is social, not mathematical. A CA promises, at a minimum, to validate a domain's

ownership before issuing a certificate to an applicant. Each web browser contains a list of CAs trusted by the browser vendor to do this. An X.509 certificate consists primarily of a public key that corresponds to your private server key, and a signature by the CA that is cryptographically tied to the public key. When a browser connects to a web server over HTTPS, the server presents a certificate for the browser to check against its list of trusted CAs. If the signer is on the list, or accessible through a *chain of trust* consisting of other trusted signers, the browser negotiates a fast encrypted data channel with the server and loads the page.

Certificates generally cost money because of the labor involved in validating the requests, so it pays to shop around. A few CAs offer basic-level certificates free of charge. The most notable of these CAs is the [Let's Encrypt](#) project, which also supports the automation of the certificate creation and renewal process. For more information about using a Let's Encrypt certificate, see [Get Certbot](#).

If you plan to offer commercial-grade services, [AWS Certificate Manager](#) is a good option.

Underlying the host certificate is the key. As of 2019, [government](#) and [industry](#) groups recommend using a minimum key (modulus) size of 2048 bits for RSA keys intended to protect documents, through 2030. The default modulus size generated by OpenSSL in AL2023 is 2048 bits, which is suitable for use in a CA-signed certificate. In the following procedure, an optional step provided for those who want a customized key, for example, one with a larger modulus or using a different encryption algorithm.

Important

These instructions for acquiring a CA-signed host certificate do not work unless you own a registered and hosted DNS domain.

To obtain a CA-signed certificate

1. Connect to your instance and navigate to `/etc/pki/tls/private/`. This is the directory where you store the server's private key for TLS. If you prefer to use an existing host key to generate the CSR, skip to Step 3. For more information about connecting to your instance, see [Connecting to AL2023 instances](#)
2. (Optional) Generate a new private key. Here are some examples of key configurations. Any of the resulting keys works with your web server, but they vary in the degree and type of security that they implement.

- **Example 1:** Create a default RSA host key. The resulting file, **custom.key**, is a 2048-bit RSA private key.

```
[ec2-user ~]$ sudo openssl genrsa -out custom.key
```

- **Example 2:** Create a stronger RSA key with a bigger modulus. The resulting file, **custom.key**, is a 4096-bit RSA private key.

```
[ec2-user ~]$ sudo openssl genrsa -out custom.key 4096
```

- **Example 3:** Create a 4096-bit encrypted RSA key with password protection. The resulting file, **custom.key**, is a 4096-bit RSA private key encrypted with the AES-128 cipher.

Important

Encrypting the key provides greater security, but because an encrypted key requires a password, services depending on it cannot be auto-started. Each time you use this key, you must supply the password (in the preceding example, "abcde12345") over an SSH connection.

```
[ec2-user ~]$ sudo openssl genrsa -aes128 -passout pass:abcde12345 -out custom.key 4096
```

- **Example 4:** Create a key using a non-RSA cipher. RSA cryptography can be relatively slow because of the size of its public keys, which are based on the product of two large prime numbers. However, it is possible to create keys for TLS that use non-RSA ciphers. Keys based on the mathematics of elliptic curves are smaller and computationally faster when delivering an equivalent level of security.

```
[ec2-user ~]$ sudo openssl ecparam -name prime256v1 -out custom.key -genkey
```

The result is a 256-bit elliptic curve private key using prime256v1, a "named curve" that OpenSSL supports. Its cryptographic strength is slightly greater than a 2048-bit RSA key, [according to NIST](#).

Note

Not all CAs provide the same level of support for elliptic-curve-based keys as for RSA keys.

Make sure that the new private key has highly restrictive ownership and permissions (owner=root, group=root, read/write for owner only). The commands would be as shown in the following example.

```
[ec2-user ~]$ sudo chown root:root custom.key
[ec2-user ~]$ sudo chmod 600 custom.key
[ec2-user ~]$ ls -al custom.key
```

The preceding commands yield the following result.

```
-rw----- root root custom.key
```

After you have created and configured a satisfactory key, you can create a CSR.

3. Create a CSR using your preferred key. The following example uses **custom.key**.

```
[ec2-user ~]$ sudo openssl req -new -key custom.key -out csr.pem
```

OpenSSL opens a dialog and prompts you for the information shown in the following table. All of the fields except **Common Name** are optional for a basic, domain-validated host certificate.

Name	Description	Example
Country Name	The two-letter ISO abbreviation for your country.	US (=United States)
State or Province Name	The name of the state or province where your organization is located. This name cannot be abbreviated.	Washington

Name	Description	Example
Locality Name	The location of your organization, such as a city.	Seattle
Organization Name	The full legal name of your organization. Do not abbreviate your organization name.	Example Corporation
Organizational Unit Name	Additional organizational information, if any.	Example Dept
Common Name	This value must exactly match the web address that you expect users to enter into a browser. Usually, this means a domain name with a prefixed hostname or alias in the form www.example.com . In testing with a self-signed certificate and no DNS resolution, the common name may consist of the hostname alone. CAs also offer more expensive certificates that accept wild-card names such as *.example.com .	www.example.com
Email Address	The server administrator's email address.	someone@example.com

Finally, OpenSSL prompts you for an optional challenge password. This password applies only to the CSR and to transactions between you and your CA, so follow the CA's recommendations about this and the other optional field, optional company name. The CSR challenge password has no effect on server operation.

The resulting file **csr.pem** contains your public key, your digital signature of your public key, and the metadata that you entered.

4. Submit the CSR to a CA. This usually consists of opening your CSR file in a text editor and copying the contents into a web form. At this time, you may be asked to supply one or more subject alternate names (SANs) to be placed on the certificate. If **www.example.com** is the common name, then **example.com** would be a good SAN, and vice versa. A visitor to your site

entering either of these names would see an error-free connection. If your CA web form allows it, include the common name in the list of SANs. Some CAs include it automatically.

After your request has been approved, you receive a new host certificate signed by the CA. You might also be instructed to download an *intermediate certificate* file that contains additional certificates needed to complete the CA's chain of trust.

Note

Your CA might send you files in multiple formats intended for various purposes. For this tutorial, you should only use a certificate file in PEM format, which is usually (but not always) marked with a `.pem` or `.crt` file extension. If you are uncertain which file to use, open the files with a text editor and find the one containing one or more blocks beginning with the following line.

```
- - - - -BEGIN CERTIFICATE - - - - -
```

The file should also end with the following line.

```
- - - - -END CERTIFICATE - - - - -
```

You can also test the file at the command line as shown in the following.

```
[ec2-user certs]$ openssl x509 -in certificate.crt -text
```

Verify that these lines appear in the file. Do not use files ending with `.p7b`, `.p7c`, or similar file extensions.

5. Place the new CA-signed certificate and any intermediate certificates in the `/etc/pki/tls/certs` directory.

Note

There are several ways to upload your new certificate to your EC2 instance, but the most straightforward and informative way is to open a text editor (for example, `vi`, `nano`, or `notepad`) on both your local computer and your instance, and then copy and paste the file contents between them. You need root `[sudo]` permissions when performing these operations on the EC2 instance. This way, you can see immediately

if there are any permission or path problems. Be careful, however, not to add any additional lines while copying the contents, or to change them in any way.

From inside the `/etc/pki/tls/certs` directory, check that the file ownership, group, and permission settings match the highly restrictive AL2023 defaults (owner=root, group=root, read/write for owner only). The following example shows the commands to use.

```
[ec2-user certs]$ sudo chown root:root custom.crt
[ec2-user certs]$ sudo chmod 600 custom.crt
[ec2-user certs]$ ls -al custom.crt
```

These commands should yield the following result.

```
-rw----- root root custom.crt
```

The permissions for the intermediate certificate file are less stringent (owner=root, group=root, owner can write, group can read, world can read). The following example shows the commands to use.

```
[ec2-user certs]$ sudo chown root:root intermediate.crt
[ec2-user certs]$ sudo chmod 644 intermediate.crt
[ec2-user certs]$ ls -al intermediate.crt
```

These commands should yield the following result.

```
-rw-r--r-- root root intermediate.crt
```

6. Place the private key that you used to create the CSR in the `/etc/pki/tls/private/` directory.

Note

There are several ways to upload your custom key to your EC2 instance, but the most straightforward and informative way is to open a text editor (for example, `vi`, `nano`, or `notepad`) on both your local computer and your instance, and then copy and paste the file contents between them. You need root [sudo] permissions when performing these operations on the EC2 instance. This way, you can see immediately if there are

any permission or path problems. Be careful, however, not to add any additional lines while copying the contents, or to change them in any way.

From inside the `/etc/pki/tls/private` directory, use the following commands to verify that the file ownership, group, and permission settings match the highly restrictive AL2023 defaults (owner=root, group=root, read/write for owner only).

```
[ec2-user private]$ sudo chown root:root custom.key
[ec2-user private]$ sudo chmod 600 custom.key
[ec2-user private]$ ls -al custom.key
```

These commands should yield the following result.

```
-rw----- root root custom.key
```

7. Edit `/etc/httpd/conf.d/ssl.conf` to reflect your new certificate and key files.

- a. Provide the path and file name of the CA-signed host certificate in Apache's `SSLCertificateFile` directive:

```
SSLCertificateFile /etc/pki/tls/certs/custom.crt
```

- b. If you received an intermediate certificate file (`intermediate.crt` in this example), provide its path and file name using Apache's `SSLCACertificateFile` directive:

```
SSLCACertificateFile /etc/pki/tls/certs/intermediate.crt
```

 **Note**

Some CAs combine the host certificate and the intermediate certificates in a single file, making the `SSLCACertificateFile` directive unnecessary. Consult the instructions provided by your CA.

- c. Provide the path and file name of the private key (`custom.key` in this example) in Apache's `SSLCertificateKeyFile` directive:

```
SSLCertificateKeyFile /etc/pki/tls/private/custom.key
```

8. Save `/etc/httpd/conf.d/ssl.conf` and restart Apache.

```
[ec2-user ~]$ sudo systemctl restart httpd
```

9. Test your server by entering your domain name into a browser URL bar with the prefix `https://`. Your browser should load the test page over HTTPS without generating errors.

Step 3: Test and harden the security configuration

After your TLS is operational and exposed to the public, you should test how secure it really is. This is easy to do using online services such as [Qualys SSL Labs](#), which performs a free and thorough analysis of your security setup. Based on the results, you may decide to harden the default security configuration by controlling which protocols you accept, which ciphers you prefer, and which you exclude. For more information, see [how Qualys formulates its scores](#).

Important

Real-world testing is crucial to the security of your server. Small configuration errors may lead to serious security breaches and loss of data. Because recommended security practices change constantly in response to research and emerging threats, periodic security audits are essential to good server administration.

On the [Qualys SSL Labs](#) site, enter the fully qualified domain name of your server, in the form **www.example.com**. After about two minutes, you receive a grade (from A to F) for your site and a detailed breakdown of the findings. The following table summarizes the report for a domain with settings identical to the default Apache configuration on AL2023, and with a default Certbot certificate.

Overall rating	B
Certificate	100%
Protocol support	95%
Key exchange	70%
Cipher strength	90%

Though the overview shows that the configuration is mostly sound, the detailed report flags several potential problems, listed here in order of severity:

x The RC4 cipher is supported for use by certain older browsers. A cipher is the mathematical core of an encryption algorithm. RC4, a fast cipher used to encrypt TLS data-streams, is known to have several [serious weaknesses](#). Unless you have very good reasons to support legacy browsers, you should disable this.

x Old TLS versions are supported. The configuration supports TLS 1.0 (already deprecated) and TLS 1.1 (on a path to deprecation). Only TLS 1.2 has been recommended since 2018.

x Forward secrecy is not fully supported. [Forward secrecy](#) is a feature of algorithms that encrypt using temporary (ephemeral) session keys derived from the private key. This means in practice that attackers cannot decrypt HTTPS data even if they possess a web server's long-term private key.

To correct and future-proof the TLS configuration

1. Open the configuration file `/etc/httpd/conf.d/ssl.conf` in a text editor and comment out the following line by entering `"#"` at the beginning of the line.

```
#SSLProtocol all -SSLv3
```

2. Add the following directive:

```
#SSLProtocol all -SSLv3  
SSLProtocol -SSLv2 -SSLv3 -TLSv1 -TLSv1.1 +TLSv1.2
```

This directive explicitly disables SSL versions 2 and 3, as well as TLS versions 1.0 and 1.1. The server now refuses to accept encrypted connections with clients using anything except TLS 1.2. The verbose wording in the directive conveys more clearly, to a human reader, what the server is configured to do.

Note

Disabling TLS versions 1.0 and 1.1 in this manner blocks a small percentage of outdated web browsers from accessing your site.

To modify the list of allowed ciphers

1. In the configuration file `/etc/httpd/conf.d/ssl.conf`, find the section with the **SSLCipherSuite** directive and comment out the existing line by entering `"#"` at the beginning of the line.

```
#SSLCipherSuite HIGH:MEDIUM:!aNULL:!MD5
```

2. Specify explicit cipher suites and a cipher order that prioritizes forward secrecy and avoids insecure ciphers. The **SSLCipherSuite** directive used here is based on output from the [Mozilla SSL Configuration Generator](#), which tailors a TLS configuration to the specific software running on your server. First determine your Apache and OpenSSL versions by using the output from the following commands.

```
[ec2-user ~]$ yum list installed | grep httpd
```

```
[ec2-user ~]$ yum list installed | grep openssl
```

For example, if the returned information is Apache 2.4.34 and OpenSSL 1.0.2, we enter this into the generator. If you choose the "modern" compatibility model, this creates an **SSLCipherSuite** directive that aggressively enforces security but still works for most browsers. If your software doesn't support the modern configuration, you can update your software or choose the "intermediate" configuration instead.

```
SSLCipherSuite ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-  
ECDSA-CHACHA20-POLY1305:  
ECDHE-RSA-CHACHA20-POLY1305:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-  
SHA256:  
ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES128-SHA256:ECDHE-  
RSA-AES128-SHA256
```

The selected ciphers have *ECDHE* in their names, an abbreviation for *Elliptic Curve Diffie-Hellman Ephemeral*. The term *ephemeral* indicates forward secrecy. As a by-product, these ciphers do not support RC4.

We recommend that you use an explicit list of ciphers instead of relying on defaults or terse directives whose content isn't visible.

Copy the generated directive into `/etc/httpd/conf.d/ssl.conf`.

Note

Though shown here on several lines for readability, the directive must be on a single line when copied to `/etc/httpd/conf.d/ssl.conf`, with only a colon (no spaces) between cipher names.

- Finally, uncomment the following line by removing the `"#"` at the beginning of the line.

```
#SSLHonorCipherOrder on
```

This directive forces the server to prefer high-ranking ciphers, including (in this case) those that support forward secrecy. With this directive turned on, the server tries to establish a strong secure connection before falling back to allowed ciphers with lesser security.

After completing both of these procedures, save the changes to `/etc/httpd/conf.d/ssl.conf` and restart Apache.

If you test the domain again on [Qualys SSL Labs](#), you should see that the RC4 vulnerability and other warnings are gone and the summary looks something like the following.

Overall rating	A
Certificate	100%
Protocol support	100%
Key exchange	90%
Cipher strength	90%

Each update to OpenSSL introduces new ciphers and removes support for old ones. Keep your EC2 AL2023 instance up-to-date, watch for security announcements from [OpenSSL](#), and be alert to reports of new security exploits in the technical press.

Troubleshoot

- My Apache webserver doesn't start unless I enter a password**

This is expected behavior if you installed an encrypted, password-protected, private server key.

You can remove the encryption and password requirement from the key. Assuming that you have a private encrypted RSA key called `custom.key` in the default directory, and that the password on it is **abcde12345**, run the following commands on your EC2 instance to generate an unencrypted version of the key.

```
[ec2-user ~]$ cd /etc/pki/tls/private/
[ec2-user private]$ sudo cp custom.key custom.key.bak
[ec2-user private]$ sudo openssl rsa -in custom.key -passin pass:abcde12345 -out
custom.key.nocrypt
[ec2-user private]$ sudo mv custom.key.nocrypt custom.key
[ec2-user private]$ sudo chown root:root custom.key
[ec2-user private]$ sudo chmod 600 custom.key
[ec2-user private]$ sudo systemctl restart httpd
```

Apache should now start without prompting you for a password.

- **I get errors when I run `sudo dnf install -y mod_ssl`.**

When you are installing the required packages for SSL, you may see errors similar to the following.

```
Error: httpd24-tools conflicts with httpd-tools-2.2.34-1.16.amzn1.x86_64
Error: httpd24 conflicts with httpd-2.2.34-1.16.amzn1.x86_64
```

This typically means that your EC2 instance is not running AL2023. This tutorial only supports instances freshly created from an official AL2023 AMI.

Tutorial: Host a WordPress blog on AL2023

The following procedures will help you install, configure, and secure a WordPress blog on your AL2023 instance. This tutorial is a good introduction to using Amazon EC2 in that you have full control over a web server that hosts your WordPress blog, which is not typical with a traditional hosting service.

You are responsible for updating the software packages and maintaining security patches for your server. For a more automated WordPress installation that does not require direct interaction with the web server configuration, the CloudFormation service provides a WordPress template that can

also get you started quickly. For more information, see [Get started](#) in the *AWS CloudFormation User Guide*. If you need a high-availability solution with a decoupled database, see [Deploying a high-availability WordPress website](#) in the *AWS Elastic Beanstalk Developer Guide*.

Important

These procedures are intended for use with AL2023. For information about other distributions, see their specific documentation. Many steps in this tutorial do not work on Ubuntu instances. For help installing WordPress on an Ubuntu instance, see [WordPress](#) in the Ubuntu documentation. You can also use [CodeDeploy](#) to accomplish this task on Amazon Linux, macOS, or Unix systems.

Topics

- [Prerequisites](#)
- [Install WordPress](#)
- [Next steps](#)
- [Help! My public DNS name changed and now my blog is broken](#)

Prerequisites

We strongly recommend that you associate an Elastic IP address (EIP) to the instance you are using to host a WordPress blog. This prevents the public DNS address for your instance from changing and breaking your installation. If you own a domain name and you want to use it for your blog, you can update the DNS record for the domain name to point to your EIP address (for help with this, contact your domain name registrar). You can have one EIP address associated with a running instance at no charge. For more information, see [Elastic IP addresses](#) in the *Amazon EC2 User Guide*. The [Tutorial: Install a LAMP server on AL2023](#) tutorial has steps for configuring a security group to allow HTTP and HTTPS traffic, as well as several steps to ensure that file permissions are set properly for your web server. For information about adding rules to your security group, see [Add rules to a security group](#).

If you don't already have a domain name for your blog, you can register a domain name with Route 53 and associate your instance's EIP address with your domain name. For more information, see [Registering domain names using Amazon Route 53](#) in the *Amazon Route 53 Developer Guide*.

Install WordPress

Connect to your instance, and download the WordPress installation package. For more information about connecting to your instance, see [Connecting to AL2023 instances](#).

1. Download and install these packages using the following command.

```
dnf install wget php-mysqlnd httpd php-fpm php-mysqli mariadb105-server php-json  
php php-devel -y
```

2. You may notice a warning displayed with similar verbiage in the output (the versions may vary over time):

```
WARNING:  
  A newer release of "Amazon Linux" is available.  
  
  Available Versions:  
  
dnf upgrade --releasever=2023.0.20230202  
  
  Release notes:  
  https://aws.amazon.com  
  
Version 2023.0.20230204:  
  Run the following command to update to 2023.0.20230204:  
  
  dnf upgrade --releasever=2023.0.20230204 ... etc
```

As a best-practice we recommend keeping the OS as up-to-date as possible, but you may want to iterate through each version to ensure there are no conflicts in your environment. **If installation of the preceding packages noted in step 1 fail, you may need to update to one of the newer releases listed, and retry.**

3. Download the latest WordPress installation package with the **wget** command. The following command should always download the latest release.

```
[ec2-user ~]$ wget https://wordpress.org/latest.tar.gz
```

4. Unzip and unarchive the installation package. The installation folder is unzipped to a folder called `wordpress`.

```
[ec2-user ~]$ tar -xzf latest.tar.gz
```

To create a database user and database for your WordPress installation

Your WordPress installation needs to store information, such as blog posts and user comments, in a database. This procedure helps you create your blog's database and a user that is authorized to read and save information to it.

1. Start the database and web server.

```
[ec2-user ~]$ sudo systemctl start mariadb httpd
```

2. Log in to the database server as the root user. Enter your database root password when prompted; this may be different than your root system password, or it might even be empty if you have not secured your database server.

If you have not secured your database server yet, it is important that you do so. For more information, see [Step 3: Secure the database server](#) (AL2023).

```
[ec2-user ~]$ mysql -u root -p
```

3. Create a user and password for your MySQL database. Your WordPress installation uses these values to communicate with your MySQL database. Enter the following command, substituting a unique user name and password.

```
CREATE USER 'wordpress-user'@'localhost' IDENTIFIED BY 'your_strong_password';
```

Make sure that you create a strong password for your user. Do not use the single quote character (') in your password, because this will break the preceding command. Do not reuse an existing password, and make sure to store this password in a safe place.

4. Create your database. Give your database a descriptive, meaningful name, such as `wordpress-db`.

Note

The punctuation marks surrounding the database name in the command below are called backticks. The backtick (`) key is usually located above the Tab key on

a standard keyboard. Backticks are not always required, but they allow you to use otherwise illegal characters, such as hyphens, in database names.

```
CREATE DATABASE `wordpress-db`;
```

5. Grant full privileges for your database to the WordPress user that you created earlier.

```
GRANT ALL PRIVILEGES ON `wordpress-db`.* TO "wordpress-user"@"localhost";
```

6. Flush the database privileges to pick up all of your changes.

```
FLUSH PRIVILEGES;
```

7. Exit the mysql client.

```
exit
```

To create and edit the wp-config.php file

The WordPress installation folder contains a sample configuration file called `wp-config-sample.php`. In this procedure, you copy this file and edit it to fit your specific configuration.

1. Copy the `wp-config-sample.php` file to a file called `wp-config.php`. This creates a new configuration file and keeps the original sample file intact as a backup.

```
[ec2-user ~]$ cp wordpress/wp-config-sample.php wordpress/wp-config.php
```

2. Edit the `wp-config.php` file with your favorite text editor (such as **nano** or **vim**) and enter values for your installation. If you do not have a favorite text editor, nano is suitable for beginners.

```
[ec2-user ~]$ nano wordpress/wp-config.php
```

- a. Find the line that defines `DB_NAME` and change `database_name_here` to the database name that you created in [Step 4](#) of [To create a database user and database for your WordPress installation](#).

```
define('DB_NAME', 'wordpress-db');
```

- b. Find the line that defines DB_USER and change username_here to the database user that you created in [Step 3](#) of [To create a database user and database for your WordPress installation](#).

```
define('DB_USER', 'wordpress-user');
```

- c. Find the line that defines DB_PASSWORD and change password_here to the strong password that you created in [Step 3](#) of [To create a database user and database for your WordPress installation](#).

```
define('DB_PASSWORD', 'your_strong_password');
```

- d. Find the section called Authentication Unique Keys and Salts. These KEY and SALT values provide a layer of encryption to the browser cookies that WordPress users store on their local machines. Basically, adding long, random values here makes your site more secure. Visit <https://api.wordpress.org/secret-key/1.1/salt/> to randomly generate a set of key values that you can copy and paste into your wp-config.php file. To paste text into a PuTTY terminal, place the cursor where you want to paste the text and right-click your mouse inside the PuTTY terminal.

For more information about security keys, go to <https://wordpress.org/support/article/editing-wp-config-php/#security-keys>.

Note

The values below are for example purposes only; do not use these values for your installation.

```
define('AUTH_KEY',          ' #U$$+[RXN8:b^-L 0(WU_+ c+WFkI~c]o]-bHw+)/
Aj[wTwSiZ<Qb[mghEXcRh-');
define('SECURE_AUTH_KEY',  'Zsz._P=l/|y.Lq)XjlkW51y5NJ76E6EJ.AV0pCKZZB,*~*r ?
6OP$eJT@;+(ndLg');
define('LOGGED_IN_KEY',    'ju}qwre3V*+8f_z0Wf?{LLGsQ]Ye@2Jh^,8x>)Y |;(^[Iw]Pi
+LG#A4R?7N`YB3');
```

```
define('NONCE_KEY',          'P(g62HeZxEes|LnI^i=H,[XwK9I&[2s|: ?0N}VJM%?;v2v]v+;
+^9eXUahg@::Cj');
define('AUTH_SALT',          'C$DpB4Hj[JK: ?{qL`sRVa: { :7yShy(9A@5wg+`JJVb1fk%-
Bx*M4(qc[Qg%JT!h');
define('SECURE_AUTH_SALT',  'd!uRu#}+q#{f$Z?Z9uFPG.$ {+S{n~1M&%@~gL>U>NV<zpD-@2-
Es7Q10-bp28EKv');
define('LOGGED_IN_SALT',    ';j{00P*owZf)kVD+FVLn-~ >.|Y%Ug4#I^*LVd9QeZ^&XmK|
e(76miC+&W&+^0P/');
define('NONCE_SALT',        '-97r*V/cgxLmp?Zy4zUU4r99QQ_rGs2LTd%P;|
_e1tS)8_B/, .6[=UK<J_y9?JWG');
```

- e. Save the file and exit your text editor.

To install your WordPress files under the Apache document root

- Now that you've unzipped the installation folder, created a MySQL database and user, and customized the WordPress configuration file, you are ready to copy your installation files to your web server document root so you can run the installation script that completes your installation. The location of these files depends on whether you want your WordPress blog to be available at the actual root of your web server (for example, *my.public.dns.amazonaws.com*) or in a subdirectory or folder under the root (for example, *my.public.dns.amazonaws.com/blog*).
- If you want WordPress to run at your document root, copy the contents of the wordpress installation directory (but not the directory itself) as follows:

```
[ec2-user ~]$ cp -r wordpress/* /var/www/html/
```

- If you want WordPress to run in an alternative directory under the document root, first create that directory, and then copy the files to it. In this example, WordPress will run from the directory `blog`:

```
[ec2-user ~]$ mkdir /var/www/html/blog
[ec2-user ~]$ cp -r wordpress/* /var/www/html/blog/
```

Important

For security purposes, if you are not moving on to the next procedure immediately, stop the Apache web server (`httpd`) now. After you move your installation under the Apache

document root, the WordPress installation script is unprotected and an attacker could gain access to your blog if the Apache web server were running. To stop the Apache web server, enter the command **sudo service httpd stop**. If you are moving on to the next procedure, you do not need to stop the Apache web server.

To allow WordPress to use permalinks

WordPress permalinks need to use Apache `.htaccess` files to work properly, but this is not enabled by default on Amazon Linux. Use this procedure to allow all overrides in the Apache document root.

1. Open the `httpd.conf` file with your favorite text editor (such as **nano** or **vim**). If you do not have a favorite text editor, nano is suitable for beginners.

```
[ec2-user ~]$ sudo vim /etc/httpd/conf/httpd.conf
```

2. Find the section that starts with `<Directory "/var/www/html">`.

```
<Directory "/var/www/html">
#
# Possible values for the Options directive are "None", "All",
# or any combination of:
#   Indexes Includes FollowSymLinks SymLinksifOwnerMatch ExecCGI MultiViews
#
# Note that "MultiViews" must be named *explicitly* --- "Options All"
# doesn't give it to you.
#
# The Options directive is both complicated and important. Please see
# http://httpd.apache.org/docs/2.4/mod/core.html#options
# for more information.
#
Options Indexes FollowSymLinks

#
# AllowOverride controls what directives may be placed in .htaccess files.
# It can be "All", "None", or any combination of the keywords:
#   Options FileInfo AuthConfig Limit
#
AllowOverride None

#
```

```
# Controls who can get stuff from this server.  
#  
Require all granted  
</Directory>
```

3. Change the AllowOverride None line in the above section to read AllowOverride **All**.

Note

There are multiple AllowOverride lines in this file; be sure you change the line in the `<Directory "/var/www/html">` section.

```
AllowOverride All
```

4. Save the file and exit your text editor.

To install the PHP graphics drawing library on AL2023

The GD library for PHP enables you to modify images. Install this library if you need to crop the header image for your blog. The version of phpMyAdmin that you install might require a specific minimum version of this library (for example, version 8.1).

Use the following command to install the PHP graphics drawing library on AL2023. For example, if you installed php8.1 from source as part of installing the LAMP stack, this command installs version 8.1 of the PHP graphics drawing library.

```
[ec2-user ~]$ sudo dnf install php-gd
```

To verify the installed version, use the following command:

```
[ec2-user ~]$ sudo dnf list installed | grep php-gd
```

The following is example output:

php-gd.x86_64	8.1.30-1.amzn2	@amazonlinux
---------------	----------------	--------------

To install the PHP graphics drawing library on the Amazon Linux AMI

The GD library for PHP enables you to modify images. Install this library if you need to crop the header image for your blog. The version of phpMyAdmin that you install might require a specific minimum version of this library (for example, version 8.1).

To verify which versions are available, use the following command:

```
[ec2-user ~]$ dnf list | grep php
```

The following is example lines from the output for the PHP graphics drawing library (version 8.1):

php8.1.aarch64		8.1.7-1.amzn2023.0.1
	@amazonlinux	
php8.1-cli.aarch64		8.1.7-1.amzn2023.0.1
	@amazonlinux	
php8.1-common.aarch64		8.1.7-1.amzn2023.0.1
	@amazonlinux	
php8.1-devel.aarch64		8.1.7-1.amzn2023.0.1
	@amazonlinux	
php8.1-fpm.aarch64		8.1.7-1.amzn2023.0.1
	@amazonlinux	
php8.1-gd.aarch64		8.1.7-1.amzn2023.0.1
	@amazonlinux	

Use the following command to install a specific version of the PHP graphics drawing library (for example, version php8.1) on the Amazon Linux AMI:

```
[ec2-user ~]$ sudo dnf install -y php8.1-gd
```

To fix file permissions for the Apache web server

Some of the available features in WordPress require write access to the Apache document root (such as uploading media through the Administration screens). If you have not already done so, apply the following group memberships and permissions (as described in greater detail in the [LAMP web server tutorial](#)).

1. Grant file ownership of `/var/www` and its contents to the apache user.

```
[ec2-user ~]$ sudo chown -R apache /var/www
```

2. Grant group ownership of `/var/www` and its contents to the apache group.

```
[ec2-user ~]$ sudo chgrp -R apache /var/www
```

3. Change the directory permissions of `/var/www` and its subdirectories to add group write permissions and to set the group ID on future subdirectories.

```
[ec2-user ~]$ sudo chmod 2775 /var/www  
[ec2-user ~]$ find /var/www -type d -exec sudo chmod 2775 {} \;
```

4. Recursively change the file permissions of `/var/www` and its subdirectories.

```
[ec2-user ~]$ find /var/www -type f -exec sudo chmod 0644 {} \;
```

Note

If you intend to also use WordPress as an FTP server, you'll need more permissive Group settings here. Please review the recommended [steps and security settings in WordPress](#) to accomplish this.

5. Restart the Apache web server to pick up the new group and permissions.

```
[ec2-user ~]$ sudo systemctl restart httpd
```

To run the WordPress installation script with AL2023

You are ready to install WordPress. The commands that you use depend on the operating system. The commands in this procedure are for use with AL2023. Use the procedure that follows this one with AL2023 AMI.

1. Use the **systemctl** command to ensure that the `httpd` and database services start at every system boot.

```
[ec2-user ~]$ sudo systemctl enable httpd && sudo systemctl enable mariadb
```

2. Verify that the database server is running.

```
[ec2-user ~]$ sudo systemctl status mariadb
```

If the database service is not running, start it.

```
[ec2-user ~]$ sudo systemctl start mariadb
```

3. Verify that your Apache web server (httpd) is running.

```
[ec2-user ~]$ sudo systemctl status httpd
```

If the httpd service is not running, start it.

```
[ec2-user ~]$ sudo systemctl start httpd
```

4. In a web browser, type the URL of your WordPress blog (either the public DNS address for your instance, or that address followed by the blog folder). You should see the WordPress installation script. Provide the information required by the WordPress installation. Choose **Install WordPress** to complete the installation. For more information, see [Step 5: Run the Install Script](#) on the WordPress website.

To run the WordPress installation script with AL2023 AMI

1. Use the **chkconfig** command to ensure that the httpd and database services start at every system boot.

```
[ec2-user ~]$ sudo chkconfig httpd on && sudo chkconfig mariadb on
```

2. Verify that the database server is running.

```
[ec2-user ~]$ sudo service mariadb status
```

If the database service is not running, start it.

```
[ec2-user ~]$ sudo service mariadb start
```

3. Verify that your Apache web server (httpd) is running.

```
[ec2-user ~]$ sudo service httpd status
```

If the httpd service is not running, start it.

```
[ec2-user ~]$ sudo service httpd start
```

4. In a web browser, type the URL of your WordPress blog (either the public DNS address for your instance, or that address followed by the `blog` folder). You should see the WordPress installation script. Provide the information required by the WordPress installation. Choose **Install WordPress** to complete the installation. For more information, see [Step 5: Run the Install Script](#) on the WordPress website.

Next steps

After you have tested your WordPress blog, consider updating its configuration.

Use a custom domain name

If you have a domain name associated with your EC2 instance's EIP address, you can configure your blog to use that name instead of the EC2 public DNS address. For more information, see [Changing The Site URL](#) on the WordPress website.

Configure your blog

You can configure your blog to use different [themes](#) and [plugins](#) to offer a more personalized experience for your readers. However, sometimes the installation process can backfire, causing you to lose your entire blog. We strongly recommend that you create a backup Amazon Machine Image (AMI) of your instance before attempting to install any themes or plugins so you can restore your blog if anything goes wrong during installation. For more information, see [Create your own AMI](#) in the *Amazon EC2 User Guide*.

Increase capacity

If your WordPress blog becomes popular and you need more compute power or storage, consider the following steps:

- Expand the storage space on your instance. For more information, see [Amazon EBS Elastic Volumes](#).
- Move your MySQL database to [Amazon RDS](#) to take advantage of the service's ability to scale easily.

Improve network performance of your internet traffic

If you expect your blog to drive traffic from users located around the world, consider [AWS Global Accelerator](#). Global Accelerator helps you achieve lower latency by improving internet traffic performance between your users' client devices and your WordPress application running on AWS. Global Accelerator uses the [AWS global network](#) to direct traffic to a healthy application endpoint in the AWS Region that is closest to the client.

Learn more about WordPress

The following links contain more information about WordPress.

- For information about WordPress, see the WordPress Codex help documentation at [Codex](#).
- For more information about troubleshooting your installation, go to [Common installation problems](#).
- For information about making your WordPress blog more secure, go to [Hardening WordPress](#).
- For information about keeping your WordPress blog up-to-date, go to [Updating WordPress](#).

Help! My public DNS name changed and now my blog is broken

Your WordPress installation is automatically configured using the public DNS address for your EC2 instance. If you stop and restart the instance, the public DNS address changes (unless it is associated with an Elastic IP address) and your blog will not work anymore because it references resources at an address that no longer exists (or is assigned to another EC2 instance). A more detailed description of the problem and several possible solutions are outlined in <https://wordpress.org/support/article/changing-the-site-url/>.

If this has happened to your WordPress installation, you may be able to recover your blog with the procedure below, which uses the **wp-cli** command line interface for WordPress.

To change your WordPress site URL with the wp-cli

1. Connect to your EC2 instance with SSH.
2. Note the old site URL and the new site URL for your instance. The old site URL is likely the public DNS name for your EC2 instance when you installed WordPress. The new site URL is the current public DNS name for your EC2 instance. If you are not sure of your old site URL, you can use **curl** to find it with the following command.

```
[ec2-user ~]$ curl localhost | grep wp-content
```

You should see references to your old public DNS name in the output, which will look like this (old site URL in red):

```
<script type='text/javascript' src='http://ec2-52-8-139-223.us-west-1.compute.amazonaws.com/wp-content/themes/twentyfifteen/js/functions.js?ver=20150330'></script>
```

3. Download the **wp-cli** with the following command.

```
[ec2-user ~]$ curl -O https://raw.githubusercontent.com/wp-cli/builds/gh-pages/phar/wp-cli.phar
```

4. Search and replace the old site URL in your WordPress installation with the following command. Substitute the old and new site URLs for your EC2 instance and the path to your WordPress installation (usually `/var/www/html` or `/var/www/html/blog`).

```
[ec2-user ~]$ php wp-cli.phar search-replace 'old_site_url' 'new_site_url' --path=/path/to/wordpress/installation --skip-columns=guid
```

5. In a web browser, enter the new site URL of your WordPress blog to verify that the site is working properly again. If it is not, see [Changing the site URL](#) and [Common installation problems](#) for more information.

Tutorial: Redis 6 to Valkey Transition on AL2023

The following documentation describes key aspects of the transition from Redis 6 to Valkey on AL2023.

Support timeline for Redis 6

Redis 6 reaches its End of Life (EOL) on January 31, 2026. After this date, Redis 6 will no longer receive updates or security patches from the Redis project. We strongly recommend users migrate to Valkey before January 2026 to ensure continued support and security updates.

For more information on Redis version support timelines, see [Redis End-Of-Life Schedule](#) documentation.

Introduction to Valkey

Valkey is an open-source fork of Redis 7, maintained by The Linux Foundation. It's fully compatible with Redis Open Source Software (OSS) versions 2.x through 7.2.x. Valkey maintains the familiar Redis API and functionality while offering several enhancements:

- Enhanced performance through multi-threading.
- Improved memory efficiency, especially in cluster mode.
- Dual-channel replication for better data consistency.

Migration plan and timeline

Users are strongly encouraged to migrate from Redis 6 to Valkey before January 31, 2026, when Redis 6 reaches its End of Life (EOL). This migration requires manual intervention and is not automatic.

Amazon Linux recommends this migration to ensure continued functionality, support, and security updates for your Redis-dependent applications.

Migration options and steps

We propose three migration paths to Valkey based on your deployment requirements and operational needs.

Option 1: New Instance Installation

For new deployments or when data migration is not needed:

1. Install Valkey:

```
[ec2-user ~]$ sudo dnf install valkey
```

2. Start Valkey:

```
[ec2-user ~]$ sudo systemctl start valkey
```

3. (Optional) Enable Valkey on boot:

```
[ec2-user ~]$ sudo systemctl enable valkey
```

4. Verify the installation:

```
[ec2-user ~]$ valkey-cli info server  
[ec2-user ~]$ valkey-cli ping
```

Option 2: In-Place Replacement

For existing instances where data persistence is not required:

1. Stop Redis 6:

```
[ec2-user ~]$ sudo systemctl stop redis6
```

2. Install Valkey:

```
[ec2-user ~]$ sudo dnf install valkey
```

3. (Optional) Use Redis 6 configuration in Valkey:

```
[ec2-user ~]$ sudo cp /etc/redis6/redis6.conf /etc/valkey/valkey.conf  
[ec2-user ~]$ sudo cp /etc/valkey/valkey.conf /etc/valkey/valkey.conf.backup  
[ec2-user ~]$ sudo chown valkey:root /etc/valkey/valkey.conf  
[ec2-user ~]$ sudo sed -i 's|^dir\s.*|dir /var/lib/valkey|g' /etc/valkey/  
valkey.conf
```

4. (Optional) Use Redis 6 sentinel configuration file in Valkey:

```
[ec2-user ~]$ sudo cp /etc/redis6/sentinel.conf /etc/valkey/sentinel.conf  
[ec2-user ~]$ sudo chown valkey:root /etc/valkey/sentinel.conf
```

5. Start Valkey:

```
[ec2-user ~]$ sudo systemctl start valkey
```

6. (Optional) Enable Valkey on boot:

```
[ec2-user ~]$ sudo systemctl enable valkey
```

7. Verify Valkey installation:

```
[ec2-user ~]$ valkey-cli info server
```

```
[ec2-user ~]$ valkey-cli ping
```

8. Remove Redis 6:

```
[ec2-user ~]$ sudo dnf remove redis6
```

Option 3: Data Migration

This option allows you to run both Redis 6 and Valkey concurrently.

1. Install Valkey without removing Redis 6:

```
[ec2-user ~]$ sudo dnf install valkey
```

2. (Optional) Use Redis 6 configuration in Valkey:

```
[ec2-user ~]$ sudo cp /etc/redis6/redis6.conf /etc/valkey/valkey.conf
[ec2-user ~]$ sudo cp /etc/valkey/valkey.conf /etc/valkey/valkey.conf.backup
[ec2-user ~]$ sudo chown valkey:root /etc/valkey/valkey.conf
[ec2-user ~]$ sudo sed -i 's|^dir\s.*|dir /var/lib/valkey|g' /etc/valkey/
valkey.conf
```

3. (Optional) Use Redis 6 sentinel configuration file in Valkey:

```
[ec2-user ~]$ sudo cp /etc/redis6/sentinel.conf /etc/valkey/sentinel.conf
[ec2-user ~]$ sudo chown valkey:root /etc/valkey/sentinel.conf
```

4. Modify Valkey configuration:

Edit `/etc/valkey/valkey.conf` and set the `'port'` directive to a different value (for example, 6380) to avoid conflicts with Redis 6.

5. Start Valkey:

```
[ec2-user ~]$ sudo systemctl start valkey
```

6. (Optional) Enable Valkey on boot:

```
[ec2-user ~]$ sudo systemctl enable valkey
```

7. Verify Valkey installation:

```
[ec2-user ~]$ valkey-cli -p port info server  
[ec2-user ~]$ valkey-cli -p port ping
```

Note

Replace **port** with the configured port number.

8. Migrate data:

You can now migrate data from Redis 6 to Valkey using replication or manual data transfer methods.

9. Update application configurations:

Gradually update your applications to use the Valkey port.

10. Remove Redis 6:

Once all data and applications have been migrated, you can stop and remove Redis 6.

```
[ec2-user ~]$ sudo systemctl stop redis6  
[ec2-user ~]$ sudo dnf remove redis6
```

Note

It is strongly recommended to validate the migration process in a test environment before implementing changes in production systems.

Related topics

For more information about Valkey:

- Valkey: <https://valkey.io/>
- Valkey migration: <https://valkey.io/topics/migration/>

Tutorial: Install the GNOME desktop environment on AL2023

The [GNOME desktop environment](#) is available as an optional graphical user interface for AL2023 as of release 2023.7 or later.

The following procedures help you install the GNOME desktop environment on your AL2023 instance. You can use this graphical interface to interact with your Linux system using a familiar desktop environment instead of just the command line interface.

Contents

- [Prerequisites](#)
- [Installation](#)
- [Related topics](#)

Prerequisites

- The desktop environment requires at least 2.4 GB of memory. Therefore, an instance of type `t2.medium` or better is recommended to ensure adequate performance. Examples of instance types with insufficient memory include `t2.nano`, `t2.micro`, and `t2.small`. This restriction also applies to `t3` and `t4` instances of this size, as well as any other instance type that doesn't meet the memory requirement.
- This tutorial assumes that you have already launched an instance using AL2023 running release 2023.7 or later. For more information, see the [AL2023 on Amazon EC2](#) and [Updating AL2023](#) pages.

Installation

- Install the GNOME desktop environment and related packages.

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

Note

To access the graphical desktop environment, you'll 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.

Related topics

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

- [What Is Amazon DCV?](#) in the *Amazon DCV Administrator Guide*
- [Tutorial: Configure TigerVNC server on AL2023](#)

Tutorial: Configure TigerVNC server on AL2023

The following procedures help you set up VNC server on your AL2023 instance. VNC allows you to remotely access and interact with the graphical desktop environment over a secure network connection.

Contents

- [Prerequisites](#)
- [Step 1: Installation](#)
- [Step 2: Configuration](#)
- [Step 3: Connect using a VNC client](#)
- [\(Optional\) Start service at boot](#)
- [\(Optional\) Disable the idle lockscreen](#)
- [Related topics](#)

Prerequisites

- This tutorial assumes you have already installed the GNOME desktop environment on your AL2023 instance. For more information, see the [Tutorial: Install the GNOME desktop environment on AL2023](#) page.

- This tutorial uses SSH port forwarding to access the VNC 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 a VNC client. You must have a VNC client installed on your local machine to be able to connect to and interact with the desktop environment.

Step 1: Installation

1. Connect to your instance. For more information, see [Connecting to AL2023 instances](#).
2. Install the TigerVNC server package for AL2023.

The `-y` option installs the package without asking for confirmation. If you would like to examine the package before installing, you can omit this option.

```
[ec2-user ~]$ sudo dnf install -y tigervnc-server
```

Step 2: Configuration

1. Ensure the user has configured a VNC password.

```
[ec2-user ~]$ vncpasswd
```

2. Assign a display number to the user.

```
[ec2-user ~]$ sudo vi /etc/tigervnc/vncserver.users
```

Add the following configuration:

```
:1=ec2-user
```

Note

You can assign any display number to the user. We are using display `:1` for the sake of this example.

3. Edit the VNC server configuration file.

```
[ec2-user ~]$ sudo vi /etc/tigervnc/vncserver-config-defaults
```

Add the following configuration:

```
session=gnome
securitytypes=vncauth,tlsvnc
geometry=1920x1080
localhost
alwaysshared
```

Note

You can change the resolution of the display using the `geometry` parameter. We are using 1920x1080 for the sake of this example.

4. Start the VNC server. This process needs to be repeated every time you restart your instance. If you would like to automate the process of starting this service, see the optional section below.

```
[ec2-user ~]$ sudo systemctl start vncserver@:1
```

Important

When starting the `vncserver` service, the part after the `@` must match the display number set for the user in the `/etc/tigervnc/vncserver.users` file.

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

Step 3: Connect using a VNC client

The VNC server exposes a TCP socket for client connections. While you could expose the VNC port directly through your security group, this tutorial demonstrates using SSH tunneling as a more secure approach by encrypting the connection between your local machine and the EC2 instance. Once connected through the tunnel, you'll authenticate to the VNC server using the password

you configured in the previous step. 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 5901:localhost:5901 ec2-user@<address>
```

Note

Replace `<keypair>` with the path to your SSH key and `<address>` with your instance's public IP or DNS name. The port changes based on the display number that was used to start the `vncserver`. For example, display `:1` uses port 5901, display `:2` uses port 5902, etc.

2. Use your VNC client to connect to `localhost:5901` or `127.0.0.1:5901` with the previously set VNC password.

Important

Keep the SSH tunnel open while using VNC. If the SSH tunnel isn't open, you will not be able to use your VNC client to view and interact with the desktop environment.

(Optional) Start service at boot

If you plan to use VNC regularly, you may want to configure the VNC server to start automatically when your instance boots. This eliminates the need to manually start the VNC server each time you restart your instance. This configuration ensures that your graphical desktop environment is ready and accessible as soon as your instance completes its startup process.

- Configure the service to start at boot.

```
[ec2-user ~]$ sudo systemctl enable vncserver@:1
```

Important

When enabling the `vncserver` service, the part after the `@` must match the display number set for the user in the `/etc/tigervnc/vncserver.users` file.

Additionally, you can pass the `--now` argument after `enable` to start the service immediately.

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

(Optional) Disable the idle lockscreen

- Set the idle delay to zero in order to disable the lockscreen when the user has been inactive for a longer period of time.

```
[ec2-user ~]$ gsettings set org.gnome.desktop.session idle-delay 0
```

Related topics

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

- [Tutorial: Install the GNOME desktop environment on AL2023](#)
- [What Is Amazon DCV?](#) in the *Amazon DCV Administrator Guide*

Using Multi-Gen LRU (MGLRU) on AL2023 kernels

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 AL2023 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.*

Bit	Components
0	The main switch for the multi-gen LRU.
1	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 (<code>mmap_lock</code>). 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.
2	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]	Enable/disable 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
```

Using Amazon Linux 2023 outside of Amazon EC2

The Amazon Linux 2023 container images can be run in compatible container runtime environments. For more information on how to use Amazon Linux 2023 inside a container, see [AL2023 in containers](#).

Amazon Linux 2023 (AL2023) can also be run as a virtualized guest outside of directly being run on Amazon EC2. There are currently KVM (qcow2), VMware (OVA), and Hyper-V (vhdx) images available.

Note

The configuration of Amazon Linux 2023 images differs from Amazon Linux 2. If you are coming from [Running Amazon Linux 2 as a virtual machine on premises](#) you will need to adapt your configuration to be compatible with AL2023.

Download Amazon Linux 2023 images for use with KVM, VMware, and Hyper-V

Amazon Linux 2023 disk images for use with KVM, VMware, and Hyper-V can be downloaded from cdn.amazonlinux.com.

Supported configurations of Amazon Linux 2023 for use in non-Amazon EC2 virtualized environments

This section covers the requirements for running Amazon Linux 2023 in non-Amazon EC2 virtualized environments such as on KVM, VMware, or and Hyper-V.

The base [AL2023 system requirements](#) apply to all non-Amazon EC2 virtualized environments. A list of supported device models is detailed for each hypervisor environment in the following topics.

KVM, VMware, and Hyper-V provide many configuration options, and care needs to be taken in order to configure them for your security, performance, and reliability needs. For more information, check the documentation provided by your hypervisor.

Topics

- [Requirements for running AL2023 on KVM](#)
- [Requirements for running AL2023 on VMware](#)
- [Requirements for running Amazon Linux 2023 on Hyper-V](#)

Requirements for running AL2023 on KVM

This section describes the requirements for running AL2023 on KVM. The KVM images of AL2023 are available for both aarch64 and x86-64 architectures. These requirements are in addition to the base [AL2023 system requirements](#) for the KVM images.

Topics

- [KVM host requirements for running AL2023 on KVM](#)
- [Device support for AL2023 on KVM](#)
- [Boot mode \(UEFI and BIOS\) support for AL2023 on KVM](#)
- [Limitations running AL2023 on KVM](#)

KVM host requirements for running AL2023 on KVM

The KVM images are currently qualified on a host running Ubuntu 22.04.3 LTS with qemu version 6.2+dfsg-2ubuntu6.15, provided by this Ubuntu version, using a q35 machine type for x86-64 and a virt machine type for aarch64.

Device support for AL2023 on KVM

The qemu device models tested for use with AL2023 KVM images (both aarch64 and x86-64) are:

- virtio-blk (virtio block device)
- virtio-scsi (virtio SCSI controller with disk device)
- virtio-net (virtio network device)
- ahci (for use with the virtual CD-ROM drive)
- usb-storage (over xhci)

Additional qemu device models enabled in AL2023 KVM image qualification, but not heavily exercised are:

- VGA (qemu VGA) on x86-64 only
- virtio-rng (virtual random number generator)
- legacy AT keyboard and PS/2 mouse devices
- legacy serial device

Boot mode (UEFI and BIOS) support for AL2023 on KVM

The x86-64 image is tested with both legacy BIOS and UEFI boot modes. The aarch64 images are tested with UEFI boot mode.

Note

By default, when using UEFI boot mode, some virtual machine managers will provision the VM with Microsoft Secure Boot keys which enables Secure Boot. This configuration will not boot AL2023.

Because the AL2023 boot loader isn't signed by Microsoft, the VM must be provisioned either without UEFI keys, or with the AL2023 keys for Secure Boot.

Important

Secure Boot support for KVMimages has not been validated yet.

Limitations running AL2023 on KVM

There are some known limitations in running AL2023 on KVM.

Note

Code implementing some of the listed unsupported functionality might exist in AL2023 and function correctly. The list of unsupported functionality exists so that you can make informed decisions about what to rely upon working today, and what the Amazon Linux team will qualify as working as part of future updates.

Known Limitations with running AL2023 on KVM

- The KVM guest agent is not currently packaged or supported.
- Hot plugging and unplugging CPU, memory, or any other device type is not supported.
- VM hibernation is not supported.
- VM migration is not supported.
- Passthrough of any device such as through PCI Passthrough, or USB Passthrough is not supported.

Requirements for running AL2023 on VMware

This section describes the requirements for running AL2023 on VMware. The VMware images of AL2023 are available for only the x86-64 architecture. VMware images for aarch64 are not available or supported. These requirements are in addition to the base [AL2023 system requirements](#) for the VMware images.

Topics

- [VMware host requirements for running AL2023 on VMware](#)
- [Device support for AL2023 on VMware](#)
- [Boot mode \(UEFI and BIOS\) support for AL2023 on VMware](#)
- [Limitations running AL2023 on VMware](#)

VMware host requirements for running AL2023 on VMware

The AL2023 VMware OVA images are currently qualified on the following:

- VMware Workstation 17.5.0 running on hosts using an Intel(R) Xeon(R) Platinum 8124M processor
- VMware vSphere 8.0 using an Intel(R) Xeon(R) Platinum 8275CL processor

The AL2023 VMware OVA images specify a *Machine Hardware Version* of 13.

VMware Machine Hardware Version 13 is supported by:

- ESXi 6.5 or later

- VMware Workstation 14 or later

Device support for AL2023 on VMware

The following VMware device models were tested for use with AL2023 VMware OVA images (x86-64 only):

- `vmw_pvscsi` (VMware paravirtualized SCSI controller)
- `vmxnet3` (VMware paravirtualized network device)
- `ata_piix` (legacy IDE for use with the virtual CD-ROM drive only)

Additional VMware device models enabled in AL2023 VMware image qualification, but not heavily exercised:

- `vmw_vmci` and related `vsock` interface (virtual socket transport for the VMware guest agent)
- `vmw_balloon` memory balloon device
- VMware SVGA controller
- legacy AT keyboard and PS/2 mouse devices

The VMware guest agent package (`open-vm-tools`) is available and installed by default in the AL2023 VMware OVA images.

Boot mode (UEFI and BIOS) support for AL2023 on VMware

As of the 2023.3.20231211 release, the AL2023 VMware OVA image has been validated in both legacy BIOS and UEFI boot modes. The OVA default configuration is still legacy BIOS but can be changed by the user.

Important

Secure Boot support requires UEFI, which has not been validated for AL2023 running on VMware.

Limitations running AL2023 on VMware

There are some known limitations in running AL2023 on VMware.

Note

Code implementing some of the listed unsupported functionality may exist in AL2023 and function correctly. The list of unsupported functionality exists so that customers can make informed decisions about what to rely upon working today, and what the Amazon Linux team will qualify as working as part of future updates.

Known limitations with running AL2023 on VMware

- UEFI Secure Boot is not currently validated with AL2023 on VMware.
- Hot plugging and unplugging CPU, memory, or any other device type is not supported.
- VM hibernation is not supported.
- VM migration is not supported.
- Passthrough of any device such as through PCI Passthrough, or USB Passthrough is not supported.

Requirements for running Amazon Linux 2023 on Hyper-V

This section covers the requirements for running Amazon Linux 2023 on Hyper-V. The Hyper-V images of AL2023 are available only for the x86-64 architecture. Hyper-V images for aarch64 are not available or supported at this time.

This section covers additional requirements on top of the base [AL2023 system requirements](#) for the Hyper-V images.

Topics

- [Hyper-V host requirements for running Amazon Linux 2023 on Hyper-V](#)
- [Device support for Amazon Linux 2023 on Hyper-V](#)
- [Limitations running Amazon Linux 2023 on Hyper-V](#)

Hyper-V host requirements for running Amazon Linux 2023 on Hyper-V

The main qualification of Amazon Linux 2023 on Hyper-V happens on Windows Server 2022 running on an EC2 `c5.metal` instance.

Device support for Amazon Linux 2023 on Hyper-V

Amazon Linux 2023 is tested on both *Generation 1* and *Generation 2* Hyper-V virtual machines with the following set of virtualized hardware:

- Generation 1 (legacy BIOS boot) VM
- Generation 2 (UEFI boot - No secure boot) VM
- The following device models are tested for use with AL2023 Hyper-V images:
 - Hyper-V virtual storage hv_storvsc for the root disk and the emulated CD-ROM drive on *Generation 2* VMs
 - Emulated PIIX IDE ata_piix for the virtual CD-ROM drive on *Generation 1* VMs
 - Hyper-V virtual ethernet hv_netvsc
- The following device models are enabled but lightly tested:
 - Legacy VGA text mode on *Generation 1* VMs
 - UEFI Firmware based framebuffer simplifiedfb on *Generation 2* VMs
 - Hyper-V Balloon hv_balloon
 - Hyper-V HID/Mouse hid_hyperv
- The following device modes are *not* enabled in AL2023 at this time:
 - Hyper-V PCI pass-through
 - Hyper-V DRM Graphics

Important

For *Generation 2* virtual machines, Secure Boot is not supported and must be disabled prior to launching the virtual machine for a successful boot of Amazon Linux 2023. Hyper-V currently only supports Secure Boot with software components signed by Microsoft's own keys while the Amazon Linux bootloader is signed by an Amazon private key. Hyper-V doesn't support importing 3rd party keys at this point.

Limitations running Amazon Linux 2023 on Hyper-V

The following are some known limitations in running Amazon Linux 2023 on Hyper-V:

Note

Code implementing some of the listed unsupported functionality may exist in AL2023 and function correctly. The list of unsupported functionality exists so that customers can make informed decisions about what to rely upon working today, and what the Amazon Linux team will qualify as working as part of future updates.

Known Limitations with running AL2023 on Hyper-V

- UEFI Secure Boot mode is not currently supported nor functional with AL2023 on Hyper-V
- Hot plugging and unplugging CPU, memory, or any other device type is not supported.
- Virtual Machine (VM) hibernation is not supported.
- Virtual Machine (VM) migration is not supported.
- Passthrough of any device such as through PCI Passthrough, or USB Passthrough is not supported.

Amazon Linux 2023 Set up and `cloud-init` configuration when used outside Amazon EC2

This section covers how to set up and configure a Amazon Linux 2023 virtual machine when not run directly on Amazon EC2, such as when on KVM, VMware, or Hyper-V.

By default, an Amazon Linux 2023 virtual machine images don't come provisioned with any user password or ssh key and will obtain its network configuration via DHCP on the first discovered network interface. This means that by default, without additional configuration, there is no way to connect to the resulting virtual machine.

Thus, some form of configuration needs to be provided to the virtual machine. The standard mechanism to do this for Amazon Linux is via `cloud-init` data sources.

Amazon Linux 2023 has been qualified with the following data sources:

NoCloud

This is the traditional method of configuring on-premises images via a virtual CD-ROM containing a seed ISO9660 image with `cloud-init` configuration files.

VMware

Amazon Linux 2023 additionally supports configuring VMware images running on vSphere via the VMware specific data source via `VMware guestinfo.userdata` and `guestinfo.metadata`.

Note

The configuration of the data sources can differ from Amazon Linux 2. More specifically, Amazon Linux 2023 uses `systemd-networkd` for its configuration and requires the use of `cloud-init` "Networking Config Version 2" as documented in [the cloud-init network configuration documentation](#).

The complete documentation for `cloud-init` configuration mechanisms for the version of `cloud-init` packaged in Amazon Linux 2023 can be found in the [upstream cloud-init documentation](#).

NoCloud (seed.iso) cloud-init configuration for Amazon Linux 2023 on KVM and VMware

This section covers how to create and use a `seed.iso` image to configure Amazon Linux 2023 running on KVM or VMware. Because KVM and VMware environments do not have [Amazon EC2 Instance Meta Data Service \(IMDS\)](#), an alternate method of configuring Amazon Linux 2023 is required, and providing a `seed.iso` image is one of those methods.

The `seed.iso` boot image includes the initial configuration information that is needed to boot and configure your new virtual machine, such as the network configuration, host name, and user data.

Note

The `seed.iso` image includes only the configuration information required to boot the VM. It does not include the Amazon Linux 2023 operating system files.

To generate the `seed.iso` image, you need at least two configuration files, sometimes three:

meta-data

This file typically includes the hostname for the virtual machine.

user-data

This file typically configures user accounts, their passwords, ssh key pairs, and/or access mechanisms. By default, the Amazon Linux 2023 KVM and VMware images create an `ec2-user` user account. You can use the `user-data` configuration file to set the password and/or ssh keys for this default user account.

network-config (optional)

This file typically provides a network configuration for the virtual machine which will override the default one. The default configuration is to use DHCP on the first available network interface.

Create the seed.iso disk image

1. On a Linux or macOS computer, create a new folder named `seedconfig` and navigate into it.

Note

Using Windows or another Operating System to complete these steps is possible, but you will have to find the equivalent tool to `mkisofs` to complete creating the `seed.iso` image.

2. Create the meta-data configuration file.
 - a. Create a new file named `meta-data`.
 - b. Open the meta-data file using your preferred editor and add the following, replacing `vm-hostname` with the host name for the VM:

```
#cloud-config
local-hostname: vm-hostname
```

- c. Save and close the meta-data configuration file.
3. Create the user-data configuration file.
 - a. Create a new file named `user-data`.

- b. Open the user-data file using your preferred editor and add the following, making substitutions as needed:

```
#cloud-config
#vim:syntax=yaml
users:
# A user by the name 'ec2-user' is created in the image by default.
- default
- name: ec2-user
ssh_authorized_keys:
- ssh-rsa ssh-key
# In the above line, replace ssh key with the content of your ssh public key.
```

- c. You can optionally add more user accounts to the user-data configuration file.

You can specify additional user accounts, their access mechanisms, passwords, and key pairs. For more information about the supported directives, see the [upstream cloud-init documentation](#).

- d. Save and close the user-data configuration file.
4. (Optional) Create the network-config configuration file.
 - a. Create a new file named network-config.
 - b. Open the network-config file using your preferred editor and add the following, replacing the various IP addresses with the appropriate ones for your setup.

```
#cloud-config
version: 2
ethernets:
  enp1s0:
    addresses:
      - 192.168.122.161/24
    gateway4: 192.168.122.1
    nameservers:
      addresses: 192.168.122.1
```

Note

cloud-init network configuration provides mechanisms to match against the MAC address of the interface instead of specifying the interface name which can

change depending on the VM configuration. This (and more) `cloud-init` features for network configuration are described in more detail in the [upstream cloud-init Network Config Version 2 documentation](#).

- c. Save and close the `network-config` configuration file.
5. Create the `seed.iso` disk image using the meta-data, user-data, and optional `network-config` configuration files created in the previous steps.

Do one of the following, depending on the OS you are creating the `seed.iso` disk image on.

- On Linux systems, use a tool such as **mkisofs** or **genisoimage** to create the completed `seed.iso` file. Navigate into the `seedconfig` folder, and run the following command:

```
$ mkisofs -output seed.iso -volid cidata -joliet -rock user-data meta-data
```

- If you use a `network-config`, include it in the invocation of **mkisofs**:

```
$ mkisofs -output seed.iso -volid cidata -joliet -rock user-data meta-data
network-config
```

- On macOS systems, you can use a tool such as **hdiutil** to generate the finished `seed.iso` file. Since **hdiutil** takes a pathname rather than a list of files, the same invocation can be used regardless of if a `network-config` configuration file has been created or not.

```
$ hdiutil makehybrid -o seed.iso -hfs -joliet -iso -default-volume-name cidata
seedconfig/
```

6. The resulting `seed.iso` file can now be attached to your new Amazon Linux 2023 Virtual Machine using a virtual CD-ROM drive for `cloud-init` to find on first boot and apply the configuration to the system.

VMware guestinfo `cloud-init` configuration for AL2023 on VMware

VMware environments do not have the [Amazon EC2 Instance Meta Data Service \(IMDS\)](#), so an alternate method of configuring AL2023 is required. This section describes how to use an alternative configuration mechanism to the `seed.iso` virtual CD-ROM drive that is available in VMware vSphere.

This method of configuration uses the VMware extraconfig mechanism to provide configuration data to cloud-init. For each of the following keys, a corresponding **keyname.encoding** property must be provided.

The following keys can be provided to the VMware extraconfig mechanism.

guestinfo.metadata

JSON or YAML containing cloud-init meta-data

guestinfo.userdata

A YAML document containing cloud-init user-data in the cloud-config format.

guestinfo.vendordata (optional)

YAML containing cloud-init vendor-data

The corresponding encoding properties (`guestinfo.metadata.encoding`, `guestinfo.userdata.encoding`, and `guestinfo.vendordata.encoding`) can contain:

base64

The content of the property is base64 encoded.

gzip+base64

The content of the property is compressed with gzip after base64 encoding.

 **Note**

The `seed.iso` method supports a separate (optional) `network-config` configuration file. VMware `guestinfo` differs in how the networking configuration is provided. Additional information is provided in the following section.

If an explicit network configuration is desired, it should be embedded in the metadata in the form of two YAML or JSON properties:

network

Contains the encoded network configuration in JSON or YAML form.

network.encoding

Contains the encoding of the above network configuration data. The cloud-init supported encodings are the same as for the guestinfo data: base64 and gzip+base64.

Example Using the VMware vSphere govc CLI tool to pass configuration with guestinfo

1. Prepare the meta-data, user-data, and optional network-config configuration files as described in [NoCloud \(seed.iso\) cloud-init configuration for Amazon Linux 2023 on KVM and VMware](#).
2. Convert the configuration files into formats usable by VMware guestinfo.

```
# 'meta-data', `user-data` and `network-config` are the configuration
# files in the same format that would be used by a NoCloud (seed.iso)
# data source, read-them and convert them to VMware guestinfo
#
# The VM_NAME variable is assumed to be set to the name of the VM
# It is assumed that the necessary govc environment (credentials etc...) are
# already set

metadata=$(cat "meta-data")
userdata=$(cat "user-data")
if [ -e "network-config" ] ; then
    # We need to embed the network config inside the meta-data
    netconf=$(base64 -w0 "network-config")
    metadata=$(printf "%s\nnetwork: %s\nnetwork.encoding: base64" "$metadata"
"$netconf")
fi
metadata=$(base64 -w0 <<< "$metadata")
govc vm.change -vm "$VM_NAME" \
    -e guestinfo.metadata="$metadata" \
    -e guestinfo.metadata.encoding="base64"
userdata=$(base64 -w0 <<< "$userdata")
govc vm.change -vm "$VM_NAME" \
    -e guestinfo.userdata="$userdata" \
    -e guestinfo.userdata.encoding="base64"
```

Comparing packages installed on Amazon Linux 2023 standard AMI with the AL2023 KVM Image

A comparison of the RPMs present on the AL2023 standard AMI compared with the RPMs present on the AL2023 KVM image.

Package	AMI	KVM
acl	2.3.1	2.3.1
acpid	2.0.32	
alternatives	1.15	1.15
amazon-chrony-config	4.3	
amazon-ec2-net-utils	2.5.1	
amazon-linux-onprem		1.2
amazon-linux-repo-cdn		2023.6.20241031
amazon-linux-repo-s3	2023.6.20241031	
amazon-linux-sb-keys	2023.1	2023.1
amazon-onprem-network		1.2
amazon-rpm-config	228	228
amazon-ssm-agent	3.3.987.0	3.3.987.0
amd-ucode-firmware	20210208 (noarch)	20210208 (noarch)
at	3.1.23	3.1.23
attr	2.5.1	2.5.1

Package	AMI	KVM
audit	3.0.6	3.0.6
audit-libs	3.0.6	3.0.6
aws-cfn-bootstrap	2.0	
awscli-2	2.15.30	2.15.30
basesystem	11	11
bash	5.2.15	5.2.15
bash-completion	2.11	2.11
bc	1.07.1	1.07.1
bind-libs	9.18.28	9.18.28
bind-license	9.18.28	9.18.28
bind-utils	9.18.28	9.18.28
binutils	2.39	2.39
boost-filesystem	1.75.0	1.75.0
boost-system	1.75.0	1.75.0
boost-thread	1.75.0	1.75.0
bzip2	1.0.8	1.0.8
bzip2-libs	1.0.8	1.0.8
ca-certificates	2023.2.68	2023.2.68
c-ares	1.19.1	
checkpolicy	3.4	3.4

Package	AMI	KVM
chkconfig	1.15	1.15
chrony	4.3	4.3
cloud-init	22.2.2	22.2.2
cloud-init-cfg-ec2	22.2.2	
cloud-init-cfg-onp rem		22.2.2
cloud-utils-growpart	0.31	0.31
coreutils	8.32	8.32
coreutils-common	8.32	8.32
cpio	2.13	2.13
cracklib	2.9.6	2.9.6
cracklib-dicts	2.9.6	2.9.6
crontabs	1.11	1.11
crypto-policies	20220428	20220428
crypto-policies-sc ripts	20220428	20220428
cryptsetup	2.6.1	2.6.1
cryptsetup-libs	2.6.1	2.6.1
curl-minimal	8.5.0	8.5.0
cyrus-sasl-lib	2.1.27	2.1.27
cyrus-sasl-plain	2.1.27	2.1.27

Package	AMI	KVM
dbus	1.12.28	1.12.28
dbus-broker	32	32
dbus-common	1.12.28	1.12.28
dbus-libs	1.12.28	1.12.28
device-mapper	1.02.185	1.02.185
device-mapper-libs	1.02.185	1.02.185
diffutils	3.8	3.8
dnf	4.14.0	4.14.0
dnf-data	4.14.0	4.14.0
dnf-plugin-release-notification	1.2	1.2
dnf-plugins-core	4.3.0	4.3.0
dnf-plugin-support-info	1.2	1.2
dnf-utils	4.3.0	4.3.0
dosfstools	4.2	4.2
dracut	055	055
dracut-config-ec2	3.0	
dracut-config-generic	055	055
dwz	0.14	0.14

Package	AMI	KVM
dyninst	10.2.1	10.2.1
e2fsprogs	1.46.5	1.46.5
e2fsprogs-libs	1.46.5	1.46.5
ec2-hibinit-agent	1.0.8	
ec2-instance-connect	1.1	
ec2-instance-connect-selinux	1.1	
ec2-utils	2.2.0	
ed	1.14.2	1.14.2
efi-filesystem	5	5
efi-srpm-macros	5	5
efivar	38	38
efivar-libs	38	38
elfutils-debuginfod-client	0.188	0.188
elfutils-default-yama-scope	0.188	0.188
elfutils-libelf	0.188	0.188
elfutils-libs	0.188	0.188
ethtool	5.15	5.15
expat	2.5.0	2.5.0

Package	AMI	KVM
file	5.39	5.39
file-libs	5.39	5.39
filesystem	3.14	3.14
findutils	4.8.0	4.8.0
fonts-srpm-macros	2.0.5	2.0.5
fstrm	0.6.1	0.6.1
fuse-libs	2.9.9	2.9.9
gawk	5.1.0	5.1.0
gdbm-libs	1.19	1.19
gdisk	1.0.8	1.0.8
gettext	0.21	0.21
gettext-libs	0.21	0.21
ghc-srpm-macros	1.5.0	1.5.0
glib2	2.74.7	2.74.7
glibc	2.34	2.34
glibc-all-langpacks	2.34	2.34
glibc-common	2.34	2.34
glibc-gconv-extra	2.34	2.34
glibc-locale-source	2.34	2.34
gmp	6.2.1	6.2.1

Package	AMI	KVM
<u>gnupg2-minimal</u>	2.3.7	2.3.7
gnutls	3.8.0	3.8.0
go-srpm-macros	3.2.0	3.2.0
gpgme	1.15.1	1.15.1
gpm-libs	1.20.7	1.20.7
grep	3.8	3.8
groff-base	1.22.4	1.22.4
grub2-common	2.06	2.06
grub2-efi-aa64-ec2	2.06 (aarch64)	2.06 (aarch64)
grub2-efi-x64-ec2	2.06 (x86_64)	2.06 (x86_64)
grub2-pc		2.06 (x86_64)
grub2-pc-modules	2.06	2.06 (noarch)
grub2-tools	2.06	2.06
grub2-tools-minimal	2.06	2.06
grubby	8.40	8.40
gssproxy	0.8.4	0.8.4
gzip	1.12	1.12
hostname	3.23	3.23
hunspell	1.7.0	1.7.0
hunspell-en	0.20140811.1	0.20140811.1

Package	AMI	KVM
hunspell-en-GB	0.20140811.1	0.20140811.1
hunspell-en-US	0.20140811.1	0.20140811.1
hunspell-filesystem	1.7.0	1.7.0
hwdata	0.384	0.384
info	6.7	6.7
inih	49	49
initscripts	10.09	10.09
iproute	6.10.0	6.10.0
iputils	20210202	20210202
irqbalance	1.9.0	1.9.0
jansson	2.14	2.14
jemalloc	5.2.1	5.2.1
jitterentropy	3.4.1	3.4.1
jq	1.7.1	
json-c	0.14	0.14
kbd	2.4.0	2.4.0
kbd-misc	2.4.0	2.4.0
kernel	6.1.112	6.1.112
kernel-libbpf	6.1.112	6.1.112
kernel-livepatch-r epo-cdn		2023.6.20241031

Package	AMI	KVM
kernel-livepatch-r epo-s3	2023.6.20241031	
kernel-modules-extra		6.1.112
kernel-modules-ext ra-common		6.1.112
kernel-srpm-macros	1.0	1.0
kernel-tools	6.1.112	6.1.112
keyutils	1.6.3	1.6.3
keyutils-libs	1.6.3	1.6.3
kmod	29	29
kmod-libs	29	29
kpatch-runtime	0.9.7	0.9.7
krb5-libs	1.21.3	1.21.3
less	608	608
libacl	2.3.1	2.3.1
libaio	0.3.111	0.3.111
libarchive	3.7.4	3.7.4
libargon2	20171227	20171227
libassuan	2.5.5	2.5.5
libattr	2.5.1	2.5.1
libbasicobjects	0.1.1	0.1.1

Package	AMI	KVM
libblkid	2.37.4	2.37.4
libcap	2.48	2.48
libcap-ng	0.8.2	0.8.2
libcbor	0.7.0	0.7.0
libcollection	0.7.0	0.7.0
libcom_err	1.46.5	1.46.5
libcomps	0.1.20	0.1.20
libconfig	1.7.2	1.7.2
libcurl-minimal	8.5.0	8.5.0
libdb	5.3.28	5.3.28
libdhash	0.5.0	
libdnf	0.69.0	0.69.0
libeconf	0.4.0	0.4.0
libedit	3.1	3.1
libev	4.33	4.33
libevent	2.1.12	2.1.12
libfdisk	2.37.4	2.37.4
libffi	3.4.4	3.4.4
libfido2	1.10.0	1.10.0
libgcc	11.4.1	11.4.1

Package	AMI	KVM
libgcrypt	1.10.2	1.10.2
libgomp	11.4.1	11.4.1
libgpg-error	1.42	1.42
libibverbs	48.0	48.0
libidn2	2.3.2	2.3.2
libini_config	1.3.1	1.3.1
libkcapi	1.4.0	1.4.0
libkcapi-hmacalc	1.4.0	1.4.0
libldb	2.6.2	
libmaxminddb	1.5.2	1.5.2
libmetalink	0.1.3	0.1.3
libmnl	1.0.4	1.0.4
libmodulemd	2.13.0	2.13.0
libmount	2.37.4	2.37.4
libnfsidmap	2.5.4	2.5.4
libnghttp2	1.59.0	1.59.0
libnl3	3.5.0	3.5.0
libpath_utils	0.2.1	0.2.1
libpcap	1.10.1	1.10.1
libpipeline	1.5.3	1.5.3

Package	AMI	KVM
libpkgconf	1.8.0	1.8.0
libpsl	0.21.1	0.21.1
libpwquality	1.4.4	1.4.4
libref_array	0.1.5	0.1.5
librepo	1.14.5	1.14.5
libreport-filessystem	2.15.2	2.15.2
libseccomp	2.5.3	2.5.3
libselinux	3.4	3.4
libselinux-utils	3.4	3.4
libsemanage	3.4	3.4
libsepol	3.4	3.4
libsigsegv	2.13	2.13
libsmartcols	2.37.4	2.37.4
libsolv	0.7.22	0.7.22
libss	1.46.5	1.46.5
libsss_certmap	2.9.4	
libsss_idmap	2.9.4	2.9.4
libsss_nss_idmap	2.9.4	2.9.4
libsss_sudo	2.9.4	
libstdc++	11.4.1	11.4.1

Package	AMI	KVM
libstoragemgmt	1.9.4	1.9.4
libtalloc	2.3.4	
libtasn1	4.19.0	4.19.0
libtdb	1.4.7	
libtevent	0.13.0	
libtextstyle	0.21	0.21
libtirpc	1.3.3	1.3.3
libunistring	0.9.10	0.9.10
libuser	0.63	0.63
libutempter	1.2.1	1.2.1
libuuid	2.37.4	2.37.4
libuv	1.47.0	1.47.0
libverto	0.3.2	0.3.2
libverto-libev	0.3.2	0.3.2
libxcrypt	4.4.33	4.4.33
libxml2	2.10.4	2.10.4
libyaml	0.2.5	0.2.5
libzstd	1.5.5	1.5.5
linux-firmware-wheel	20210208 (noarch)	20210208 (noarch)
lm_sensors-libs	3.6.0	3.6.0

Package	AMI	KVM
lmdb-libs	0.9.29	0.9.29
logrotate	3.20.1	3.20.1
lsyf	4.94.0	4.94.0
lua-libs	5.4.4	5.4.4
lua-srpm-macros	1	1
lz4-libs	1.9.4	1.9.4
man-db	2.9.3	2.9.3
man-pages	5.10	5.10
microcode_ctl	2.1 (x86_64)	2.1 (x86_64)
mpfr	4.1.0	4.1.0
nano	5.8	5.8
ncurses	6.2	6.2
ncurses-base	6.2	6.2
ncurses-libs	6.2	6.2
nettle	3.8	3.8
net-tools	2.0	2.0
newt	0.52.21	0.52.21
nfs-utils	2.5.4	2.5.4
npth	1.6	1.6
nspr	4.35.0	4.35.0

Package	AMI	KVM
nss	3.90.0	3.90.0
nss-softokn	3.90.0	3.90.0
nss-softokn-freebl	3.90.0	3.90.0
nss-sysinit	3.90.0	3.90.0
nss-util	3.90.0	3.90.0
ntsysv	1.15	1.15
numactl-libs	2.0.14	2.0.14
ocaml-srpm-macros	6	6
oniguruma	6.9.7.1	
openblas-srpm-macros	2	2
openldap	2.4.57	2.4.57
openssh	8.7p1	8.7p1
openssh-clients	8.7p1	8.7p1
openssh-server	8.7p1	8.7p1
openssl	3.0.8	3.0.8
openssl-libs	3.0.8	3.0.8
openssl-pkcs11	0.4.12	0.4.12
os-prober	1.77	1.77
p11-kit	0.24.1	0.24.1
p11-kit-trust	0.24.1	0.24.1

Package	AMI	KVM
package-notes-srpm-macros	0.4	0.4
pam	1.5.1	1.5.1
parted	3.4	3.4
passwd	0.80	0.80
pciutils	3.7.0	3.7.0
pciutils-libs	3.7.0	3.7.0
pcre2	10.40	10.40
pcre2-syntax	10.40	10.40
perl-Carp	1.50	1.50
perl-Class-Struct	0.66	0.66
perl-constant	1.33	1.33
perl-DynaLoader	1.47	1.47
perl-Encode	3.15	3.15
perl-Errno	1.30	1.30
perl-Exporter	5.74	5.74
perl-Fcntl	1.13	1.13
perl-File-Basename	2.85	2.85
perl-File-Path	2.18	2.18
perl-File-stat	1.09	1.09
perl-File-Temp	0.231.100	0.231.100

Package	AMI	KVM
perl-Getopt-Long	2.52	2.52
perl-Getopt-Std	1.12	1.12
perl-HTTP-Tiny	0.078	0.078
perl-if	0.60.800	0.60.800
perl-interpreter	5.32.1	5.32.1
perl-IO	1.43	1.43
perl-IPC-Open3	1.21	1.21
perl-libs	5.32.1	5.32.1
perl-MIME-Base64	3.16	3.16
perl-mro	1.23	1.23
perl-overload	1.31	1.31
perl-overloading	0.02	0.02
perl-parent	0.238	0.238
perl-PathTools	3.78	3.78
perl-Pod-Escapes	1.07	1.07
perl-podlators	4.14	4.14
perl-Pod-Perldoc	3.28.01	3.28.01
perl-Pod-Simple	3.42	3.42
perl-Pod-Usage	2.01	2.01
perl-POSIX	1.94	1.94

Package	AMI	KVM
perl-Scalar-List-Utils	1.56	1.56
perl-SelectSaver	1.02	1.02
perl-Socket	2.032	2.032
perl-srpm-macros	1	1
perl-Storable	3.21	3.21
perl-subs	1.03	1.03
perl-Symbol	1.08	1.08
perl-Term-ANSIColor	5.01	5.01
perl-Term-Cap	1.17	1.17
perl-Text-ParseWords	3.30	3.30
perl-Text-Tabs+Wrap	2021.0726	2021.0726
perl-Time-Local	1.300	1.300
perl-vars	1.05	1.05
pkgconf	1.8.0	1.8.0
pkgconf-m4	1.8.0	1.8.0
pkgconf-pkg-config	1.8.0	1.8.0
policycoreutils	3.4	3.4
policycoreutils-python-utils	3.4	
popt	1.18	1.18

Package	AMI	KVM
procps-ng	3.3.17	3.3.17
protobuf-c	1.4.1	1.4.1
psacct	6.6.4	6.6.4
psmisc	23.4	23.4
publicsuffix-list-dafsa	20240212	20240212
python3	3.9.16	3.9.16
python3-attrs	20.3.0	20.3.0
python3-audit	3.0.6	3.0.6
python3-awscrt	0.19.19	0.19.19
python3-babel	2.9.1	2.9.1
python3-cffi	1.14.5	1.14.5
python3-chardet	4.0.0	4.0.0
python3-colorama	0.4.4	0.4.4
python3-configobj	5.0.6	5.0.6
python3-cryptography	36.0.1	36.0.1
python3-daemon	2.3.0	
python3-dateutil	2.8.1	2.8.1
python3-dbus	1.2.18	1.2.18
python3-distro	1.5.0	1.5.0
python3-dnf	4.14.0	4.14.0

Package	AMI	KVM
python3-dnf-plugins-core	4.3.0	4.3.0
python3-docutils	0.16	0.16
python3-gpg	1.15.1	1.15.1
python3-hawkey	0.69.0	0.69.0
python3-idna	2.10	2.10
python3-jinja2	2.11.3	2.11.3
python3-jmespath	0.10.0	0.10.0
python3-jsonpatch	1.21	1.21
python3-jsonpointer	2.0	2.0
python3-jsonschema	3.2.0	3.2.0
python3-libcomps	0.1.20	0.1.20
python3-libdnf	0.69.0	0.69.0
python3-libs	3.9.16	3.9.16
python3-libselenium	3.4	3.4
python3-libsemanage	3.4	3.4
python3-libstorage mgmt	1.9.4	1.9.4
python3-lockfile	0.12.2	
python3-markupsafe	1.1.1	1.1.1
python3-netifaces	0.10.6	0.10.6

Package	AMI	KVM
python3-oauthlib	3.0.2	3.0.2
python3-pip-wheel	21.3.1	21.3.1
python3-ply	3.11	3.11
python3-policycore utils	3.4	3.4
python3-prettytable	0.7.2	0.7.2
python3-prompt-too lkit	3.0.24	3.0.24
python3-pycparser	2.20	2.20
python3-pyrsistent	0.17.3	0.17.3
python3-pyserial	3.4	3.4
python3-pysocks	1.7.1	1.7.1
python3-pytz	2022.7.1	2022.7.1
python3-pyyaml	5.4.1	5.4.1
python3-requests	2.25.1	2.25.1
python3-rpm	4.16.1.3	4.16.1.3
python3-ruamel-yaml	0.16.6	0.16.6
python3-ruamel-yaml- clib	0.1.2	0.1.2
python3-setuptools	4.4.1	4.4.1
python3-setuptools	59.6.0	59.6.0

Package	AMI	KVM
python3-setuptools-wheel	59.6.0	59.6.0
python3-six	1.15.0	1.15.0
python3-systemd	235	235
python3-urllib3	1.25.10	1.25.10
python3-wcwidth	0.2.5	0.2.5
python-chevron	0.13.1	
python-srpm-macros	3.9	3.9
quota	4.06	4.06
quota-nls	4.06	4.06
readline	8.1	8.1
rng-tools	6.14	6.14
rootfiles	8.1	8.1
rpcbind	1.2.6	1.2.6
rpm	4.16.1.3	4.16.1.3
rpm-build-libs	4.16.1.3	4.16.1.3
rpm-libs	4.16.1.3	4.16.1.3
rpm-plugin-selinux	4.16.1.3	4.16.1.3
rpm-plugin-systemd-inhibit	4.16.1.3	4.16.1.3
rpm-sign-libs	4.16.1.3	4.16.1.3

Package	AMI	KVM
rsync	3.2.6	3.2.6
rust-srpm-macros	21	21
sbsigntools	0.9.4	0.9.4
screen	4.8.0	4.8.0
sed	4.8	4.8
selinux-policy	38.1.45	38.1.45
selinux-policy-targeted	38.1.45	38.1.45
setup	2.13.7	2.13.7
shadow-utils	4.9	4.9
slang	2.3.2	2.3.2
sqlite-libs	3.40.0	3.40.0
sssd-client	2.9.4	2.9.4
sssd-common	2.9.4	
sssd-kcm	2.9.4	
sssd-nfs-idmap	2.9.4	
strace	6.8	6.8
sudo	1.9.15	1.9.15
sysctl-defaults	1.0	1.0
sysstat	12.5.6	12.5.6
systemd	252.23	252.23

Package	AMI	KVM
systemd-libs	252.23	252.23
systemd-networkd	252.23	252.23
systemd-pam	252.23	252.23
systemd-resolved	252.23	252.23
systemd-udev	252.23	252.23
system-release	2023.6.20241031	2023.6.20241031
systemtap-runtime	4.8	4.8
tar	1.34	1.34
tbb	2020.3	2020.3
tcpdump	4.99.1	4.99.1
tcsh	6.24.07	6.24.07
time	1.9	1.9
traceroute	2.1.3	2.1.3
tzdata	2024a	2024a
unzip	6.0	6.0
update-motd	2.2	2.2
userspace-rcu	0.12.1	0.12.1
util-linux	2.37.4	2.37.4
util-linux-core	2.37.4	2.37.4
vim-common	9.0.2153	9.0.2153

Package	AMI	KVM
vim-data	9.0.2153	9.0.2153
vim-enhanced	9.0.2153	9.0.2153
vim-filesystem	9.0.2153	9.0.2153
vim-minimal	9.0.2153	9.0.2153
wget	1.21.3	1.21.3
which	2.21	2.21
words	3.0	3.0
xfsdump	3.1.11	3.1.11
xfspgrog	5.18.0	5.18.0
xxd	9.0.2153	9.0.2153
xxhash-libs	0.8.0	0.8.0
xz	5.2.5	5.2.5
xz-libs	5.2.5	5.2.5
yum	4.14.0	4.14.0
zip	3.0	3.0
zlib	1.2.11	1.2.11
zram-generator	1.1.2	
zram-generator-def aults	1.1.2	
zstd	1.5.5	1.5.5

Comparing packages installed on Amazon Linux 2023 standard AMI with the AL2023 VMware OVA Image

A comparison of the RPMs present on the AL2023 standard AMI compared with the RPMs present on the AL2023 VMware OVA image.

Package	AMI	VMware OVA
acl	2.3.1	2.3.1
acpid	2.0.32	
alternatives	1.15	1.15
amazon-chrony-config	4.3	
amazon-ec2-net-utils	2.5.1	
amazon-linux-onprem		1.2
amazon-linux-repo-cdn		2023.6.20241031
amazon-linux-repo-s3	2023.6.20241031	
amazon-linux-sb-keys	2023.1	2023.1
amazon-onprem-network		1.2
amazon-rpm-config	228	228
amazon-ssm-agent	3.3.987.0	3.3.987.0
amd-ucode-firmware	20210208	20210208
at	3.1.23	3.1.23
attr	2.5.1	2.5.1

Package	AMI	VMware OVA
audit	3.0.6	3.0.6
audit-libs	3.0.6	3.0.6
aws-cfn-bootstrap	2.0	
awscli-2	2.15.30	2.15.30
basesystem	11	11
bash	5.2.15	5.2.15
bash-completion	2.11	2.11
bc	1.07.1	1.07.1
bind-libs	9.18.28	9.18.28
bind-license	9.18.28	9.18.28
bind-utils	9.18.28	9.18.28
binutils	2.39	2.39
boost-filesystem	1.75.0	1.75.0
boost-system	1.75.0	1.75.0
boost-thread	1.75.0	1.75.0
bzip2	1.0.8	1.0.8
bzip2-libs	1.0.8	1.0.8
ca-certificates	2023.2.68	2023.2.68
c-ares	1.19.1	
checkpolicy	3.4	3.4

Package	AMI	VMware OVA
chkconfig	1.15	1.15
chrony	4.3	4.3
cloud-init	22.2.2	22.2.2
cloud-init-cfg-ec2	22.2.2	
cloud-init-cfg-onp rem		22.2.2
cloud-utils-growpart	0.31	0.31
coreutils	8.32	8.32
coreutils-common	8.32	8.32
cpio	2.13	2.13
cracklib	2.9.6	2.9.6
cracklib-dicts	2.9.6	2.9.6
crontabs	1.11	1.11
crypto-policies	20220428	20220428
crypto-policies-sc ripts	20220428	20220428
cryptsetup	2.6.1	2.6.1
cryptsetup-libs	2.6.1	2.6.1
curl-minimal	8.5.0	8.5.0
cyrus-sasl-lib	2.1.27	2.1.27
cyrus-sasl-plain	2.1.27	2.1.27

Package	AMI	VMware OVA
dbus	1.12.28	1.12.28
dbus-broker	32	32
dbus-common	1.12.28	1.12.28
dbus-libs	1.12.28	1.12.28
device-mapper	1.02.185	1.02.185
device-mapper-libs	1.02.185	1.02.185
diffutils	3.8	3.8
dnf	4.14.0	4.14.0
dnf-data	4.14.0	4.14.0
dnf-plugin-release-notification	1.2	1.2
dnf-plugins-core	4.3.0	4.3.0
dnf-plugin-support-info	1.2	1.2
dnf-utils	4.3.0	4.3.0
dosfstools	4.2	4.2
dracut	055	055
dracut-config-ec2	3.0	
dracut-config-generic	055	055
dwz	0.14	0.14

Package	AMI	VMware OVA
dyninst	10.2.1	10.2.1
e2fsprogs	1.46.5	1.46.5
e2fsprogs-libs	1.46.5	1.46.5
ec2-hibinit-agent	1.0.8	
ec2-instance-connect	1.1	
ec2-instance-connect-selinux	1.1	
ec2-utils	2.2.0	
ed	1.14.2	1.14.2
efi-filesystem	5	5
efi-srpm-macros	5	5
efivar	38	38
efivar-libs	38	38
elfutils-debuginfod-client	0.188	0.188
elfutils-default-yama-scope	0.188	0.188
elfutils-libelf	0.188	0.188
elfutils-libs	0.188	0.188
ethtool	5.15	5.15
expat	2.5.0	2.5.0

Package	AMI	VMware OVA
file	5.39	5.39
file-libs	5.39	5.39
filesystem	3.14	3.14
findutils	4.8.0	4.8.0
fonts-srpm-macros	2.0.5	2.0.5
fstrm	0.6.1	0.6.1
fuse3		3.10.4
fuse3-libs		3.10.4
fuse-common		3.10.4
fuse-libs	2.9.9	2.9.9
gawk	5.1.0	5.1.0
gdbm-libs	1.19	1.19
gdisk	1.0.8	1.0.8
gettext	0.21	0.21
gettext-libs	0.21	0.21
ghc-srpm-macros	1.5.0	1.5.0
glib2	2.74.7	2.74.7
glibc	2.34	2.34
glibc-all-langpacks	2.34	2.34
glibc-common	2.34	2.34

Package	AMI	VMware OVA
glibc-gconv-extra	2.34	2.34
glibc-locale-source	2.34	2.34
gmp	6.2.1	6.2.1
gnupg2-minimal	2.3.7	2.3.7
gnutls	3.8.0	3.8.0
go-srpm-macros	3.2.0	3.2.0
gpgme	1.15.1	1.15.1
gpm-libs	1.20.7	1.20.7
grep	3.8	3.8
groff-base	1.22.4	1.22.4
grub2-common	2.06	2.06
grub2-efi-x64-ec2	2.06	2.06
grub2-pc		2.06
grub2-pc-modules	2.06	2.06
grub2-tools	2.06	2.06
grub2-tools-minimal	2.06	2.06
grubby	8.40	8.40
gssproxy	0.8.4	0.8.4
gzip	1.12	1.12
hostname	3.23	3.23

Package	AMI	VMware OVA
hunspell	1.7.0	1.7.0
hunspell-en	0.20140811.1	0.20140811.1
hunspell-en-GB	0.20140811.1	0.20140811.1
hunspell-en-US	0.20140811.1	0.20140811.1
hunspell-filesystem	1.7.0	1.7.0
hwdata	0.384	0.384
info	6.7	6.7
inih	49	49
initscripts	10.09	10.09
iproute	6.10.0	6.10.0
iputils	20210202	20210202
irqbalance	1.9.0	1.9.0
jansson	2.14	2.14
jemalloc	5.2.1	5.2.1
jitterentropy	3.4.1	3.4.1
jq	1.7.1	
json-c	0.14	0.14
kbd	2.4.0	2.4.0
kbd-misc	2.4.0	2.4.0
kernel	6.1.112	6.1.112

Package	AMI	VMware OVA
kernel-libbpf	6.1.112	6.1.112
kernel-livepatch-r epo-cdn		2023.6.20241031
kernel-livepatch-r epo-s3	2023.6.20241031	
kernel-modules-extra		6.1.112
kernel-modules-ext ra-common		6.1.112
kernel-srpm-macros	1.0	1.0
kernel-tools	6.1.112	6.1.112
keyutils	1.6.3	1.6.3
keyutils-libs	1.6.3	1.6.3
kmod	29	29
kmod-libs	29	29
kpatch-runtime	0.9.7	0.9.7
krb5-libs	1.21.3	1.21.3
less	608	608
libacl	2.3.1	2.3.1
libaio	0.3.111	0.3.111
libarchive	3.7.4	3.7.4
libargon2	20171227	20171227

Package	AMI	VMware OVA
libassuan	2.5.5	2.5.5
libattr	2.5.1	2.5.1
libbasicobjects	0.1.1	0.1.1
libblkid	2.37.4	2.37.4
libcap	2.48	2.48
libcap-ng	0.8.2	0.8.2
libcbor	0.7.0	0.7.0
libcollection	0.7.0	0.7.0
libcom_err	1.46.5	1.46.5
libcomps	0.1.20	0.1.20
libconfig	1.7.2	1.7.2
libcurl-minimal	8.5.0	8.5.0
libdb	5.3.28	5.3.28
libdhash	0.5.0	
libdnf	0.69.0	0.69.0
libeconf	0.4.0	0.4.0
libedit	3.1	3.1
libev	4.33	4.33
libevent	2.1.12	2.1.12
libfdisk	2.37.4	2.37.4

Package	AMI	VMware OVA
libffi	3.4.4	3.4.4
libfido2	1.10.0	1.10.0
libgcc	11.4.1	11.4.1
libgcrypt	1.10.2	1.10.2
libgomp	11.4.1	11.4.1
libgpg-error	1.42	1.42
libibverbs	48.0	48.0
libidn2	2.3.2	2.3.2
libini_config	1.3.1	1.3.1
libkcapi	1.4.0	1.4.0
libkcapi-hmacalc	1.4.0	1.4.0
libldb	2.6.2	
libmaxminddb	1.5.2	1.5.2
libmetalink	0.1.3	0.1.3
libmnl	1.0.4	1.0.4
libmodulemd	2.13.0	2.13.0
libmount	2.37.4	2.37.4
libmspack		0.10.1
libnfsidmap	2.5.4	2.5.4
libnghttp2	1.59.0	1.59.0

Package	AMI	VMware OVA
libnl3	3.5.0	3.5.0
libpath_utils	0.2.1	0.2.1
libpcap	1.10.1	1.10.1
libpipeline	1.5.3	1.5.3
libpkgconf	1.8.0	1.8.0
libpsl	0.21.1	0.21.1
libpwquality	1.4.4	1.4.4
libref_array	0.1.5	0.1.5
librepo	1.14.5	1.14.5
libreport-filesystem	2.15.2	2.15.2
libseccomp	2.5.3	2.5.3
libselinux	3.4	3.4
libselinux-utils	3.4	3.4
libsemanage	3.4	3.4
libsepol	3.4	3.4
libsigsegv	2.13	2.13
libsmartcols	2.37.4	2.37.4
libsolv	0.7.22	0.7.22
libss	1.46.5	1.46.5
libsss_certmap	2.9.4	

Package	AMI	VMware OVA
libsss_idmap	2.9.4	2.9.4
libsss_nss_idmap	2.9.4	2.9.4
libsss_sudo	2.9.4	
libstdc++	11.4.1	11.4.1
libstoragemgmt	1.9.4	1.9.4
libtalloc	2.3.4	
libtasn1	4.19.0	4.19.0
libtdb	1.4.7	
libtevent	0.13.0	
libtextstyle	0.21	0.21
libtirpc	1.3.3	1.3.3
libtool-ltdl		2.4.7
libunistring	0.9.10	0.9.10
libuser	0.63	0.63
libutempter	1.2.1	1.2.1
libuuid	2.37.4	2.37.4
libuv	1.47.0	1.47.0
libverto	0.3.2	0.3.2
libverto-libev	0.3.2	0.3.2
libxcrypt	4.4.33	4.4.33

Package	AMI	VMware OVA
libxml2	2.10.4	2.10.4
libxslt		1.1.34
libyaml	0.2.5	0.2.5
libzstd	1.5.5	1.5.5
linux-firmware-whe nce	20210208	20210208
lm_sensors-libs	3.6.0	3.6.0
lmdb-libs	0.9.29	0.9.29
logrotate	3.20.1	3.20.1
lsof	4.94.0	4.94.0
lua-libs	5.4.4	5.4.4
lua-srpm-macros	1	1
lz4-libs	1.9.4	1.9.4
man-db	2.9.3	2.9.3
man-pages	5.10	5.10
microcode_ctl	2.1	2.1
mpfr	4.1.0	4.1.0
nano	5.8	5.8
ncurses	6.2	6.2
ncurses-base	6.2	6.2
ncurses-libs	6.2	6.2

Package	AMI	VMware OVA
nettle	3.8	3.8
net-tools	2.0	2.0
newt	0.52.21	0.52.21
nfs-utils	2.5.4	2.5.4
npth	1.6	1.6
nspr	4.35.0	4.35.0
nss	3.90.0	3.90.0
nss-softokn	3.90.0	3.90.0
nss-softokn-freebl	3.90.0	3.90.0
nss-sysinit	3.90.0	3.90.0
nss-util	3.90.0	3.90.0
ntsysv	1.15	1.15
numactl-libs	2.0.14	2.0.14
ocaml-srpm-macros	6	6
oniguruma	6.9.7.1	
openblas-srpm-macros	2	2
openldap	2.4.57	2.4.57
openssh	8.7p1	8.7p1
openssh-clients	8.7p1	8.7p1
openssh-server	8.7p1	8.7p1

Package	AMI	VMware OVA
openssl	3.0.8	3.0.8
openssl-lib	3.0.8	3.0.8
openssl-pkcs11	0.4.12	0.4.12
open-vm-tools		12.3.0
os-prober	1.77	1.77
p11-kit	0.24.1	0.24.1
p11-kit-trust	0.24.1	0.24.1
package-notes-srpm-macros	0.4	0.4
pam	1.5.1	1.5.1
parted	3.4	3.4
passwd	0.80	0.80
pciutils	3.7.0	3.7.0
pciutils-lib	3.7.0	3.7.0
pcre2	10.40	10.40
pcre2-syntax	10.40	10.40
perl-Carp	1.50	1.50
perl-Class-Struct	0.66	0.66
perl-constant	1.33	1.33
perl-DynaLoader	1.47	1.47
perl-Encode	3.15	3.15

Package	AMI	VMware OVA
perl-Errno	1.30	1.30
perl-Exporter	5.74	5.74
perl-Fcntl	1.13	1.13
perl-File-Basename	2.85	2.85
perl-File-Path	2.18	2.18
perl-File-stat	1.09	1.09
perl-File-Temp	0.231.100	0.231.100
perl-Getopt-Long	2.52	2.52
perl-Getopt-Std	1.12	1.12
perl-HTTP-Tiny	0.078	0.078
perl-if	0.60.800	0.60.800
perl-interpreter	5.32.1	5.32.1
perl-IO	1.43	1.43
perl-IPC-Open3	1.21	1.21
perl-libs	5.32.1	5.32.1
perl-MIME-Base64	3.16	3.16
perl-mro	1.23	1.23
perl-overload	1.31	1.31
perl-overloading	0.02	0.02
perl-parent	0.238	0.238

Package	AMI	VMware OVA
perl-PathTools	3.78	3.78
perl-Pod-Escapes	1.07	1.07
perl-podlators	4.14	4.14
perl-Pod-Perldoc	3.28.01	3.28.01
perl-Pod-Simple	3.42	3.42
perl-Pod-Usage	2.01	2.01
perl-POSIX	1.94	1.94
perl-Scalar-List-Utils	1.56	1.56
perl-SelectSaver	1.02	1.02
perl-Socket	2.032	2.032
perl-srpm-macros	1	1
perl-Storable	3.21	3.21
perl-subs	1.03	1.03
perl-Symbol	1.08	1.08
perl-Term-ANSIColor	5.01	5.01
perl-Term-Cap	1.17	1.17
perl-Text-ParseWords	3.30	3.30
perl-Text-Tabs+Wrap	2021.0726	2021.0726
perl-Time-Local	1.300	1.300
perl-vars	1.05	1.05

Package	AMI	VMware OVA
pkgconf	1.8.0	1.8.0
pkgconf-m4	1.8.0	1.8.0
pkgconf-pkg-config	1.8.0	1.8.0
policycoreutils	3.4	3.4
policycoreutils-python-utils	3.4	
popt	1.18	1.18
procps-ng	3.3.17	3.3.17
protobuf-c	1.4.1	1.4.1
psacct	6.6.4	6.6.4
psmisc	23.4	23.4
publicsuffix-list-dafsa	20240212	20240212
python3	3.9.16	3.9.16
python3-attrs	20.3.0	20.3.0
python3-audit	3.0.6	3.0.6
python3-awscli	0.19.19	0.19.19
python3-babel	2.9.1	2.9.1
python3-cffi	1.14.5	1.14.5
python3-chardet	4.0.0	4.0.0
python3-colorama	0.4.4	0.4.4

Package	AMI	VMware OVA
python3-configobj	5.0.6	5.0.6
python3-cryptography	36.0.1	36.0.1
python3-daemon	2.3.0	
python3-dateutil	2.8.1	2.8.1
python3-dbus	1.2.18	1.2.18
python3-distro	1.5.0	1.5.0
python3-dnf	4.14.0	4.14.0
python3-dnf-plugins-core	4.3.0	4.3.0
python3-docutils	0.16	0.16
python3-gpg	1.15.1	1.15.1
python3-hawkey	0.69.0	0.69.0
python3-idna	2.10	2.10
python3-jinja2	2.11.3	2.11.3
python3-jmespath	0.10.0	0.10.0
python3-jsonpatch	1.21	1.21
python3-jsonpointer	2.0	2.0
python3-jsonschema	3.2.0	3.2.0
python3-libcomps	0.1.20	0.1.20
python3-libdnf	0.69.0	0.69.0
python3-libs	3.9.16	3.9.16

Package	AMI	VMware OVA
python3-libselenium	3.4	3.4
python3-libsemanage	3.4	3.4
python3-libstorage mgmt	1.9.4	1.9.4
python3-lockfile	0.12.2	
python3-markupsafe	1.1.1	1.1.1
python3-netifaces	0.10.6	0.10.6
python3-oauthlib	3.0.2	3.0.2
python3-pip-wheel	21.3.1	21.3.1
python3-ply	3.11	3.11
python3-policycore utils	3.4	3.4
python3-prettytable	0.7.2	0.7.2
python3-prompt-toolkit	3.0.24	3.0.24
python3-pycparser	2.20	2.20
python3-pyrsistent	0.17.3	0.17.3
python3-pyserial	3.4	3.4
python3-pysocks	1.7.1	1.7.1
python3-pytz	2022.7.1	2022.7.1
python3-pyyaml	5.4.1	5.4.1

Package	AMI	VMware OVA
python3-requests	2.25.1	2.25.1
python3-rpm	4.16.1.3	4.16.1.3
python3-ruamel-yaml	0.16.6	0.16.6
python3-ruamel-yaml-clib	0.1.2	0.1.2
python3-setools	4.4.1	4.4.1
python3-setuptools	59.6.0	59.6.0
python3-setuptools-wheel	59.6.0	59.6.0
python3-six	1.15.0	1.15.0
python3-systemd	235	235
python3-urllib3	1.25.10	1.25.10
python3-wcwidth	0.2.5	0.2.5
python-chevron	0.13.1	
python-srpm-macros	3.9	3.9
quota	4.06	4.06
quota-nls	4.06	4.06
readline	8.1	8.1
rng-tools	6.14	6.14
rootfiles	8.1	8.1
rpcbind	1.2.6	1.2.6

Package	AMI	VMware OVA
rpm	4.16.1.3	4.16.1.3
rpm-build-lib	4.16.1.3	4.16.1.3
rpm-lib	4.16.1.3	4.16.1.3
rpm-plugin-selinux	4.16.1.3	4.16.1.3
rpm-plugin-systemd-inhibit	4.16.1.3	4.16.1.3
rpm-sign-lib	4.16.1.3	4.16.1.3
rsync	3.2.6	3.2.6
rust-srpm-macros	21	21
sbsigntools	0.9.4	0.9.4
screen	4.8.0	4.8.0
sed	4.8	4.8
selinux-policy	38.1.45	38.1.45
selinux-policy-targeted	38.1.45	38.1.45
setup	2.13.7	2.13.7
shadow-utils	4.9	4.9
slang	2.3.2	2.3.2
sqlite-lib	3.40.0	3.40.0
sssd-client	2.9.4	2.9.4
sssd-common	2.9.4	

Package	AMI	VMware OVA
sssd-kcm	2.9.4	
sssd-nfs-idmap	2.9.4	
strace	6.8	6.8
sudo	1.9.15	1.9.15
sysctl-defaults	1.0	1.0
sysstat	12.5.6	12.5.6
systemd	252.23	252.23
systemd-libs	252.23	252.23
systemd-networkd	252.23	252.23
systemd-pam	252.23	252.23
systemd-resolved	252.23	252.23
systemd-udev	252.23	252.23
system-release	2023.6.20241031	2023.6.20241031
systemtap-runtime	4.8	4.8
tar	1.34	1.34
tbb	2020.3	2020.3
tcpdump	4.99.1	4.99.1
tcsh	6.24.07	6.24.07
time	1.9	1.9
traceroute	2.1.3	2.1.3

Package	AMI	VMware OVA
tzdata	2024a	2024a
unzip	6.0	6.0
update-motd	2.2	2.2
userspace-rcu	0.12.1	0.12.1
util-linux	2.37.4	2.37.4
util-linux-core	2.37.4	2.37.4
vim-common	9.0.2153	9.0.2153
vim-data	9.0.2153	9.0.2153
vim-enhanced	9.0.2153	9.0.2153
vim-filesystem	9.0.2153	9.0.2153
vim-minimal	9.0.2153	9.0.2153
wget	1.21.3	1.21.3
which	2.21	2.21
words	3.0	3.0
xfsdump	3.1.11	3.1.11
xfspgrog	5.18.0	5.18.0
xmlsec1		1.2.33
xmlsec1-openssl		1.2.33
xxd	9.0.2153	9.0.2153
xxhash-libs	0.8.0	0.8.0

Package	AMI	VMware OVA
xz	5.2.5	5.2.5
xz-libs	5.2.5	5.2.5
yum	4.14.0	4.14.0
zip	3.0	3.0
zlib	1.2.11	1.2.11
zram-generator	1.1.2	
zram-generator-defaults	1.1.2	
zstd	1.5.5	1.5.5

Comparing packages installed on Amazon Linux 2023 standard AMI with the AL2023 Hyper-V image

A comparison of the RPMs present on the AL2023 standard AMI compared with the RPMs present on the AL2023 Hyper-V image.

Package	AMI	Hyper-V VHDX
acl	2.3.1	2.3.1
acpid	2.0.32	
alternatives	1.15	1.15
amazon-chrony-config	4.3	
amazon-ec2-net-utils	2.4.1	
amazon-linux-onprem		1.2

Package	AMI	Hyper-V VHDX
amazon-linux-repo-cdn		2023.4.20240319
amazon-linux-repo-s3	2023.4.20240319	
amazon-linux-sb-keys	2023.1	2023.1
amazon-onprem-netw ork		1.2
amazon-rpm-config	228	228
amazon-ssm-agent	3.2.2303.0	3.2.2303.0
at	3.1.23	3.1.23
attr	2.5.1	2.5.1
audit	3.0.6	3.0.6
audit-libs	3.0.6	3.0.6
aws-cfn-bootstrap	2.0	
awscli-2	2.14.5	2.14.5
basesystem	11	11
bash	5.2.15	5.2.15
bash-completion	2.11	2.11
bc	1.07.1	1.07.1
bind-libs	9.16.48	9.16.48
bind-license	9.16.48	9.16.48
bind-utils	9.16.48	9.16.48

Package	AMI	Hyper-V VHDX
binutils	2.39	2.39
boost-filesystem	1.75.0	1.75.0
boost-system	1.75.0	1.75.0
boost-thread	1.75.0	1.75.0
bzip2	1.0.8	1.0.8
bzip2-libs	1.0.8	1.0.8
ca-certificates	2023.2.64	2023.2.64
c-ares	1.19.0	
checkpolicy	3.4	3.4
chkconfig	1.15	1.15
chrony	4.3	4.3
cloud-init	22.2.2	22.2.2
cloud-init-cfg-ec2	22.2.2	
cloud-init-cfg-onp rem		22.2.2
cloud-utils-growpart	0.31	0.31
coreutils	8.32	8.32
coreutils-common	8.32	8.32
cpio	2.13	2.13
cracklib	2.9.6	2.9.6
cracklib-dicts	2.9.6	2.9.6

Package	AMI	Hyper-V VHDX
crontabs	1.11	1.11
crypto-policies	20220428	20220428
crypto-policies-scripts	20220428	20220428
cryptsetup	2.6.1	2.6.1
cryptsetup-libs	2.6.1	2.6.1
curl-minimal	8.5.0	8.5.0
cyrus-sasl-lib	2.1.27	2.1.27
cyrus-sasl-plain	2.1.27	2.1.27
dbus	1.12.28	1.12.28
dbus-broker	32	32
dbus-common	1.12.28	1.12.28
dbus-libs	1.12.28	1.12.28
device-mapper	1.02.185	1.02.185
device-mapper-libs	1.02.185	1.02.185
diffutils	3.8	3.8
dnf	4.14.0	4.14.0
dnf-data	4.14.0	4.14.0
dnf-plugin-release-notification	1.2	1.2
dnf-plugins-core	4.3.0	4.3.0

Package	AMI	Hyper-V VHDX
dnf-plugin-support-info	1.2	1.2
dnf-utils	4.3.0	4.3.0
dosfstools	4.2	4.2
dracut	055	055
dracut-config-ec2	3.0	
dracut-config-generic	055	055
dwz	0.14	0.14
dyninst	10.2.1	10.2.1
e2fsprogs	1.46.5	1.46.5
e2fsprogs-libs	1.46.5	1.46.5
ec2-hibinit-agent	1.0.8	
ec2-instance-connect	1.1	
ec2-instance-connect-selinux	1.1	
ec2-utils	2.2.0	
ed	1.14.2	1.14.2
efi-filesystem	5	5
efi-srpm-macros	5	5
efivar	38	38

Package	AMI	Hyper-V VHDX
efivar-libs	38	38
elfutils-debuginfod-client	0.188	0.188
elfutils-default-yama-scope	0.188	0.188
elfutils-libelf	0.188	0.188
elfutils-libs	0.188	0.188
ethtool	5.15	5.15
expat	2.5.0	2.5.0
file	5.39	5.39
file-libs	5.39	5.39
filesystem	3.14	3.14
findutils	4.8.0	4.8.0
fonts-srpm-macros	2.0.5	2.0.5
fstrm	0.6.1	0.6.1
fuse-libs	2.9.9	2.9.9
gawk	5.1.0	5.1.0
gdbm-libs	1.19	1.19
gdisk	1.0.8	1.0.8
gettext	0.21	0.21
gettext-libs	0.21	0.21

Package	AMI	Hyper-V VHDX
ghc-srpm-macros	1.5.0	1.5.0
glib2	2.74.7	2.74.7
glibc	2.34	2.34
glibc-all-langpacks	2.34	2.34
glibc-common	2.34	2.34
glibc-gconv-extra	2.34	2.34
glibc-locale-source	2.34	2.34
gmp	6.2.1	6.2.1
gnupg2-minimal	2.3.7	2.3.7
gnutls	3.8.0	3.8.0
go-srpm-macros	3.2.0	3.2.0
gpgme	1.15.1	1.15.1
gpm-libs	1.20.7	1.20.7
grep	3.8	3.8
groff-base	1.22.4	1.22.4
grub2-common	2.06	2.06
grub2-efi-x64-ec2	2.06	2.06
grub2-pc		2.06
grub2-pc-modules	2.06	2.06
grub2-tools	2.06	2.06

Package	AMI	Hyper-V VHDX
grub2-tools-minimal	2.06	2.06
grubby	8.40	8.40
gssproxy	0.8.4	0.8.4
gzip	1.12	1.12
hostname	3.23	3.23
hunspell	1.7.0	1.7.0
hunspell-en	0.20140811.1	0.20140811.1
hunspell-en-GB	0.20140811.1	0.20140811.1
hunspell-en-US	0.20140811.1	0.20140811.1
hunspell-filesystem	1.7.0	1.7.0
hwdata	0.353	0.353
hyperv-daemons		0
hyperv-daemons-lic ense		0
hypervfcopyd		0
hypervkvpd		0
hyperv-tools		0
hypervvssd		0
info	6.7	6.7
inih	49	49
initscripts	10.09	10.09

Package	AMI	Hyper-V VHDX
iproute	5.10.0	5.10.0
iputils	20210202	20210202
irqbalance	1.9.0	1.9.0
jansson	2.14	2.14
jitterentropy	3.4.1	3.4.1
jq	1.7.1	1.7.1
json-c	0.14	0.14
kbd	2.4.0	2.4.0
kbd-misc	2.4.0	2.4.0
kernel	6.1.79	6.1.79
kernel-livepatch-r epo-cdn		2023.4.20240319
kernel-livepatch-r epo-s3	2023.4.20240319	
kernel-modules-extra		6.1.79
kernel-modules-ext ra-common		6.1.79
kernel-srpm-macros	1.0	1.0
kernel-tools	6.1.79	6.1.79
keyutils	1.6.3	1.6.3
keyutils-libs	1.6.3	1.6.3

Package	AMI	Hyper-V VHDX
kmod	29	29
kmod-libs	29	29
kpatch-runtime	0.9.7	0.9.7
krb5-libs	1.21	1.21
less	608	608
libacl	2.3.1	2.3.1
libaio	0.3.111	0.3.111
libarchive	3.5.3	3.5.3
libargon2	20171227	20171227
libassuan	2.5.5	2.5.5
libattr	2.5.1	2.5.1
libbasicobjects	0.1.1	0.1.1
libblkid	2.37.4	2.37.4
libcap	2.48	2.48
libcap-ng	0.8.2	0.8.2
libcbor	0.7.0	0.7.0
libcollection	0.7.0	0.7.0
libcom_err	1.46.5	1.46.5
libcomps	0.1.20	0.1.20
libconfig	1.7.2	1.7.2

Package	AMI	Hyper-V VHDX
<u>libcurl-minimal</u>	8.5.0	8.5.0
<u>libdb</u>	5.3.28	5.3.28
libdhash	0.5.0	
libdnf	0.69.0	0.69.0
libeconf	0.4.0	0.4.0
libedit	3.1	3.1
libev	4.33	4.33
libevent	2.1.12	2.1.12
libfdisk	2.37.4	2.37.4
libffi	3.4.4	3.4.4
libfido2	1.10.0	1.10.0
libgcc	11.4.1	11.4.1
libgcrypt	1.10.2	1.10.2
libgomp	11.4.1	11.4.1
libgpg-error	1.42	1.42
libibverbs	48.0	48.0
libidn2	2.3.2	2.3.2
libini_config	1.3.1	1.3.1
libkcapi	1.4.0	1.4.0
libkcapi-hmaccalc	1.4.0	1.4.0

Package	AMI	Hyper-V VHDX
libldb	2.6.2	
libmaxminddb	1.5.2	1.5.2
libmetalink	0.1.3	0.1.3
libmnl	1.0.4	1.0.4
libmodulemd	2.13.0	2.13.0
libmount	2.37.4	2.37.4
libnfsidmap	2.5.4	2.5.4
libnghttp2	1.57.0	1.57.0
libnl3	3.5.0	3.5.0
libpath_utils	0.2.1	0.2.1
libpcap	1.10.1	1.10.1
libpipeline	1.5.3	1.5.3
libpkgconf	1.8.0	1.8.0
libpsl	0.21.1	0.21.1
libpwquality	1.4.4	1.4.4
libref_array	0.1.5	0.1.5
librepo	1.14.5	1.14.5
libreport-filessystem	2.15.2	2.15.2
libseccomp	2.5.3	2.5.3
libselinux	3.4	3.4

Package	AMI	Hyper-V VHDX
libselinux-utils	3.4	3.4
libsemanage	3.4	3.4
libsepol	3.4	3.4
libsigsegv	2.13	2.13
libsmartcols	2.37.4	2.37.4
libsolv	0.7.22	0.7.22
libss	1.46.5	1.46.5
libsss_certmap	2.9.4	
libsss_idmap	2.9.4	2.9.4
libsss_nss_idmap	2.9.4	2.9.4
libsss_sudo	2.9.4	
libstdc++	11.4.1	11.4.1
libstoragegmt	1.9.4	1.9.4
libtalloc	2.3.4	
libtasn1	4.19.0	4.19.0
libtdb	1.4.7	
libtevent	0.13.0	
libtextstyle	0.21	0.21
libtirpc	1.3.3	1.3.3
libunistring	0.9.10	0.9.10

Package	AMI	Hyper-V VHDX
libuser	0.63	0.63
libutempter	1.2.1	1.2.1
libuuid	2.37.4	2.37.4
libuv	1.47.0	1.47.0
libverto	0.3.2	0.3.2
libverto-libev	0.3.2	0.3.2
libxcrypt	4.4.33	4.4.33
libxml2	2.10.4	2.10.4
libyaml	0.2.5	0.2.5
libzstd	1.5.5	1.5.5
lm_sensors-libs	3.6.0	3.6.0
lmdb-libs	0.9.29	0.9.29
logrotate	3.20.1	3.20.1
lsof	4.94.0	4.94.0
lua-libs	5.4.4	5.4.4
lua-srpm-macros	1	1
lz4-libs	1.9.4	1.9.4
man-db	2.9.3	2.9.3
man-pages	5.10	5.10
microcode_ctl	2.1	2.1

Package	AMI	Hyper-V VHDX
mpfr	4.1.0	4.1.0
nano	5.8	5.8
ncurses	6.2	6.2
ncurses-base	6.2	6.2
ncurses-libs	6.2	6.2
nettle	3.8	3.8
net-tools	2.0	2.0
newt	0.52.21	0.52.21
nfs-utils	2.5.4	2.5.4
npth	1.6	1.6
nspr	4.35.0	4.35.0
nss	3.90.0	3.90.0
nss-softoken	3.90.0	3.90.0
nss-softoken-freebl	3.90.0	3.90.0
nss-sysinit	3.90.0	3.90.0
nss-util	3.90.0	3.90.0
ntsysv	1.15	1.15
numactl-libs	2.0.14	2.0.14
ocaml-srpm-macros	6	6
oniguruma	6.9.7.1	6.9.7.1

Package	AMI	Hyper-V VHDX
openblas-srpm-macros	2	2
openldap	2.4.57	2.4.57
openssh	8.7p1	8.7p1
openssh-clients	8.7p1	8.7p1
openssh-server	8.7p1	8.7p1
openssl	3.0.8	3.0.8
openssl-lib	3.0.8	3.0.8
openssl-pkcs11	0.4.12	0.4.12
os-prober	1.77	1.77
p11-kit	0.24.1	0.24.1
p11-kit-trust	0.24.1	0.24.1
package-notes-srpm-macros	0.4	0.4
pam	1.5.1	1.5.1
parted	3.4	3.4
passwd	0.80	0.80
pciutils	3.7.0	3.7.0
pciutils-lib	3.7.0	3.7.0
pcre2	10.40	10.40
pcre2-syntax	10.40	10.40
perl-Carp	1.50	1.50

Package	AMI	Hyper-V VHDX
perl-Class-Struct	0.66	0.66
perl-constant	1.33	1.33
perl-DynaLoader	1.47	1.47
perl-Encode	3.15	3.15
perl-Errno	1.30	1.30
perl-Exporter	5.74	5.74
perl-Fcntl	1.13	1.13
perl-File-Basename	2.85	2.85
perl-File-Path	2.18	2.18
perl-File-stat	1.09	1.09
perl-File-Temp	0.231.100	0.231.100
perl-Getopt-Long	2.52	2.52
perl-Getopt-Std	1.12	1.12
perl-HTTP-Tiny	0.078	0.078
perl-if	0.60.800	0.60.800
perl-interpreter	5.32.1	5.32.1
perl-I0	1.43	1.43
perl-IPC-Open3	1.21	1.21
perl-libs	5.32.1	5.32.1
perl-MIME-Base64	3.16	3.16

Package	AMI	Hyper-V VHDX
perl-mro	1.23	1.23
perl-overload	1.31	1.31
perl-overloading	0.02	0.02
perl-parent	0.238	0.238
perl-PathTools	3.78	3.78
perl-Pod-Escapes	1.07	1.07
perl-podlators	4.14	4.14
perl-Pod-Perldoc	3.28.01	3.28.01
perl-Pod-Simple	3.42	3.42
perl-Pod-Usage	2.01	2.01
perl-POSIX	1.94	1.94
perl-Scalar-List-Utils	1.56	1.56
perl-SelectSaver	1.02	1.02
perl-Socket	2.032	2.032
perl-srpm-macros	1	1
perl-Storable	3.21	3.21
perl-subs	1.03	1.03
perl-Symbol	1.08	1.08
perl-Term-ANSIColor	5.01	5.01
perl-Term-Cap	1.17	1.17

Package	AMI	Hyper-V VHDX
perl-Text-ParseWords	3.30	3.30
perl-Text-Tabs+Wrap	2021.0726	2021.0726
perl-Time-Local	1.300	1.300
perl-vars	1.05	1.05
pkgconf	1.8.0	1.8.0
pkgconf-m4	1.8.0	1.8.0
pkgconf-pkg-config	1.8.0	1.8.0
polycoreutils	3.4	3.4
polycoreutils-python-utils	3.4	
popt	1.18	1.18
procps-ng	3.3.17	3.3.17
protobuf-c	1.4.1	1.4.1
psacct	6.6.4	6.6.4
psmisc	23.4	23.4
publicsuffix-list-dafsa	20240212	20240212
python3	3.9.16	3.9.16
python3-attrs	20.3.0	20.3.0
python3-audit	3.0.6	3.0.6
python3-awscli	0.19.19	0.19.19

Package	AMI	Hyper-V VHDX
python3-babel	2.9.1	2.9.1
python3-cffi	1.14.5	1.14.5
python3-chardet	4.0.0	4.0.0
python3-colorama	0.4.4	0.4.4
python3-configobj	5.0.6	5.0.6
python3-cryptography	36.0.1	36.0.1
python3-daemon	2.3.0	
python3-dateutil	2.8.1	2.8.1
python3-dbus	1.2.18	1.2.18
python3-distro	1.5.0	1.5.0
python3-dnf	4.14.0	4.14.0
python3-dnf-plugins-core	4.3.0	4.3.0
python3-docutils	0.16	0.16
python3-gpg	1.15.1	1.15.1
python3-hawkey	0.69.0	0.69.0
python3-idna	2.10	2.10
python3-jinja2	2.11.3	2.11.3
python3-jmespath	0.10.0	0.10.0
python3-jsonpatch	1.21	1.21
python3-jsonpointer	2.0	2.0

Package	AMI	Hyper-V VHDX
python3-jsonschema	3.2.0	3.2.0
python3-libcomps	0.1.20	0.1.20
python3-libdnf	0.69.0	0.69.0
python3-libs	3.9.16	3.9.16
python3-libselenium	3.4	3.4
python3-libsemanage	3.4	3.4
python3-libstorage mgmt	1.9.4	1.9.4
python3-lockfile	0.12.2	
python3-markupsafe	1.1.1	1.1.1
python3-netifaces	0.10.6	0.10.6
python3-oauthlib	3.0.2	3.0.2
python3-pip-wheel	21.3.1	21.3.1
python3-ply	3.11	3.11
python3-policycore utils	3.4	3.4
python3-prettytable	0.7.2	0.7.2
python3-prompt-too lkit	3.0.24	3.0.24
python3-pycparser	2.20	2.20
python3-pyrsistent	0.17.3	0.17.3

Package	AMI	Hyper-V VHDX
python3-pyserial	3.4	3.4
python3-pysocks	1.7.1	1.7.1
python3-pytz	2022.7.1	2022.7.1
python3-pyyaml	5.4.1	5.4.1
python3-requests	2.25.1	2.25.1
python3-rpm	4.16.1.3	4.16.1.3
python3-ruamel-yaml	0.16.6	0.16.6
python3-ruamel-yaml-clib	0.1.2	0.1.2
python3-setools	4.4.1	4.4.1
python3-setuptools	59.6.0	59.6.0
python3-setuptools-wheel	59.6.0	59.6.0
python3-six	1.15.0	1.15.0
python3-systemd	235	235
python3-urllib3	1.25.10	1.25.10
python3-wcwidth	0.2.5	0.2.5
python-chevront	0.13.1	
python-srpm-macros	3.9	3.9
quota	4.06	4.06
quota-nls	4.06	4.06

Package	AMI	Hyper-V VHDX
readline	8.1	8.1
rng-tools	6.14	6.14
rootfiles	8.1	8.1
rpcbind	1.2.6	1.2.6
rpm	4.16.1.3	4.16.1.3
rpm-build-libs	4.16.1.3	4.16.1.3
rpm-libs	4.16.1.3	4.16.1.3
rpm-plugin-selinux	4.16.1.3	4.16.1.3
rpm-plugin-systemd-inhibit	4.16.1.3	4.16.1.3
rpm-sign-libs	4.16.1.3	4.16.1.3
rsync	3.2.6	3.2.6
rust-srpm-macros	21	21
sbsigntools	0.9.4	0.9.4
screen	4.8.0	4.8.0
sed	4.8	4.8
selinux-policy	37.22	37.22
selinux-policy-targeted	37.22	37.22
setup	2.13.7	2.13.7
shadow-utils	4.9	4.9

Package	AMI	Hyper-V VHDX
slang	2.3.2	2.3.2
sqlite-libs	3.40.0	3.40.0
sssd-client	2.9.4	2.9.4
sssd-common	2.9.4	
sssd-kcm	2.9.4	
sssd-nfs-idmap	2.9.4	
strace	5.16	5.16
sudo	1.9.14	1.9.14
sysctl-defaults	1.0	1.0
sysstat	12.5.6	12.5.6
systemd	252.16	252.16
systemd-libs	252.16	252.16
systemd-networkd	252.16	252.16
systemd-pam	252.16	252.16
systemd-resolved	252.16	252.16
systemd-udev	252.16	252.16
system-release	2023.4.20240319	2023.4.20240319
systemtap-runtime	4.8	4.8
tar	1.34	1.34
tbb	2020.3	2020.3

Package	AMI	Hyper-V VHDX
tcpdump	4.99.1	4.99.1
tcsh	6.24.07	6.24.07
time	1.9	1.9
traceroute	2.1.3	2.1.3
tzdata	2024a	2024a
unzip	6.0	6.0
update-motd	2.2	2.2
userspace-rcu	0.12.1	0.12.1
util-linux	2.37.4	2.37.4
util-linux-core	2.37.4	2.37.4
vim-common	9.0.2153	9.0.2153
vim-data	9.0.2153	9.0.2153
vim-enhanced	9.0.2153	9.0.2153
vim-filesystem	9.0.2153	9.0.2153
vim-minimal	9.0.2153	9.0.2153
wget	1.21.3	1.21.3
which	2.21	2.21
words	3.0	3.0
xfsdump	3.1.11	3.1.11
xfstools	5.18.0	5.18.0

Package	AMI	Hyper-V VHDX
xxd	9.0.2153	9.0.2153
xxhash-libs	0.8.0	0.8.0
xz	5.2.5	5.2.5
xz-libs	5.2.5	5.2.5
yum	4.14.0	4.14.0
zip	3.0	3.0
zlib	1.2.11	1.2.11
zram-generator	1.1.2	
zram-generator-def aults	1.1.2	
zstd	1.5.5	1.5.5

Identifying Amazon Linux instances and versions

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 will cover the different methods that can be used, their limitations, and go through some examples of their use.

Topics

- [Using the os-release standard](#)
- [Amazon Linux Specific](#)
- [Example code for OS detection](#)

Using the os-release standard

Amazon Linux complies with the [os-release standard](#) for identifying Linux distributions. This file provides machine-readable information about the operating system identification and version information.

Note

The standard dictates that `/etc/os-release` is attempted to be parsed first, followed by `/usr/lib/os-release`. Care should be taken to follow the standard around file names and paths.

Topics

- [Key identification differences](#)
- [Field types: Machine-readable vs. 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 an instance is running Amazon Linux is to check the `ID` field in `os-release`.

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

- Amazon Linux AMI: `VERSION_ID` contains a date-based version (e.g., `2018.03`)
- AL2: `VERSION_ID="2"`
- AL2023: `VERSION_ID="2023"`

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 vs. 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 (e.g., `2023`)

- `ID_LIKE` – Space-separated list of related distributions (e.g., `fedora`)
- `VERSION_CODENAME` – Release codename for scripts (e.g., `karoo`)
- `VARIANT_ID` – Variant identifier for programmatic decisions
- `BUILD_ID` – Build identifier for system images
- `IMAGE_ID` – Image identifier for containerized environments
- `PLATFORM_ID` – Platform identifier (e.g., `platform:a12023`)

Human-readable fields

These fields are intended for display to users and may contain spaces, mixed case, and descriptive text. They should be used when presenting operating system information in user interfaces.

- `NAME` – Operating system name for display (e.g., `Amazon Linux`)
- `PRETTY_NAME` – Full operating system name with version for display (e.g., `Amazon Linux 2023.8.20250721`)
- `VERSION` – Version information suitable for user presentation
- `VARIANT` – Variant or edition name for display (e.g., `Server Edition`)

Other informational fields

These fields provide additional metadata about the operating system:

- `HOME_URL` – Project homepage URL
- `DOCUMENTATION_URL` – Documentation URL
- `SUPPORT_URL` – Support information URL
- `BUG_REPORT_URL` – Bug reporting URL
- `VENDOR_NAME` – Vendor name
- `VENDOR_URL` – Vendor 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:

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 2023.8.20250721"
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="42 (Container Image)"  
RELEASE_TYPE=stable  
ID=fedora  
VERSION_ID=42  
VERSION_CODENAME=""  
PLATFORM_ID="platform:f42"  
PRETTY_NAME="Fedora Linux 42 (Container Image)"  
ANSI_COLOR="0;38;2;60;110;180"  
LOGO=fedora-logo-icon  
CPE_NAME="cpe:/o:fedoraproject:fedora:42"  
DEFAULT_HOSTNAME="fedora"  
HOME_URL="https://fedoraproject.org/"
```

```
DOCUMENTATION_URL="https://docs.fedoraproject.org/en-US/fedora/f42/system-  
administrators-guide/"  
SUPPORT_URL="https://ask.fedoraproject.org/"  
BUG_REPORT_URL="https://bugzilla.redhat.com/"  
REDHAT_BUGZILLA_PRODUCT="Fedora"  
REDHAT_BUGZILLA_PRODUCT_VERSION=42  
REDHAT_SUPPORT_PRODUCT="Fedora"  
REDHAT_SUPPORT_PRODUCT_VERSION=42  
SUPPORT_END=2026-05-13  
VARIANT="Container Image"  
VARIANT_ID=container
```

Debian

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

```
PRETTY_NAME="Debian GNU/Linux 12 (bookworm)"  
NAME="Debian GNU/Linux"  
VERSION_ID="12"  
VERSION="12 (bookworm)"  
VERSION_CODENAME=bookworm  
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 24.04.2 LTS"  
NAME="Ubuntu"  
VERSION_ID="24.04"  
VERSION="24.04.2 LTS (Noble Numbat)"  
VERSION_CODENAME=noble  
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=noble  
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: Amazon Linux uses `fedora (AL2023)` or `centos rhel fedora (AL2)`, while Ubuntu shows `debian` to indicate its Debian heritage
- **VERSION_ID** – Provides machine-parseable version information: `2023` for AL2023, `42` for Fedora, `12` for Debian, `24.04` 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 2023.8.20250721`, `Fedora Linux 42 (Container Image)`, `Debian GNU/Linux 12 (bookworm)`, `Ubuntu 24.04.2 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

There are some files that are specific to Amazon Linux that can be used for identifying Amazon Linux and what version it is. New code should use the [/etc/os-release](#) standard in order to be cross-distribution compatible. Use of any Amazon Linux specific files is discouraged.

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 using package managers and on Amazon Linux is part of the `system-release` package. While 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.

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.

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="x86_64"  
image_file="al2023-container-2023.8.20250721.2-x86_64"  
image_stamp="822b-1a9e"  
image_date="20250719211531"  
recipe_name="al2023 container"  
recipe_id="89b25f7b-be82-2215-a8eb-6e63-0830-94ea-658d41c4"
```

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

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

```
Amazon Linux release 2023.8.20250721 (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="x86_64"
```

```
image_file="amzn2-container-raw-2.0.20250721.2-x86_64"  
image_stamp="4126-16ad"  
image_date="20250721225801"  
recipe_name="amzn2 container"  
recipe_id="948422df-a4e6-5fc8-ba89-ef2e-0e1f-e1bb-16f84087"
```

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 script below is implemented in several different programming languages, and each implementation will produce 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"
        ;;
    *)
        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")
    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 (Python 3.10+)
    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")
    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";
    } 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 will produce 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 2023.8.20250721
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: 24.04

PRETTY_NAME: Ubuntu 24.04.2 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: 12
PRETTY_NAME: Debian GNU/Linux 12 (bookworm)
```

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: 42
PRETTY_NAME: Fedora Linux 42 (Container Image)
```

Filesystem Layout

This section covers the filesystem layout of an AL2023 system, including details that may be specific to instances or AL2023 based containers. See the `file-hierarchy(7)` man page for more information.

Topics

- [/ \(The root directory\)](#)
- [/boot \(Kernel, initramfs, etc.\)](#)
- [/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, AL2023 images are configured with a writable `/`, allowing privileged users to create new files and directories.

It is possible to configure `systemd` services to use a different path or image to appear as `/` for that service, as well as place access restrictions on any path.

Note

It is best practice for `systemd` services to be configured to restrict what the service has access to. This can include using the `ReadOnlyPaths=` directive which makes `/` read only for that service.

For more information on using `systemd` to restrict what access a service has to the system, see the `systemd.exec(5)` man page.

/boot (Kernel, initramfs, etc.)

By default, bootable AL2023 images are configured with `/boot` being on the root file system. The `/boot` path is only relevant for bootable images, so is unused in AL2023 container images.

This directory is home to files needed for AL2023 to boot such as the Linux kernel and initramfs. The content of this directory should only be manipulated using the tools provided with the OS.

/boot/efi (EFI System Partition)

By default, bootable AL2023 images are configured with the EFI System partition being mounted at `/boot/efi`. This file system is managed by the OS and contains code and configuration critical to booting the system.

This path is not relevant for container images.

/etc (System Configuration)

The `/etc` directory on AL2023 contains system-specific configuration. By default, AL2023 images come with `/etc` on the root filesystem and writable by privileged users.

Note

It is common for applications (including `systemd`) to keep default configuration under [/usr \(System Resources\)](#) which can be overridden by placing configuration in [/etc \(System Configuration\)](#).

For these applications, changing files in [/usr \(System Resources\)](#) rather than overriding default configuration in `/etc` will likely result in the changes being overridden when the package is updated.

/home (User home directories)

Normal users have their home directories under `/home`, but software should always look for the per-user `$HOME` environment variable rather than relying on a pattern such as `/home/$USER`.

By default, AL2023 images have `/home` on the root file system, but software should not rely on this. It is perfectly valid for the OS to be configured for `/home` to be a separate file system, which is mounted later during boot, or only after a user authenticates to the system.

The root user home directory is not in `/home` but rather is [/root \(root user home directory\)](#) so that it is available in the event that the `/home` file system cannot be mounted.

 **Note**

It is best practice for systemd services which do not need write access to `/home` to be configured with the `ProtectHome=read-only` directive. With this option, `/home`, `/root`, and `/run/user` are made read-only for the service.

It is also best practice for services that do not need any access to `/home` to be configured with the `ProtectHome=tmpfs` directive, which will run the service in a sandbox where `/home`, `/root`, and `/run/user` are empty read-only tmpfs file systems.

For more information on using systemd to restrict what access a service has to the system, see the `systemd.exec(5)` man page.

`/root` (root user home directory)

The home directory of the root user is the `/root` directory, purposefully separate from [/home \(User home directories\)](#) so that it is present in the event that [/home \(User home directories\)](#) is on a file system which is not available.

The best practice for configuring systemd services is the same for `/root` as it is for [/home \(User home directories\)](#).

`/srv` (Server Payload)

The `/srv` directory is managed by the administrator of the system and Amazon Linux 2023 places no restrictions on how this directory is organized.

It is possible to configure the `/srv` directory to be on a separate file system, so it may only become available later in boot.

/tmp (small temporary files)

Note

Amazon Linux 2023 is different to Amazon Linux 2 as by default /tmp is now tmpfs rather than a path on the root file system.

Note

When running in a container, it will typically be your container runtime configuration that dictates if /tmp is tmpfs, or a path on disk, and if there is a running clean-up process or not.

The /tmp directory is for small, size-bounded temporary files. By default, AL2023 configures it to be a tmpfs file system with a size limit of 50% of RAM and a maximum of one million inodes.

Applications should prefer the path in the \$TMPDIR environment variable over /tmp. Users can then set the \$TMPDIR environment variable to override the path an application should use for /tmp

For larger temporary files, [/var/tmp](#) should be used instead.

Warning

Since /tmp is shared, it is important to use safe methods of creating temporary files. For details, see the upstream systemd documentation on [Using /tmp and /var/tmp Safely](#).

Note

It is best practice for systemd services to be configured with the PrivateTmp= directive set to yes or disconnected which runs the service in a sandbox where /tmp and [/var/tmp](#) are not shared with the host or other services.

For more information, including how to configure two services to share the same private temporary directories, see the `systemd.exec(5)` man page.

The content of `/tmp` is typically cleaned at boot time, and unused files are regularly cleaned up. By default, the cleanup process runs shortly after boot and then every day. For information on how to configure the clean-up of temporary files, see the `tmpfiles.d(5)` and `systemd-tmpfiles(8)` man pages.

The `/tmp` and [/var/tmp](#) paths are closely related and exist for different purposes.

/run (runtime data)

The `/run` directory is used by system packages to store small amounts of runtime data (such as socket files). It is a `tmpfs` file system, and is writable only by privileged programs.

The `/run/log` directory can be used by system components to store logs in, either before being written out to `/var/log` or before the `/var/log` file system is available.

The `/run/user/` path contains per-user runtime directories. By default, these will be individual `tmpfs` file systems mounted by `systemd` when the user logs in, and erased when the user is no longer logged in. As per the [XDG Base Directory Specification](#), these paths should not be referenced directly but rather via the `$XDG_RUNTIME_DIR` environment variable.

/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 applications must assume that `/usr` can be read-only. The `/usr` hierarchy must not be used for volatile data. Except for [/usr/local](#), the `/usr` hierarchy must not be used for any data that is added or changed outside of package installation/removal as done by the OS package manager. The OS package manager may assume that all of the `/usr` hierarchy (except [/usr/local](#)) is the same mountpoint.

Software being installed outside of the OS package manager should not store data in `/usr` as this may impede any future invocation of the OS package manager. The [/usr/local](#) hierarchy is the exception, and is reserved for software outside of the OS package manager.

/usr/bin (Executables)

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

/usr/include (C/C++ Headers)

The `/usr/include` directory contains C and C++ header files, usually contained in packages with the `-devel` suffix.

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

On Amazon Linux 2023, the `/usr/lib64` path is used for 64-bit shared libraries, and package data which is architecture dependent. Since AL2023 does not ship with any 32-bit userspace support, there are only 64-bit shared libraries available.

The `/usr/lib` path is for static data from OS packages which is compatible with all architectures. This may include executables not usually invoked from a shell, which may also be found in `/usr/libexec`. Shared libraries are found in `/usr/lib64` rather than `/usr/lib`.

/usr/local (System administrator installed software)

On Amazon Linux 2023, the `/usr/local` path is available for the system administrator to install software in, software that is not owned by the OS, and will not be touched by the OS. The default `/usr/local` hierarchy mirrors the `/` hierarchy.

/usr/share (Shared resources)

Shared resources such as documentation, fonts, and time zone data live in `/usr/share`. It is common for various specifications to dictate exactly where and in what format data is stored in this directory.

/usr/share/doc (Documentation)

Documentation that comes with packages will be stored in `/usr/share/doc`.

/var (Persistent Variable System Data)

/var/cache (Cache)

In contrast to [/var/lib](#), erasing data in `/var/cache` will not result in data loss, as applications are required to be able to rebuild their `/var/cache` data from other sources.

/var/lib (Persistent system data)

The `/var/lib` directory is used for persistent system data. Various system components will place data here which is private to that component. In contrast to [/var/cache](#), erasing data in `/var/lib` will result in data loss.

For example, the PostgreSQL database server will by default store database data in `/var/lib/pgsql`. The layout and file formats of this data is private to PostgreSQL, and it is persistent data as if erased, the user experiences data loss.

/var/log (Persistent logs)

This directory is used for storing persistent logs. It is recommended that software use the `syslog(3)` or `sd_journal_print(3)` API calls rather than directly storing log files under `/var/log`.

Note

In AL2023 the [systemd journal replaces rsyslog](#), which is a notable difference from the default Amazon Linux 2 configuration.

For more information on reading logs using `journalctl`, see the [journalctl](#) manual page.

Many applications use their own mechanisms for writing, and sometimes rotating, log files found in `/var/log`. See the documentation for these applications on how to configure their log files.

/var/spool (Mail and Printer queues)

This directory is used for persistent data such as mail or printer queues.

/var/tmp (larger temporary files)

For small, size-bound temporary files, [/tmp](#) should possibly be used instead.

While [/tmp](#) is by default configured to be a tmpfs volume, `/var/tmp` is by default configured to be a path on the root file system, and is thus the place for larger and more persistent temporary files. By default, there is a cleanup job run on a regular schedule which removes files not recently accessed.

For information on how to configure the clean-up of temporary files, see the `tmpfiles.d(5)` and `systemd-tmpfiles(8)` man pages.

As with [/tmp](#), applications should prefer the path specified in the `$TMPDIR` environment variable over `/var/tmp`. Users can then set the `$TMPDIR` environment variable to override the path an application should use for `/var/tmp`.

Warning

Since `/var/tmp` is shared (as is [/tmp](#)), it is important to use safe methods of creating temporary files. For details, see the upstream `systemd` documentation on [Using /tmp and /var/tmp Safely](#).

Note

It is best practice for `systemd` services to be configured with the `PrivateTmp=` directive set to `yes` or `disconnected` which runs the service in a sandbox where [/tmp](#) and [/var/tmp](#) are not shared with the host or other services. For more information, including how to configure two services to share the same private temporary directories, see the `systemd.exec(5)` man page.

The [/tmp](#) and [/var/tmp](#) paths are closely related and exist for different purposes.

Updating AL2023

It's important to keep up to date with AL2023 releases so that you can benefit from security updates and new features. With AL2023, you can ensure consistency between package versions and updates across your environment through [Deterministic upgrades through versioned repositories on AL2023](#).

Warning

Running `dnf --releasever=latest update` is not best practice, and is likely to result in an OS update being first tested in production.

Instead of using `latest`, use a specific AL2023 release version. This ensures you are deploying the same changes across production instances as you previously tested. For example, `dnf --releasever=2023.9.20251117 update` will always update to the 2023.9.20251117 release.

For more information, see the [Updating AL2023](#) section in the [AL2023 User Guide](#).

Topics

- [Best practices for safely deploying updates](#)
- [Receive notifications on new updates](#)
- [Deterministic upgrades through versioned repositories on AL2023](#)
- [Manage package and operating system updates in AL2023](#)
- [Kernel Live Patching on AL2023](#)
- [Updating the Linux kernel on AL2023](#)

Best practices for safely deploying updates

Amazon Linux 2023 (AL2023) has several features designed to aid in safely deploying updates to the Operating System, and being able to know what changed between updates, and if necessary, easily revert to the older version. This section explores lessons learned by AWS from more than a decade of internal and external use of Amazon Linux.

⚠ Warning

Running `dnf --releasever=latest update` is not best practice, and is likely to result in an OS update being first tested in production.

Instead of using `latest`, use a specific AL2023 release version. This ensures you are deploying the same changes across production instances as you previously tested. For example, `dnf --releasever=2023.9.20251117 update` will always update to the 2023.9.20251117 release.

For more information, see the [Updating AL2023](#) section in the [AL2023 User Guide](#).

Without planning for deployment safety of OS updates, the impact of an unexpected negative interaction between your application/service and an OS update can be significantly greater, up to and including a total outage. As with any software issue, the earlier the issue is detected, the less impact it can have on end users.

It is important to not fall into the trap of believing two things which 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 behavior and interfaces that the OS vendor would consider something to be relied upon.

i.e. both the OS vendor and you would agree that there was a problem with the update.

Do not rely on good intentions, put systems in place to ensure that deployment safety *includes* any update to the OS.

It is not recommended to test new OS updates by deploying to production environments. It is best practice to consider the OS as another part of your deployment, and think about applying the same deployment safety mechanisms you consider suitable for any other change to a production environment.

It is best practice to test any and all OS updates before deploying to production systems. When deploying, staged rollouts combined with good monitoring are recommended. Staged rollouts can ensure that if a problem occurs, even if not immediate, impact is restricted to a subset of a fleet, and further deployment of the update can be halted while further investigation and mitigation can occur.

The mitigation of any negative impact of taking an update to the OS is often the first priority, followed by resolving the issue, wherever it may be. Where the introduction of an OS update is correlated to negative impact, the ability to revert to the previous known-good version of the OS is a powerful tool to have.

Amazon Linux 2023 introduces [Deterministic upgrades through versioned repositories](#), a powerful new feature to ensure any change to the version of the OS (or individual packages) is repeatable. Thus, if a problem is encountered when moving from one OS version to the next, there are simple to use mechanisms available to stick to the known-working OS version while working out how to resolve the problem.

With AL2023, whenever we release new package updates, there's a new version to lock to, and new AMIs that lock to that version. The [AL2023 Release Notes](#) cover changes in each release, and [Amazon Linux Security advisories for AL2023](#) covers security issues addressed in package updates.

For example, if you were affected by the issue present in the [2023.6.20241028](#) release, you could immediately revert to using the AMIs and container images of the prior release, [2023.6.20241010](#). In this case, there was a bug in a package that was fixed in the subsequent [2023.6.20241031](#) release, but with [Deterministic upgrades through versioned repositories](#) anyone affected could *immediately* take simple action to mitigate: just use the previous images.

[Deterministic upgrades through versioned repositories](#) also gives assurance that any in-progress deployment of an OS update, either in place or by launching new AMIs or container images, are not affected by subsequently released OS updates.

For our first example, fleet A is a large fleet which is halfway through deploying the update from [2023.5.20241001](#) to the [2023.6.20241010](#) release when the [2023.6.20241028](#) release comes out. [Deterministic upgrades through versioned repositories](#) means that the deployment for fleet A continues without any change to what updates it is applying.

The purpose of wave based or phase based deployment strategies such as first deploying to 1% of a fleet, then 5%, 10%, 20%, 40%, until reaching 100%, is to be able to test a change in a limited fashion before rolling it out wider. This type of deployment strategy is commonly considered best practice for deploying any production change.

With a wave based deployment strategy and the fleet A update to [2023.6.20241010](#) being at a stage where it's being deployed to a lot of hosts at once, the fact that [2023.6.20241028](#) was released has *no impact on the in-progress deployment* thanks to using [Deterministic upgrades through versioned repositories](#).

If fleet B was running an older version, say [2023.5.20240708](#), and had started deploying the update to [2023.6.20241028](#), and fleet B was affected by the issue in that version, this would be noticed early in the deployment. At that point, a decision can be made on if to pause any rollout until a fix for that issue is available, or if in the meantime to start a deployment of the same version fleet A was running, [2023.6.20241010](#) so that fleet B gets all the updates between [2023.5.20240708](#) and [2023.6.20241010](#).

It is important to note that *not* taking OS updates promptly can cause issues. New updates likely contain bug and security fixes which may be relevant to your environment. For more information, see [Security and Compliance in Amazon Linux 2023](#) and [Manage package and operating system updates in AL2023](#).

It is important to configure your deployment systems to be able to easily take new OS updates, test them before deploying to production, and use mechanisms such as wave based deployments to minimize any negative impact. In order to be able to mitigate any negative impact of an OS update, it is important to know how to make your deployment systems point to a previous known-good version of the OS, and once the issue is addressed, no longer be locked to the older known-good version but rather move to a new known-good version.

Preparing for Minor Updates

Preparing for smaller updates to the OS, such as a new point release of AL2023 is intended to be limited to zero effort. Be sure to read the [AL2023 Release Notes](#) for any upcoming changes.

The [support period of a package](#) coming to an end may involve moving to a newer version of the language runtime (such as with [PHP in AL2023](#)). It is best practice to prepare for this in advance by moving to new language run time versions comfortably in advance of the support period ending.

For packages such as [pcre version 1](#), there is also the opportunity to plan in advance and migrate any of your code to its replacement, which in this case is pcre version 2. It is best practice to do so as soon as possible, to allow time for any setbacks.

Where there is no direct replacement, such as with [Berkeley DB \(libdb\)](#), you may need to make a choice based on your use case.

Preparing for Major Updates

Updating to a new major version of an Operating System is near universally viewed as something which requires planning, work to adapt to changed or deprecated functionality, and also testing prior to deployment. It is not uncommon to be able to prepare for the next major version of

Amazon Linux 2023 more incrementally, such as addressing any use of deprecated or removed functionality before proceeding with moving to the next major version.

For example, when moving from AL2 to AL2023, reading the [Functionality deprecated in AL2 and removed in AL2023](#) section can result in a number of safe and small steps which can happen while still using AL2 to prepare for AL2023. For example, any [Python 2.7 has been replaced with Python 3](#) usage (outside of OS use such as in the yum package manager) can be migrated to Python 3 in preparation for using [Python in AL2023](#). If using [PHP](#), both AL2 (through the PHP 8.2 [AL2 Extra](#)) and AL2023 ship PHP 8.2, and thus both PHP version migration and OS migration do not have to occur simultaneously.

While using AL2023, it is also possible to prepare for the next major version of Amazon Linux 2023 today, while using AL2023. The [Deprecated in AL2023](#) section covers features and packages which are deprecated in AL2023 and due to be removed.

For example, migrating any remaining [System V init \(sysvinit\)](#) use, such as `init` scripts over to their `systemd` equivalent will prepare you for the future, as well as allow you to use the full set of `systemd` features to monitor the service, how and if to restart it, what other services it needs, and if any resource or permission constraints should be applied.

For features such as 32-bit support, deprecation can span multiple major versions of the OS. For 32-bit, Amazon Linux 1 (AL1) deprecated [32-bit x86 \(i686\) AMIs](#), Amazon Linux 2 deprecated [32-bit x86 \(i686\) Packages](#), and Amazon Linux 2023 deprecates [32bit x86 \(i686\) runtime support](#). The transition away from [IMDSv1](#) also spans multiple major versions of the OS. For these types of changes, it is understood that some customers require a longer time to adapt to them, thus there is a large amount of leeway before the functionality is no longer available in Amazon Linux 2023.

The list of deprecated functionality is updated over the lifetime of the OS, and it is advisable to keep up to date with changes to it.

Receive notifications on new updates

You can receive notifications whenever a new AL2023 AMI is released. Notifications are published with [Amazon SNS](#) using the following topic.

```
arn:aws:sns:us-east-1:137112412989:amazon-linux-2023-ami-updates
```

Messages are posted here when a new AL2023 AMI is published. The version of the AMI will be included in the message.

These messages can be received using several different methods. We recommend that you use 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 must select the Region where the SNS notification that you're subscribing to was created.
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-2023-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 AL2023

Note

By default, your AL2023 instance doesn't automatically receive additional critical and important security updates at launch. Your instance initially contains the updates that were available in the version of AL2023 and the chosen AMI.

Control the updates received from major and minor releases

With AL2023, you can ensure consistency between package versions and updates across your environment. You can also ensure consistency for multiple instances of the same Amazon Machine Image (AMI). With the deterministic upgrades through versioned repositories feature, which is turned on by default, you can apply updates based on a schedule that meets your specific needs.

Whenever we release new package updates, there's a new version to lock to, and new AMIs that lock to that version.

AL2023 locks to a specific version of your repository. This is supported for both major or minor versions. The AL2023 AMI, exposed through our SSM parameters, is always the latest version. It has the most up-to-date packages and updates, including critical and important security updates.

If you launch an instance from an existing AMI, updates aren't automatically applied. Any additional packages that are installed as part of your provisioning map to the repository version of the existing AMI.

With this feature, you're in charge of ensuring consistency among package versions and updates across your environment. This is particularly the case if you're launching multiple instances from the same AMI. You can apply updates based on a schedule that meets your needs. You can also apply a specific set of updates on launch because these can also be locked to a specific repository version.

Differences between minor and major version upgrades

Major version releases of AL2023 include large-scale updates and might add, delete, or update packages. To ensure compatibility, upgrade your instance to a new major version only after you test your application on that version.

Minor version releases of AL2023 include feature and security updates, but don't include package changes. This ensures that Linux features and the system library API stay available on new versions. Testing your application before updating isn't necessary.

Knowing when updates are available

In order to apply an update, you need to know that one is available, and then know how to deploy the update.

For building derived AMIs when new AL2023 AMIs are released, [EC2 Image Builder](#) can automatically build, patch, and test AMIs. To trigger your own AMI building pipelines, or to use the base AMIs, you can [Receive notifications on new updates](#).

For patching in-place, you can use tools such as [AWS Systems Manager Patch Manager](#) to orchestrate applying updates across a fleet.

For other public AMIs based on AL2023, the providers of those AMIs may have their own release schedule and notification methods. When using derived AMIs or container images, check the documentation from the publisher as to when updates are released.

The changes in each release are documented in the [AL2023 release notes](#). Security updates are published on [Amazon Linux Security Center \(ALAS\)](#).

Control the package updates available from the AL2023 repositories

When we publish a new version of the AL2023 repositories, all previous versions are still available. By default, the plugin for managing repository versions locks to the same version that was used to build the AMI. If you want to control package updates, follow these steps.

1. Discover available repository versions by running the following command.

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

2. Select a version by running the following command.

```
$ sudo dnf upgrade --releasever=version
```

This command starts an update using dnf from your current Amazon Linux release version to the release version that's specified in the command line. A list of the package updates is presented by dnf. Before the update is processed, you must confirm the update. After the update is complete, the new release version becomes the default release version that dnf uses for all future activities.

For more information, see [Manage package and operating system updates in AL2023](#).

Deterministic updates via instance replacement

The [Deterministic upgrades through versioned repositories on AL2023](#) feature of Amazon Linux 2023 makes instance replacement an easy way to deterministically and safely roll out updated versions of AL2023. Deterministic updates mean that as a new version is progressively rolled out, if any issue is found, it's simple to revert to the previous AMI while determining the cause of the issue.

Using instance replacement rather than patching in-place means that updates are more deterministic and predictable as launching new capacity can be a well tested code-path with clear A and B states. Each of the before and after states can be well tested in a CI/CD system before deployment starts.

When doing in-place patching, there are a lot of intermediary states between before and after applying updates, which is harder to test for all combinations of states.

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

Using Deterministic upgrades through versioned repositories

Topics

- [Using a deterministic upgraded system](#)
- [Selective update of a deterministic upgraded system](#)
- [Using persistent override with deterministic upgrade](#)

Using a deterministic upgraded system

Note

The default behavior of the package manager has changed from AL2.

Deterministic upgrades are a powerful way to ensure all changes to production environments can be fully tested before wide deployment. Each new AL2023 AMI is locked to a particular version of AL2023. This provides deterministic behavior of what versions of OS packages are installed when launching the specific AMI. In-place updates can be to a specific release version, ensuring deterministic behavior across a fleet. As you move to new AMIs or in-place update versions, you can test each one in your CI/CD pipeline, catching any potential issues before deploying to production environments.

You can use tools such as [AWS Systems Manager Patch Manager](#) to orchestrate applying updates across a fleet. For building derived AMIs when new AL2023 AMIs are released, [EC2 Image Builder](#) can automatically build, patch, and test AMIs, or you can [Receive notifications on new updates](#) to know when new base AMIs are available, or to trigger your own AMI building pipelines.

For information on restricting updates to those from a particular advisory, see [Applying security updates in-place](#)

For patching in-place, you can use the `dnf` package manager. When you run the `dnf upgrade` command, the system checks for upgrades in the repository that the `releasever` variable specifies. A valid `releasever` is either *latest* or a date-stamped version such as *2023.9.20251117*.

You can change the value of `releasever` using one of the following methods. These methods are listed in descending system priority. This means that method 1 overrides methods 2 and 3, and method 2 overrides method 3.

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

In the following example, the version is `2023.0.20230210`:

```
$ rpm -q system-release
system-release-2023.0.20230210-0.amzn2023.noarch
```

In a newly installed system, the override variable is not present. No upgrades are available because the system is locked to the installed version of `system-release`.

```
$ cat /etc/dnf/vars/releasever
cat: /etc/dnf/vars/releasever: No such file or directory
```

```
$ sudo dnf upgrade
Last metadata expiration check: 0:00:02 ago on Wed 15 Feb 2023 06:14:12 PM UTC.
Dependencies resolved.
Nothing to do.
Complete!
```

You can get packages of a specific version by using the `releasever` flag to provide the version that you want.

```
$ rpm -q system-release
system-release-2023.0.20230222-0.amzn2023.noarch
```

```
$ sudo dnf upgrade --releasever=2023.0.20230329
Amazon Linux 2023 repository                26 MB/s | 12 MB      00:00
Dependencies resolved.
=====
Package                                Arch      Version                                Repository      Size
=====
Installing:
```

```

kernel                aarch64 6.1.21-1.45.amzn2023      amazonlinux 26 M
Upgrading:
amazon-linux-repo-s3   noarch  2023.0.20230329-0.amzn2023    amazonlinux 18 k
ca-certificates        noarch  2023.2.60-1.0.amzn2023.0.1    amazonlinux 828 k
cloud-init             noarch  22.2.2-1.amzn2023.1.7         amazonlinux 1.1 M

... [ list edited for clarity ]

system-release         noarch  2023.0.20230329-0.amzn2023    amazonlinux 29 k

... [ list edited for clarity ]

vim-data               noarch  2:9.0.1403-1.amzn2023.0.1     amazonlinux 25 k
vim-minimal            aarch64 2:9.0.1403-1.amzn2023.0.1     amazonlinux 753 k

Transaction Summary
=====
Install    1 Package
Upgrade    42 Packages

Total download size: 56 M

```

Because the `--releasever` option overrides both `system-release` and `/etc/dnf/vars/releasever`, the result of this upgrade is the following:

1. The upgrade replaces all installed packages that changed between the previous and new versions.
2. The upgrade locks the system to the repository for the new version of `system-release`.

By always specifying what `releasever` (i.e. AL2023 release) to update to, you have 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 that all updates in a new release are installed rather than selecting specific updates. Only applying part of an update to the OS should be an exception to standard practice of taking the whole update.

You might want to install selected packages from a recent release, while leaving the system locked to the original release version.

You can use `dnf check-update` to identify the packages that you want to upgrade.

```
$ sudo dnf check-update --releasever=latest --security
Amazon Linux 2023 repository          13 MB/s | 10 MB      00:00
Last metadata expiration check: 0:00:02 ago on Wed 15 Feb 2023 02:52:21 AM UTC.

bind-libs.aarch64                     32:9.16.27-1.amzn2023.0.1      amazonlinux
bind-license.noarch                   32:9.16.27-1.amzn2023.0.1      amazonlinux
bind-utils.aarch64                    32:9.16.27-1.amzn2023.0.1      amazonlinux
cryptsetup.aarch64                    2.4.3-2.amzn2023.0.1           amazonlinux
cryptsetup-libs.aarch64               2.4.3-2.amzn2023.0.1           amazonlinux
curl-minimal.aarch64                  7.85.0-1.amzn2023.0.1          amazonlinux
glibc.aarch64                         2.34-40.amzn2023.0.2           amazonlinux
glibc-all-langpacks.aarch64           2.34-40.amzn2023.0.2           amazonlinux
glibc-common.aarch64                  2.34-40.amzn2023.0.2           amazonlinux
glibc-locale-source.aarch64           2.34-40.amzn2023.0.2           amazonlinux
gmp.aarch64                           1:6.2.1-2.amzn2023.0.1         amazonlinux
gnupg2-minimal.aarch64                2.3.7-1.amzn2023.0.2           amazonlinux
gzip.aarch64                          1.10-5.amzn2023.0.1            amazonlinux
kernel.aarch64                        6.1.12-17.42.amzn2023          amazonlinux
kernel-tools.aarch64                  6.1.12-17.42.amzn2023          amazonlinux
libarchive.aarch64                    3.5.3-2.amzn2023.0.1           amazonlinux
libcurl-minimal.aarch64               7.85.0-1.amzn2023.0.1          amazonlinux
libsepol.aarch64                      3.4-3.amzn2023.0.2             amazonlinux
libsolv.aarch64                       0.7.22-1.amzn2023.0.1          amazonlinux
libxml2.aarch64                       2.9.14-1.amzn2023.0.1          amazonlinux
logrotate.aarch64                     3.20.1-2.amzn2023.0.2          amazonlinux
lua-libs.aarch64                      5.4.4-3.amzn2023.0.1           amazonlinux
lz4-libs.aarch64                      1.9.4-1.amzn2023.0.1           amazonlinux
openssl.aarch64                       1:3.0.5-1.amzn2023.0.3         amazonlinux
openssl-libs.aarch64                  1:3.0.5-1.amzn2023.0.3         amazonlinux
pcre2.aarch64                         10.40-1.amzn2023.0.1           amazonlinux
pcre2-syntax.noarch                   10.40-1.amzn2023.0.1           amazonlinux
rsync.aarch64                         3.2.6-1.amzn2023.0.2           amazonlinux
vim-common.aarch64                    2:9.0.475-1.amzn2023.0.1       amazonlinux
vim-data.noarch                       2:9.0.475-1.amzn2023.0.1       amazonlinux
vim-enhanced.aarch64                  2:9.0.475-1.amzn2023.0.1       amazonlinux
vim-filesystem.noarch                 2:9.0.475-1.amzn2023.0.1       amazonlinux
vim-minimal.aarch64                   2:9.0.475-1.amzn2023.0.1       amazonlinux
xz.aarch64                            5.2.5-9.amzn2023.0.1           amazonlinux
xz-libs.aarch64                       5.2.5-9.amzn2023.0.1           amazonlinux
```

zlib.aarch64

1.2.11-32.amzn2023.0.3

amazonlinux

Install the packages that you want to upgrade. Use `sudo dnf upgrade --releasever=latest` and the package names to ensure that the `system-release` package remains unchanged.

```
$ sudo dnf upgrade --releasever=latest openssl openssl-libs
```

```
Last metadata expiration check: 0:01:28 ago on Wed 15 Feb 2023 02:52:21 AM UTC.
```

```
Dependencies resolved.
```

```
=====
Package           Arch      Version                               Repository      Size
=====
Upgrading:
openssl           aarch64   1:3.0.5-1.amzn2023.0.3             amazonlinux     1.1 M
openssl-libs      aarch64   1:3.0.5-1.amzn2023.0.3             amazonlinux     2.1 M
```

```
Transaction Summary
```

```
=====
Upgrade 2 Packages
```

```
Total download size: 3.2 M
```

Note

Using `sudo dnf upgrade --releasever=latest` updates all packages, including `system-release`. Then, the version remains locked to the new `system-release` unless you set the persistent override.

Using persistent override with deterministic upgrade

Note

With deterministic updates, you can integrate OS changes into your CI/CD pipeline. Disabling deterministic updates removes the ability to test before deploying.

Instead of adding `--releasever=latest`, you can use persistent override to *unlock* the system by setting the variable value to *latest*. By always using `latest`, this reverts the behavior of AL2023 to the AL2 update model, where any call to the package manager will *always* look at the latest release, and is not locked to any specific version of the OS.

⚠ Warning

By unlocking the package manager by using a persistent override of deterministic updates, you take the risk discovering any possible incompatibility between your application and an OS update in production.

While incompatibilities *are* rare, with an OS update you are integrating new code changes into your environment, integration tests can prevent deploying code changes that have a negative impact on production environments.

```
$ echo latest | sudo tee /etc/dnf/vars/releasever
latest
```

```
$ sudo dnf upgrade
```

```
Last metadata expiration check: 0:03:36 ago on Wed 15 Feb 2023 02:52:21 AM UTC.
Dependencies resolved.
```

```
=====
Package                                Arch    Version                                Repository    Size
=====
```

Installing:				
kernel	aarch64	6.1.73-45.135.amzn2023	amazonlinux	24 M
Upgrading:				
acl	aarch64	2.3.1-2.amzn2023.0.1	amazonlinux	72 k
alternatives	aarch64	1.15-2.amzn2023.0.1	amazonlinux	36 k
amazon-ec2-net-utils	noarch	2.3.0-1.amzn2023.0.1	amazonlinux	16 k
at	aarch64	3.1.23-6.amzn2023.0.1	amazonlinux	60 k
attr	aarch64	2.5.1-3.amzn2023.0.1	amazonlinux	59 k
audit	aarch64	3.0.6-1.amzn2023.0.1	amazonlinux	249 k
audit-libs	aarch64	3.0.6-1.amzn2023.0.1	amazonlinux	116 k
aws-c-auth-libs	aarch64	0.6.5-6.amzn2023.0.2	amazonlinux	79 k
aws-c-cal-libs	aarch64	0.5.12-7.amzn2023.0.2	amazonlinux	34 k
aws-c-common-libs	aarch64	0.6.14-6.amzn2023.0.2	amazonlinux	119 k
aws-c-compression-libs	aarch64	0.2.14-5.amzn2023.0.2	amazonlinux	22 k
aws-c-event-stream-libs	aarch64	0.2.7-5.amzn2023.0.2	amazonlinux	47 k
aws-c-http-libs	aarch64	0.6.8-6.amzn2023.0.2	amazonlinux	147 k
aws-c-io-libs	aarch64	0.10.12-5.amzn2023.0.6	amazonlinux	109 k
aws-c-mqtt-libs	aarch64	0.7.8-7.amzn2023.0.2	amazonlinux	61 k
aws-c-s3-libs	aarch64	0.1.27-5.amzn2023.0.3	amazonlinux	54 k
aws-c-sdkutils-libs	aarch64	0.1.1-5.amzn2023.0.2	amazonlinux	26 k
aws-checksums-libs	aarch64	0.1.12-5.amzn2023.0.2	amazonlinux	50 k
awscli-2	noarch	2.7.8-1.amzn2023.0.4	amazonlinux	7.3 M

basesystem	noarch	11-11.amzn2023.0.1	amazonlinux	7.8 k
bash	aarch64	5.1.8-2.amzn2023.0.1	amazonlinux	1.6 M
bash-completion	noarch	1:2.11-2.amzn2023.0.1	amazonlinux	292 k
bc	aarch64	1.07.1-14.amzn2023.0.1	amazonlinux	120 k
bind-libs	aarch64	32:9.16.27-1.amzn2023.0.1	amazonlinux	1.2 M
bind-license	noarch	32:9.16.27-1.amzn2023.0.1	amazonlinux	14 k
bind-utils	aarch64	32:9.16.27-1.amzn2023.0.1	amazonlinux	206 k
binutils	aarch64	2.38-20.amzn2023.0.3	amazonlinux	4.6 M
boost-filesystem	aarch64	1.75.0-4.amzn2023.0.1	amazonlinux	55 k
boost-system	aarch64	1.75.0-4.amzn2023.0.1	amazonlinux	14 k
boost-thread	aarch64	1.75.0-4.amzn2023.0.1	amazonlinux	54 k
bzip2	aarch64	1.0.8-6.amzn2023.0.1	amazonlinux	53 k
bzip2-libs	aarch64	1.0.8-6.amzn2023.0.1	amazonlinux	44 k
c-ares	aarch64	1.17.2-1.amzn2023.0.1	amazonlinux	107 k
ca-certificates	noarch	2021.2.50-1.0.amzn2023.0.3	amazonlinux	343 k
checkpolicy	aarch64	3.4-3.amzn2023.0.1	amazonlinux	345 k
chkconfig	aarch64	1.15-2.amzn2023.0.1	amazonlinux	162 k
chrony	aarch64	4.2-7.amzn2023.0.4	amazonlinux	314 k
cloud-init	noarch	22.2.2-1.amzn2023.1.7	amazonlinux	1.1 M
cloud-utils-growpart	aarch64	0.31-8.amzn2023.0.2	amazonlinux	31 k
coreutils	aarch64	8.32-30.amzn2023.0.2	amazonlinux	1.1 M
coreutils-common	aarch64	8.32-30.amzn2023.0.2	amazonlinux	2.0 M
cpio	aarch64	2.13-10.amzn2023.0.1	amazonlinux	269 k
cracklib	aarch64	2.9.6-27.amzn2023.0.1	amazonlinux	83 k
cracklib-dicts	aarch64	2.9.6-27.amzn2023.0.1	amazonlinux	3.6 M
crontabs	noarch	1.11-24.20190603git.amzn2023.0.1	amazonlinux	19 k
crypto-policies	noarch	20230128-1.gitdfb10ea.amzn2023.0.1	amazonlinux	61 k
crypto-policies-scripts	noarch	20230128-1.gitdfb10ea.amzn2023.0.1	amazonlinux	81 k
...				
Installing dependencies:				
amazon-linux-repo-cdn	noarch	2023.0.20230210-0.amzn2023	amazonlinux	16 k
xxhash-libs	aarch64	0.8.0-3.amzn2023.0.1	amazonlinux	32 k
Installing weak dependencies:				
amazon-chrony-config	noarch	4.2-7.amzn2023.0.4	amazonlinux	14 k
gawk-all-langpacks	aarch64	5.1.0-3.amzn2023.0.1	amazonlinux	207 k

Transaction Summary

```
=====
Install    5 Packages
Upgrade   413 Packages
```

Total download size: 199 M

Note

If you used the override variable `/etc/dnf/vars/releasever`, use the following command to restore the default locking behavior by erasing the override value.

```
$ sudo rm /etc/dnf/vars/releasever
```

The use of a persistent override to using `latest` rather than a specific version is akin to the default behavior of AL2. There are services that build AMIs based on AL2 which disable this behavior, and lock to specific package versions like you get by default on AL2023.

Rather than disabling deterministic updates, we recommend replacing instances with ones launched from a new AMI. If instance replacement is not an option, we recommend using tools such as [AWS Systems Manager Patch Manager](#) to orchestrate applying updates across a fleet. [EC2 Image Builder](#) can also automatically build, patch, and test your own AMIs derived from AL2023 base images. You can also [Receive notifications on new updates](#) which can be used to trigger your own AMI building pipelines.

Using `latest` in a pre-production environment, and then deploying to production using `latest` does *not* provide protection from any issue between an OS update and your application. A new AL2023 release can be at any point in time, and thus all uses of `latest` in production carry risk.

Manage package and operating system updates in AL2023

Unlike previous versions of Amazon Linux, AL2023 AMIs are locked to a specific version of the Amazon Linux repository. To apply both security and bug fixes to an AL2023 instance, update the DNF configuration to the latest available release version. Alternatively, launch a newer AL2023 instance.

This section describes how to manage DNF packages and repositories on a running instance. It also describes how to configure DNF from a user data script to enable the latest available Amazon Linux repository at launch time. For more information, see [DNF Command Reference](#).

It is recommended to apply *all* updates available in a new AL2023 release. Picking just security updates, or only specific updates should be the exception rather than rule. For listing which

[Security advisories](#) are relevant to a particular instance, see [Listing applicable Advisories](#). For information on installing *only* updates relevant to 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 security updates using DNF and repository versions](#)
- [Automatic service restart after \(security\) updates](#)
- [When is a reboot required to apply security updates?](#)
- [Launching an instance with the latest repository version enabled](#)
- [Getting package support information](#)
- [Checking for newer repository versions with `dnf check-release-update`](#)
- [Adding, enabling, or disabling new repositories](#)
- [Adding repositories with cloud-init](#)

Checking for available package updates

You can use the `dnf check-update` command to check for any updates for your system. For AL2023, we recommend that you add the `--releasever=version-number` option to the command.

When you add this option, DNF also checks for updates for a later version of the repository. For example, after you run the `dnf check-update` command, use the latest returned version as the value for the *version-number*.

If the instance is updated to use the latest version of the repository, the output includes a list of all the packages to be updated.

Note

If you don't specify the release version with the optional flag to the `dnf check-update` command, only the currently configured repository version is checked. This means that packages in the later version of the repository aren't checked.

Updates in a specific version

In this example we are going to look at what updates are available in the [2023.1.20230628](#) release if we launched a container of the [2023.0.20230315](#) release.

Note

This example uses the [2023.0.20230315](#) and [2023.1.20230628](#) releases, and these *are not* the latest release of AL2023. See the [AL2023 Release Notes](#) for the latest releases, which contain the latest security updates.

In this example we will be starting with a container image for the [2023.0.20230315](#) release.

First, we fetch this container image from the container registry. The `.0` at the end indicates the version of the image for a particular release; this image version is usually zero.

```
$ docker pull public.ecr.aws/amazonlinux/amazonlinux:2023.0.20230315.0
2023.0.20230315.0: Pulling from amazonlinux/amazonlinux
b76f3b09316a: Pull complete
Digest: sha256:94e7183b0739140dbd5b639fb7600f0a2299cec5df8780c26d9cb409da5315a9
Status: Downloaded newer image for public.ecr.aws/amazonlinux/
amazonlinux:2023.0.20230315.0
public.ecr.aws/amazonlinux/amazonlinux:2023.0.20230315.0
```

We can now spawn a shell inside the container, from which we will check for updates.

```
$ docker run -it public.ecr.aws/amazonlinux/amazonlinux:2023.0.20230315.0
bash-5.2#
```

The `dnf check-update` command is now used to check updates available in the [2023.1.20230628](#) release.

Note

Applying package updates is a privileged operation. Although elevating privileges is typically not required when running in a container, if running in a non-containerized environment such as an Amazon EC2 instance, you can *check* for updates without elevating privileges.

```
$ dnf check-update --releasever=2023.1.20230628
```

```
Amazon Linux 2023 repository
```

```
60 MB/s | 15 MB 00:00
```

```
Last metadata expiration check: 0:00:02 ago on Mon Jul 22 17:25:34 2024.
```

amazon-linux-repo-cdn.noarch	2023.1.20230628-0.amzn2023	amazonlinux
ca-certificates.noarch	2023.2.60-1.0.amzn2023.0.2	amazonlinux
curl-minimal.x86_64	8.0.1-1.amzn2023	amazonlinux
glib2.x86_64	2.74.7-688.amzn2023.0.1	amazonlinux
glibc.x86_64	2.34-52.amzn2023.0.3	amazonlinux
glibc-common.x86_64	2.34-52.amzn2023.0.3	amazonlinux
glibc-minimal-langpack.x86_64	2.34-52.amzn2023.0.3	amazonlinux
gnupg2-minimal.x86_64	2.3.7-1.amzn2023.0.4	amazonlinux
keyutils-libs.x86_64	1.6.3-1.amzn2023	amazonlinux
libcap.x86_64	2.48-2.amzn2023.0.3	amazonlinux
libcurl-minimal.x86_64	8.0.1-1.amzn2023	amazonlinux
libgcc.x86_64	11.3.1-4.amzn2023.0.3	amazonlinux
libgomp.x86_64	11.3.1-4.amzn2023.0.3	amazonlinux
libstdc++.x86_64	11.3.1-4.amzn2023.0.3	amazonlinux
libxml2.x86_64	2.10.4-1.amzn2023.0.1	amazonlinux
ncurses-base.noarch	6.2-4.20200222.amzn2023.0.4	amazonlinux
ncurses-libs.x86_64	6.2-4.20200222.amzn2023.0.4	amazonlinux
openssl-libs.x86_64	1:3.0.8-1.amzn2023.0.3	amazonlinux
python3-rpm.x86_64	4.16.1.3-12.amzn2023.0.6	amazonlinux
rpm.x86_64	4.16.1.3-12.amzn2023.0.6	amazonlinux
rpm-build-libs.x86_64	4.16.1.3-12.amzn2023.0.6	amazonlinux
rpm-libs.x86_64	4.16.1.3-12.amzn2023.0.6	amazonlinux
rpm-sign-libs.x86_64	4.16.1.3-12.amzn2023.0.6	amazonlinux
system-release.noarch	2023.1.20230628-0.amzn2023	amazonlinux
tzdata.noarch	2023c-1.amzn2023.0.1	amazonlinux
bash-5.2#		

The version of the system-release package shows the release that a `dnf upgrade` command would update to, which is the [2023.1.20230628](#) release that was requested in the `dnf check-update --releasever=2023.1.20230628` command.

Updates in the latest version

In this example we are going to look at what updates are available in the latest version of AL2023 if we launched a container of the [2023.4.20240319](#) release. At the time of writing, the latest release is [2023.5.20240708](#), so the listed updates in this example will be as of that release.

Note

This example uses the [2023.4.20240319](#) and [2023.5.20240708](#) releases, the latter being the latest release *at the time of writing*. For more information on the latest releases, see the [AL2023 Release Notes](#).

In this example we will be starting with a container image for the [2023.4.20240319](#) release.

First, we fetch this container image from the container registry. The `.1` at the end indicates the version of the image for a particular release. While the image version is typically zero, this example uses a release where the image version is one.

```
$ docker pull public.ecr.aws/amazonlinux/amazonlinux:2023.4.20240319.1
2023.4.20240319.1: Pulling from amazonlinux/amazonlinux
6de065fda9a2: Pull complete
Digest: sha256:b4838c4cc9211d966b6ea158dacc9eda7433a16ba94436508c2d9f01f7658b4e
Status: Downloaded newer image for public.ecr.aws/amazonlinux/
amazonlinux:2023.4.20240319.1
public.ecr.aws/amazonlinux/amazonlinux:2023.4.20240319.1
```

We can now spawn a shell inside the container, from which we will check for updates.

```
$ docker run -it public.ecr.aws/amazonlinux/amazonlinux:2023.4.20240319.1
bash-5.2#
```

The `dnf check-update` command is now used to check updates available in the latest release, which *at the time of writing* was [2023.5.20240708](#).

Note

Applying package updates is a privileged operation. Although elevating privileges is typically not required when running in a container, if running in a non-containerized environment such as an Amazon EC2 instance, you can *check* for updates without elevating privileges.

```
$ dnf --releasever=latest check-update
```

```
Amazon Linux 2023 repository
```

```
78 MB/s | 25 MB 00:00
```

```
Last metadata expiration check: 0:00:04 ago on Mon Jul 22 17:39:13 2024.
```

amazon-linux-repo-cdn.noarch	2023.5.20240708-1.amzn2023	amazonlinux
curl-minimal.x86_64	8.5.0-1.amzn2023.0.4	amazonlinux
dnf.noarch	4.14.0-1.amzn2023.0.5	amazonlinux
dnf-data.noarch	4.14.0-1.amzn2023.0.5	amazonlinux
expat.x86_64	2.5.0-1.amzn2023.0.4	amazonlinux
glibc.x86_64	2.34-52.amzn2023.0.10	amazonlinux
glibc-common.x86_64	2.34-52.amzn2023.0.10	amazonlinux
glibc-minimal-langpack.x86_64	2.34-52.amzn2023.0.10	amazonlinux
krb5-libs.x86_64	1.21-3.amzn2023.0.4	amazonlinux
libblkid.x86_64	2.37.4-1.amzn2023.0.4	amazonlinux
libcurl-minimal.x86_64	8.5.0-1.amzn2023.0.4	amazonlinux
libmount.x86_64	2.37.4-1.amzn2023.0.4	amazonlinux
libnghttp2.x86_64	1.59.0-3.amzn2023.0.1	amazonlinux
libsmartcols.x86_64	2.37.4-1.amzn2023.0.4	amazonlinux
libuuid.x86_64	2.37.4-1.amzn2023.0.4	amazonlinux
openssl-libs.x86_64	1:3.0.8-1.amzn2023.0.12	amazonlinux
python3.x86_64	3.9.16-1.amzn2023.0.8	amazonlinux
python3-dnf.noarch	4.14.0-1.amzn2023.0.5	amazonlinux
python3-libs.x86_64	3.9.16-1.amzn2023.0.8	amazonlinux
system-release.noarch	2023.5.20240708-1.amzn2023	amazonlinux
yum.noarch	4.14.0-1.amzn2023.0.5	amazonlinux
bash-5.2#		

The version of the system-release package shows the release that a `dnf upgrade` command would update to.

For this command, if there are newer packages available, the return code is 100. If there aren't any newer packages available, the return code is 0. In addition, the output also lists all the packages to update.

Applying security updates using DNF and repository versions

New package updates and security updates are made available to new repository versions only. For instances that you launched from earlier AL2023 AMI versions, you must update the repository version before you can install security updates. The `dnf check-release-update` command includes an example update command that updates all the packages that are installed on the system to versions in a newer repository.

Note

If you don't specify the release version with the optional flag to the `dnf check-update` command, only the currently configured repository version is checked. This means that any update to installed packages present in any later version of the repository aren't applied.

This section covers the recommended upgrade path of applying all available updates rather than picking and choosing individual updates or only ones marked as security updates. By applying all updates, existing instances are moved to the same package set as launching an updated AMI. This consistency reduces the variation of package versions across a fleet. For more information on applying specific updates, see [Applying security updates in-place](#).

Applying updates in a specific version

In this example we are going to apply updates available in the [2023.1.20230628](#) release if we launched a container of the [2023.0.20230315](#) release.

Note

This example uses the [2023.0.20230315](#) and [2023.1.20230628](#) releases, and these *are not* the latest release of AL2023. See the [AL2023 Release Notes](#) for the latest releases, which contain the latest security updates.

In this example we will be starting with a container image for the [2023.0.20230315](#) release.

First, we fetch this container image from the container registry. The `.0` at the end indicates the version of the image for a particular release; this image version is usually zero.

```
$ docker pull public.ecr.aws/amazonlinux/amazonlinux:2023.0.20230315.0
2023.0.20230315.0: Pulling from amazonlinux/amazonlinux
b76f3b09316a: Pull complete
Digest: sha256:94e7183b0739140dbd5b639fb7600f0a2299cec5df8780c26d9cb409da5315a9
Status: Downloaded newer image for public.ecr.aws/amazonlinux/
amazonlinux:2023.0.20230315.0
public.ecr.aws/amazonlinux/amazonlinux:2023.0.20230315.0
```

We can now spawn a shell inside the container, from which we will apply updates.

```
$ docker run -it public.ecr.aws/amazonlinux/amazonlinux:2023.0.20230315.0
bash-5.2#
```

The `dnf upgrade` command is now used to apply all of the updates present in the [2023.1.20230628](#) release.

Note

Applying package updates is a privileged operation. Although elevating privileges is typically not required when running in a container, if running in a non-containerized environment such as an Amazon EC2 instance, you will need to run the `dnf upgrade` command as the root user. This can be done using the `sudo` or `su` commands.

```
$ dnf upgrade --releasever=2023.1.20230628
Amazon Linux 2023 repository                38 MB/s | 15 MB      00:00
Last metadata expiration check: 0:00:02 ago on Mon Jul 22 17:49:08 2024.
Dependencies resolved.
=====
Package                                Arch    Version                                Repository    Size
=====
Upgrading:
amazon-linux-repo-cdn                  noarch  2023.1.20230628-0.amzn2023            amazonlinux   18 k
ca-certificates                        noarch  2023.2.60-1.0.amzn2023.0.2            amazonlinux   829 k
curl-minimal                           x86_64  8.0.1-1.amzn2023                      amazonlinux   150 k
glib2                                   x86_64  2.74.7-688.amzn2023.0.1               amazonlinux   2.7 M
glibc                                   x86_64  2.34-52.amzn2023.0.3                  amazonlinux   1.9 M
glibc-common                           x86_64  2.34-52.amzn2023.0.3                  amazonlinux   307 k
```

glibc-minimal-langpack	x86_64	2.34-52.amzn2023.0.3	amazonlinux	35 k
gnupg2-minimal	x86_64	2.3.7-1.amzn2023.0.4	amazonlinux	421 k
keyutils-libs	x86_64	1.6.3-1.amzn2023	amazonlinux	33 k
libcap	x86_64	2.48-2.amzn2023.0.3	amazonlinux	67 k
libcurl-minimal	x86_64	8.0.1-1.amzn2023	amazonlinux	249 k
libgcc	x86_64	11.3.1-4.amzn2023.0.3	amazonlinux	105 k
libgomp	x86_64	11.3.1-4.amzn2023.0.3	amazonlinux	280 k
libstdc++	x86_64	11.3.1-4.amzn2023.0.3	amazonlinux	744 k
libxml2	x86_64	2.10.4-1.amzn2023.0.1	amazonlinux	706 k
ncurses-base	noarch	6.2-4.20200222.amzn2023.0.4	amazonlinux	60 k
ncurses-libs	x86_64	6.2-4.20200222.amzn2023.0.4	amazonlinux	328 k
openssl-libs	x86_64	1:3.0.8-1.amzn2023.0.3	amazonlinux	2.2 M
python3-rpm	x86_64	4.16.1.3-12.amzn2023.0.6	amazonlinux	88 k
rpm	x86_64	4.16.1.3-12.amzn2023.0.6	amazonlinux	486 k
rpm-build-libs	x86_64	4.16.1.3-12.amzn2023.0.6	amazonlinux	90 k
rpm-libs	x86_64	4.16.1.3-12.amzn2023.0.6	amazonlinux	309 k
rpm-sign-libs	x86_64	4.16.1.3-12.amzn2023.0.6	amazonlinux	21 k
system-release	noarch	2023.1.20230628-0.amzn2023	amazonlinux	29 k
tzdata	noarch	2023c-1.amzn2023.0.1	amazonlinux	433 k

Transaction Summary

=====

Upgrade 25 Packages

Total download size: 12 M

Is this ok [y/N]:

The version of the system-release package shows the release that a `dnf upgrade` command would update to, which is the [2023.1.20230628](#) release that was requested in the `dnf upgrade --releasever=2023.1.20230628` command.

By default, `dnf` will ask you to confirm you wish to apply the updates. You can bypass this prompt by using the `-y` flag to `dnf`. For this example, the `dnf upgrade -y --releasever=2023.1.20230628` command would not ask for confirmation before applying the updates. This is useful in scripts or other automation environments.

Once confirming you want to apply the updates, `dnf` applies them.

Is this ok [y/N]:y

Downloading Packages:

(1/25): libcap-2.48-2.amzn2023.0.3.x86_64.rpm	1.5 MB/s 67 kB	00:00
(2/25): python3-rpm-4.16.1.3-12.amzn2023.0.6.x86	2.1 MB/s 88 kB	00:00
(3/25): libcurl-minimal-8.0.1-1.amzn2023.x86_64.	2.6 MB/s 249 kB	00:00

```

(4/25): glib2-2.74.7-688.amzn2023.0.1.x86_64.rpm 26 MB/s | 2.7 MB 00:00
(5/25): glibc-minimal-langpack-2.34-52.amzn2023. 1.3 MB/s | 35 kB 00:00
(6/25): rpm-build-libs-4.16.1.3-12.amzn2023.0.6. 2.8 MB/s | 90 kB 00:00
(7/25): rpm-libs-4.16.1.3-12.amzn2023.0.6.x86_64 6.6 MB/s | 309 kB 00:00
(8/25): libgcc-11.3.1-4.amzn2023.0.3.x86_64.rpm 3.9 MB/s | 105 kB 00:00
(9/25): glibc-common-2.34-52.amzn2023.0.3.x86_64 11 MB/s | 307 kB 00:00
(10/25): glibc-2.34-52.amzn2023.0.3.x86_64.rpm 31 MB/s | 1.9 MB 00:00
(11/25): rpm-sign-libs-4.16.1.3-12.amzn2023.0.6. 877 kB/s | 21 kB 00:00
(12/25): gnupg2-minimal-2.3.7-1.amzn2023.0.4.x86 15 MB/s | 421 kB 00:00
(13/25): openssl-libs-3.0.8-1.amzn2023.0.3.x86_6 35 MB/s | 2.2 MB 00:00
(14/25): libxml2-2.10.4-1.amzn2023.0.1.x86_64.rp 14 MB/s | 706 kB 00:00
(15/25): curl-minimal-8.0.1-1.amzn2023.x86_64.rp 4.2 MB/s | 150 kB 00:00
(16/25): rpm-4.16.1.3-12.amzn2023.0.6.x86_64.rpm 11 MB/s | 486 kB 00:00
(17/25): libgomp-11.3.1-4.amzn2023.0.3.x86_64.rp 7.0 MB/s | 280 kB 00:00
(18/25): libstdc++-11.3.1-4.amzn2023.0.3.x86_64. 14 MB/s | 744 kB 00:00
(19/25): keyutils-libs-1.6.3-1.amzn2023.x86_64.r 1.6 MB/s | 33 kB 00:00
(20/25): ncurses-libs-6.2-4.20200222.amzn2023.0. 10 MB/s | 328 kB 00:00
(21/25): tzdata-2023c-1.amzn2023.0.1.noarch.rpm 11 MB/s | 433 kB 00:00
(22/25): amazon-linux-repo-cdn-2023.1.20230628-0 781 kB/s | 18 kB 00:00
(23/25): ca-certificates-2023.2.60-1.0.amzn2023. 16 MB/s | 829 kB 00:00
(24/25): system-release-2023.1.20230628-0.amzn20 1.5 MB/s | 29 kB 00:00
(25/25): ncurses-base-6.2-4.20200222.amzn2023.0. 3.1 MB/s | 60 kB 00:00

```

```
-----
Total                                28 MB/s | 12 MB 00:00
```

Running transaction check

Transaction check succeeded.

Running transaction test

Transaction test succeeded.

Running transaction

```

Preparing      : 1/1
Upgrading      : libgcc-11.3.1-4.amzn2023.0.3.x86_64 1/50
Running scriptlet: libgcc-11.3.1-4.amzn2023.0.3.x86_64 1/50
Upgrading      : system-release-2023.1.20230628-0.amzn2023.noarch 2/50
Upgrading      : amazon-linux-repo-cdn-2023.1.20230628-0.amzn2023.no 3/50
Upgrading      : ncurses-base-6.2-4.20200222.amzn2023.0.4.noarch 4/50
Upgrading      : tzdata-2023c-1.amzn2023.0.1.noarch 5/50
Upgrading      : glibc-common-2.34-52.amzn2023.0.3.x86_64 6/50
Running scriptlet: glibc-2.34-52.amzn2023.0.3.x86_64 7/50
Upgrading      : glibc-2.34-52.amzn2023.0.3.x86_64 7/50
Running scriptlet: glibc-2.34-52.amzn2023.0.3.x86_64 7/50
Upgrading      : glibc-minimal-langpack-2.34-52.amzn2023.0.3.x86_64 8/50
Upgrading      : libcap-2.48-2.amzn2023.0.3.x86_64 9/50
Upgrading      : gnupg2-minimal-2.3.7-1.amzn2023.0.4.x86_64 10/50
Upgrading      : libgomp-11.3.1-4.amzn2023.0.3.x86_64 11/50

```

```

Running scriptlet: ca-certificates-2023.2.60-1.0.amzn2023.0.2.noarch 12/50
Upgrading       : ca-certificates-2023.2.60-1.0.amzn2023.0.2.noarch 12/50
Running scriptlet: ca-certificates-2023.2.60-1.0.amzn2023.0.2.noarch 12/50
Upgrading       : openssl-libs-1:3.0.8-1.amzn2023.0.3.x86_64 13/50
Upgrading       : libcurl-minimal-8.0.1-1.amzn2023.x86_64 14/50
Upgrading       : curl-minimal-8.0.1-1.amzn2023.x86_64 15/50
Upgrading       : rpm-libs-4.16.1.3-12.amzn2023.0.6.x86_64 16/50
Upgrading       : rpm-4.16.1.3-12.amzn2023.0.6.x86_64 17/50
Upgrading       : rpm-build-libs-4.16.1.3-12.amzn2023.0.6.x86_64 18/50
Upgrading       : rpm-sign-libs-4.16.1.3-12.amzn2023.0.6.x86_64 19/50
Upgrading       : python3-rpm-4.16.1.3-12.amzn2023.0.6.x86_64 20/50
Upgrading       : glib2-2.74.7-688.amzn2023.0.1.x86_64 21/50
Upgrading       : libxml2-2.10.4-1.amzn2023.0.1.x86_64 22/50
Upgrading       : libstdc++-11.3.1-4.amzn2023.0.3.x86_64 23/50
Upgrading       : keyutils-libs-1.6.3-1.amzn2023.x86_64 24/50
Upgrading       : ncurses-libs-6.2-4.20200222.amzn2023.0.4.x86_64 25/50
Cleanup         : glib2-2.73.2-680.amzn2023.0.3.x86_64 26/50
Cleanup         : libstdc++-11.3.1-4.amzn2023.0.2.x86_64 27/50
Cleanup         : libxml2-2.10.3-2.amzn2023.0.1.x86_64 28/50
Cleanup         : python3-rpm-4.16.1.3-12.amzn2023.0.5.x86_64 29/50
Cleanup         : rpm-build-libs-4.16.1.3-12.amzn2023.0.5.x86_64 30/50
Cleanup         : rpm-sign-libs-4.16.1.3-12.amzn2023.0.5.x86_64 31/50
Cleanup         : rpm-libs-4.16.1.3-12.amzn2023.0.5.x86_64 32/50
Cleanup         : libcap-2.48-2.amzn2023.0.2.x86_64 33/50
Cleanup         : gnupg2-minimal-2.3.7-1.amzn2023.0.3.x86_64 34/50
Cleanup         : ncurses-libs-6.2-4.20200222.amzn2023.0.3.x86_64 35/50
Cleanup         : libgomp-11.3.1-4.amzn2023.0.2.x86_64 36/50
Cleanup         : rpm-4.16.1.3-12.amzn2023.0.5.x86_64 37/50
Cleanup         : curl-minimal-7.88.1-1.amzn2023.0.1.x86_64 38/50
Cleanup         : libcurl-minimal-7.88.1-1.amzn2023.0.1.x86_64 39/50
Cleanup         : openssl-libs-1:3.0.8-1.amzn2023.0.1.x86_64 40/50
Cleanup         : keyutils-libs-1.6.1-2.amzn2023.0.2.x86_64 41/50
Cleanup         : amazon-linux-repo-cdn-2023.0.20230315-1.amzn2023.no 42/50
Cleanup         : system-release-2023.0.20230315-1.amzn2023.noarch 43/50
Cleanup         : ca-certificates-2023.2.60-1.0.amzn2023.0.1.noarch 44/50
Cleanup         : ncurses-base-6.2-4.20200222.amzn2023.0.3.noarch 45/50
Cleanup         : glibc-minimal-langpack-2.34-52.amzn2023.0.2.x86_64 46/50
Cleanup         : glibc-2.34-52.amzn2023.0.2.x86_64 47/50
Cleanup         : glibc-common-2.34-52.amzn2023.0.2.x86_64 48/50
Cleanup         : tzdata-2022g-1.amzn2023.0.1.noarch 49/50
Cleanup         : libgcc-11.3.1-4.amzn2023.0.2.x86_64 50/50
Running scriptlet: libgcc-11.3.1-4.amzn2023.0.2.x86_64 50/50
Running scriptlet: ca-certificates-2023.2.60-1.0.amzn2023.0.2.noarch 50/50
Running scriptlet: rpm-4.16.1.3-12.amzn2023.0.6.x86_64 50/50

```

Running scriptlet:	libgcc-11.3.1-4.amzn2023.0.2.x86_64	50/50
Verifying	: libcurl-minimal-8.0.1-1.amzn2023.x86_64	1/50
Verifying	: libcurl-minimal-7.88.1-1.amzn2023.0.1.x86_64	2/50
Verifying	: libcap-2.48-2.amzn2023.0.3.x86_64	3/50
Verifying	: libcap-2.48-2.amzn2023.0.2.x86_64	4/50
Verifying	: glib2-2.74.7-688.amzn2023.0.1.x86_64	5/50
Verifying	: glib2-2.73.2-680.amzn2023.0.3.x86_64	6/50
Verifying	: python3-rpm-4.16.1.3-12.amzn2023.0.6.x86_64	7/50
Verifying	: python3-rpm-4.16.1.3-12.amzn2023.0.5.x86_64	8/50
Verifying	: glibc-minimal-langpack-2.34-52.amzn2023.0.3.x86_64	9/50
Verifying	: glibc-minimal-langpack-2.34-52.amzn2023.0.2.x86_64	10/50
Verifying	: rpm-libs-4.16.1.3-12.amzn2023.0.6.x86_64	11/50
Verifying	: rpm-libs-4.16.1.3-12.amzn2023.0.5.x86_64	12/50
Verifying	: rpm-build-libs-4.16.1.3-12.amzn2023.0.6.x86_64	13/50
Verifying	: rpm-build-libs-4.16.1.3-12.amzn2023.0.5.x86_64	14/50
Verifying	: glibc-2.34-52.amzn2023.0.3.x86_64	15/50
Verifying	: glibc-2.34-52.amzn2023.0.2.x86_64	16/50
Verifying	: libgcc-11.3.1-4.amzn2023.0.3.x86_64	17/50
Verifying	: libgcc-11.3.1-4.amzn2023.0.2.x86_64	18/50
Verifying	: glibc-common-2.34-52.amzn2023.0.3.x86_64	19/50
Verifying	: glibc-common-2.34-52.amzn2023.0.2.x86_64	20/50
Verifying	: rpm-sign-libs-4.16.1.3-12.amzn2023.0.6.x86_64	21/50
Verifying	: rpm-sign-libs-4.16.1.3-12.amzn2023.0.5.x86_64	22/50
Verifying	: openssl-libs-1:3.0.8-1.amzn2023.0.3.x86_64	23/50
Verifying	: openssl-libs-1:3.0.8-1.amzn2023.0.1.x86_64	24/50
Verifying	: gnupg2-minimal-2.3.7-1.amzn2023.0.4.x86_64	25/50
Verifying	: gnupg2-minimal-2.3.7-1.amzn2023.0.3.x86_64	26/50
Verifying	: libxml2-2.10.4-1.amzn2023.0.1.x86_64	27/50
Verifying	: libxml2-2.10.3-2.amzn2023.0.1.x86_64	28/50
Verifying	: curl-minimal-8.0.1-1.amzn2023.x86_64	29/50
Verifying	: curl-minimal-7.88.1-1.amzn2023.0.1.x86_64	30/50
Verifying	: rpm-4.16.1.3-12.amzn2023.0.6.x86_64	31/50
Verifying	: rpm-4.16.1.3-12.amzn2023.0.5.x86_64	32/50
Verifying	: libstdc++-11.3.1-4.amzn2023.0.3.x86_64	33/50
Verifying	: libstdc++-11.3.1-4.amzn2023.0.2.x86_64	34/50
Verifying	: libgomp-11.3.1-4.amzn2023.0.3.x86_64	35/50
Verifying	: libgomp-11.3.1-4.amzn2023.0.2.x86_64	36/50
Verifying	: keyutils-libs-1.6.3-1.amzn2023.x86_64	37/50
Verifying	: keyutils-libs-1.6.1-2.amzn2023.0.2.x86_64	38/50
Verifying	: ncurses-libs-6.2-4.20200222.amzn2023.0.4.x86_64	39/50
Verifying	: ncurses-libs-6.2-4.20200222.amzn2023.0.3.x86_64	40/50
Verifying	: ca-certificates-2023.2.60-1.0.amzn2023.0.2.noarch	41/50
Verifying	: ca-certificates-2023.2.60-1.0.amzn2023.0.1.noarch	42/50
Verifying	: tzdata-2023c-1.amzn2023.0.1.noarch	43/50

```

Verifying      : tzdata-2022g-1.amzn2023.0.1.noarch      44/50
Verifying      : amazon-linux-repo-cdn-2023.1.20230628-0.amzn2023.no 45/50
Verifying      : amazon-linux-repo-cdn-2023.0.20230315-1.amzn2023.no 46/50
Verifying      : system-release-2023.1.20230628-0.amzn2023.noarch 47/50
Verifying      : system-release-2023.0.20230315-1.amzn2023.noarch 48/50
Verifying      : ncurses-base-6.2-4.20200222.amzn2023.0.4.noarch 49/50
Verifying      : ncurses-base-6.2-4.20200222.amzn2023.0.3.noarch 50/50

```

Upgraded:

```

amazon-linux-repo-cdn-2023.1.20230628-0.amzn2023.noarch
ca-certificates-2023.2.60-1.0.amzn2023.0.2.noarch
curl-minimal-8.0.1-1.amzn2023.x86_64
glib2-2.74.7-688.amzn2023.0.1.x86_64
glibc-2.34-52.amzn2023.0.3.x86_64
glibc-common-2.34-52.amzn2023.0.3.x86_64
glibc-minimal-langpack-2.34-52.amzn2023.0.3.x86_64
gnupg2-minimal-2.3.7-1.amzn2023.0.4.x86_64
keyutils-libs-1.6.3-1.amzn2023.x86_64
libcap-2.48-2.amzn2023.0.3.x86_64
libcurl-minimal-8.0.1-1.amzn2023.x86_64
libgcc-11.3.1-4.amzn2023.0.3.x86_64
libgomp-11.3.1-4.amzn2023.0.3.x86_64
libstdc++-11.3.1-4.amzn2023.0.3.x86_64
libxml2-2.10.4-1.amzn2023.0.1.x86_64
ncurses-base-6.2-4.20200222.amzn2023.0.4.noarch
ncurses-libs-6.2-4.20200222.amzn2023.0.4.x86_64
openssl-libs-1:3.0.8-1.amzn2023.0.3.x86_64
python3-rpm-4.16.1.3-12.amzn2023.0.6.x86_64
rpm-4.16.1.3-12.amzn2023.0.6.x86_64
rpm-build-libs-4.16.1.3-12.amzn2023.0.6.x86_64
rpm-libs-4.16.1.3-12.amzn2023.0.6.x86_64
rpm-sign-libs-4.16.1.3-12.amzn2023.0.6.x86_64
system-release-2023.1.20230628-0.amzn2023.noarch
tzdata-2023c-1.amzn2023.0.1.noarch

```

Complete!

```
bash-5.2#
```

Updates in the latest version

In this example we are going to apply updates available in the latest version of AL2023 if we launched a container of the [2023.4.20240319](#) release. At the time of writing, the latest release is [2023.5.20240708](#), so the listed updates in this example will be as of that release.

Note

This example uses the [2023.4.20240319](#) and [2023.5.20240708](#) releases, the latter being the latest release *at the time of writing*. For more information on the latest releases, see the [AL2023 Release Notes](#).

In this example we will be starting with a container image for the [2023.4.20240319](#) release.

First, we fetch this container image from the container registry. The .1 at the end indicates the version of the image for a particular release. While the image version is typically zero, this example uses a release where the image version is one.

```
$ docker pull public.ecr.aws/amazonlinux/amazonlinux:2023.4.20240319.1
2023.4.20240319.1: Pulling from amazonlinux/amazonlinux
6de065fda9a2: Pull complete
Digest: sha256:b4838c4cc9211d966b6ea158dacc9eda7433a16ba94436508c2d9f01f7658b4e
Status: Downloaded newer image for public.ecr.aws/amazonlinux/
amazonlinux:2023.4.20240319.1
public.ecr.aws/amazonlinux/amazonlinux:2023.4.20240319.1
```

We can now spawn a shell inside the container, from which we will apply updates.

```
$ docker run -it public.ecr.aws/amazonlinux/amazonlinux:2023.4.20240319.1
bash-5.2#
```

The `dnf upgrade` command is now used to apply updates available in the latest release, which *at the time of writing* was [2023.5.20240708](#).

Note

Applying package updates is a privileged operation. Although elevating privileges is typically not required when running in a container, if running in a non-containerized environment such as an Amazon EC2 instance, you will need to run the `dnf upgrade` command as the root user. This can be done using the `sudo` or `su` commands.

By default, `dnf` will ask you to confirm you wish to apply the updates. In this example, we are bypassing this prompt by using the `-y` flag to `dnf`.

```
$ dnf -y --releasever=latest update
```

```
Amazon Linux 2023 repository 75 MB/s | 25 MB 00:00
```

```
Last metadata expiration check: 0:00:04 ago on Mon Jul 22 18:00:10 2024.
```

```
Dependencies resolved.
```

```
=====
```

Package	Arch	Version	Repository	Size
=====				
Upgrading:				
amazon-linux-repo-cdn	noarch	2023.5.20240708-1.amzn2023	amazonlinux	17 k
curl-minimal	x86_64	8.5.0-1.amzn2023.0.4	amazonlinux	160 k
dnf	noarch	4.14.0-1.amzn2023.0.5	amazonlinux	460 k
dnf-data	noarch	4.14.0-1.amzn2023.0.5	amazonlinux	34 k
expat	x86_64	2.5.0-1.amzn2023.0.4	amazonlinux	117 k
glibc	x86_64	2.34-52.amzn2023.0.10	amazonlinux	1.9 M
glibc-common	x86_64	2.34-52.amzn2023.0.10	amazonlinux	295 k
glibc-minimal-langpack	x86_64	2.34-52.amzn2023.0.10	amazonlinux	23 k
krb5-libs	x86_64	1.21-3.amzn2023.0.4	amazonlinux	758 k
libblkid	x86_64	2.37.4-1.amzn2023.0.4	amazonlinux	105 k
libcurl-minimal	x86_64	8.5.0-1.amzn2023.0.4	amazonlinux	275 k
libmount	x86_64	2.37.4-1.amzn2023.0.4	amazonlinux	132 k
libnghttp2	x86_64	1.59.0-3.amzn2023.0.1	amazonlinux	79 k
libsmartcols	x86_64	2.37.4-1.amzn2023.0.4	amazonlinux	62 k
libuuid	x86_64	2.37.4-1.amzn2023.0.4	amazonlinux	26 k
openssl-libs	x86_64	1:3.0.8-1.amzn2023.0.12	amazonlinux	2.2 M
python3	x86_64	3.9.16-1.amzn2023.0.8	amazonlinux	27 k
python3-dnf	noarch	4.14.0-1.amzn2023.0.5	amazonlinux	409 k
python3-libs	x86_64	3.9.16-1.amzn2023.0.8	amazonlinux	7.3 M
system-release	noarch	2023.5.20240708-1.amzn2023	amazonlinux	28 k
yum	noarch	4.14.0-1.amzn2023.0.5	amazonlinux	32 k

Transaction Summary

```
=====
```

```
Upgrade 21 Packages
```

```
Total download size: 14 M
```

```
Downloading Packages:
```

```
(1/21): amazon-linux-repo-cdn-2023.5.20240708-1. 345 kB/s | 17 kB 00:00
(2/21): dnf-4.14.0-1.amzn2023.0.5.noarch.rpm 6.8 MB/s | 460 kB 00:00
(3/21): dnf-data-4.14.0-1.amzn2023.0.5.noarch.rp 1.6 MB/s | 34 kB 00:00
(4/21): expat-2.5.0-1.amzn2023.0.4.x86_64.rpm 4.6 MB/s | 117 kB 00:00
(5/21): glibc-2.34-52.amzn2023.0.10.x86_64.rpm 38 MB/s | 1.9 MB 00:00
(6/21): glibc-common-2.34-52.amzn2023.0.10.x86_6 8.8 MB/s | 295 kB 00:00
(7/21): glibc-minimal-langpack-2.34-52.amzn2023. 1.7 MB/s | 23 kB 00:00
```

```

(8/21): curl-minimal-8.5.0-1.amzn2023.0.4.x86_64 998 kB/s | 160 kB      00:00
(9/21): libblkid-2.37.4-1.amzn2023.0.4.x86_64.rpm 4.1 MB/s | 105 kB      00:00
(10/21): krb5-libs-1.21-3.amzn2023.0.4.x86_64.rpm 16 MB/s | 758 kB      00:00
(11/21): libmount-2.37.4-1.amzn2023.0.4.x86_64.r 7.9 MB/s | 132 kB      00:00
(12/21): libnghttp2-1.59.0-3.amzn2023.0.1.x86_64 5.6 MB/s | 79 kB       00:00
(13/21): libsmartcols-2.37.4-1.amzn2023.0.4.x86_ 4.4 MB/s | 62 kB       00:00
(14/21): libcurl-minimal-8.5.0-1.amzn2023.0.4.x8 7.1 MB/s | 275 kB      00:00
(15/21): libuuid-2.37.4-1.amzn2023.0.4.x86_64.rpm 1.1 MB/s | 26 kB       00:00
(16/21): python3-3.9.16-1.amzn2023.0.8.x86_64.rpm 1.5 MB/s | 27 kB       00:00
(17/21): python3-dnf-4.14.0-1.amzn2023.0.5.noarc 19 MB/s | 409 kB      00:00
(18/21): system-release-2023.5.20240708-1.amzn20 1.9 MB/s | 28 kB       00:00
(19/21): yum-4.14.0-1.amzn2023.0.5.noarch.rpm    1.6 MB/s | 32 kB       00:00
(20/21): openssl-libs-3.0.8-1.amzn2023.0.12.x86_ 26 MB/s | 2.2 MB      00:00
(21/21): python3-libs-3.9.16-1.amzn2023.0.8.x86_ 59 MB/s | 7.3 MB      00:00

```

```

-----
Total                                     34 MB/s | 14 MB      00:00

```

Running transaction check

Transaction check succeeded.

Running transaction test

Transaction test succeeded.

Running transaction

```

Preparing           :                               1/1
Upgrading           : glibc-common-2.34-52.amzn2023.0.10.x86_64      1/42
Upgrading           : glibc-minimal-langpack-2.34-52.amzn2023.0.10.x86_64 2/42
Running scriptlet: glibc-2.34-52.amzn2023.0.10.x86_64      3/42
Upgrading           : glibc-2.34-52.amzn2023.0.10.x86_64      3/42
Running scriptlet: glibc-2.34-52.amzn2023.0.10.x86_64      3/42
Upgrading           : libuuid-2.37.4-1.amzn2023.0.4.x86_64        4/42
Upgrading           : openssl-libs-1:3.0.8-1.amzn2023.0.12.x86_64    5/42
Upgrading           : krb5-libs-1.21-3.amzn2023.0.4.x86_64        6/42
Upgrading           : libblkid-2.37.4-1.amzn2023.0.4.x86_64        7/42
Running scriptlet: libblkid-2.37.4-1.amzn2023.0.4.x86_64        7/42
Upgrading           : expat-2.5.0-1.amzn2023.0.4.x86_64           8/42
Upgrading           : python3-3.9.16-1.amzn2023.0.8.x86_64        9/42
Upgrading           : python3-libs-3.9.16-1.amzn2023.0.8.x86_64    10/42
Upgrading           : libnghttp2-1.59.0-3.amzn2023.0.1.x86_64     11/42
Upgrading           : libcurl-minimal-8.5.0-1.amzn2023.0.4.x86_64  12/42
Upgrading           : system-release-2023.5.20240708-1.amzn2023.noarch 13/42
Upgrading           : amazon-linux-repo-cdn-2023.5.20240708-1.amzn2023.no 14/42
Upgrading           : dnf-data-4.14.0-1.amzn2023.0.5.noarch       15/42
Upgrading           : python3-dnf-4.14.0-1.amzn2023.0.5.noarch     16/42
Upgrading           : dnf-4.14.0-1.amzn2023.0.5.noarch           17/42
Running scriptlet: dnf-4.14.0-1.amzn2023.0.5.noarch           17/42
Upgrading           : yum-4.14.0-1.amzn2023.0.5.noarch           18/42

```

Upgrading	: curl-minimal-8.5.0-1.amzn2023.0.4.x86_64	19/42
Upgrading	: libmount-2.37.4-1.amzn2023.0.4.x86_64	20/42
Upgrading	: libsmartcols-2.37.4-1.amzn2023.0.4.x86_64	21/42
Cleanup	: yum-4.14.0-1.amzn2023.0.4.noarch	22/42
Running scriptlet:	dnf-4.14.0-1.amzn2023.0.4.noarch	23/42
Cleanup	: dnf-4.14.0-1.amzn2023.0.4.noarch	23/42
Running scriptlet:	dnf-4.14.0-1.amzn2023.0.4.noarch	23/42
Cleanup	: python3-dnf-4.14.0-1.amzn2023.0.4.noarch	24/42
Cleanup	: amazon-linux-repo-cdn-2023.4.20240319-1.amzn2023.no	25/42
Cleanup	: libmount-2.37.4-1.amzn2023.0.3.x86_64	26/42
Cleanup	: curl-minimal-8.5.0-1.amzn2023.0.2.x86_64	27/42
Cleanup	: libcurl-minimal-8.5.0-1.amzn2023.0.2.x86_64	28/42
Cleanup	: krb5-libs-1.21-3.amzn2023.0.3.x86_64	29/42
Cleanup	: libblkid-2.37.4-1.amzn2023.0.3.x86_64	30/42
Cleanup	: libnghttp2-1.57.0-1.amzn2023.0.1.x86_64	31/42
Cleanup	: libsmartcols-2.37.4-1.amzn2023.0.3.x86_64	32/42
Cleanup	: system-release-2023.4.20240319-1.amzn2023.noarch	33/42
Cleanup	: dnf-data-4.14.0-1.amzn2023.0.4.noarch	34/42
Cleanup	: python3-3.9.16-1.amzn2023.0.6.x86_64	35/42
Cleanup	: python3-libs-3.9.16-1.amzn2023.0.6.x86_64	36/42
Cleanup	: openssl-libs-1:3.0.8-1.amzn2023.0.11.x86_64	37/42
Cleanup	: libuuid-2.37.4-1.amzn2023.0.3.x86_64	38/42
Cleanup	: expat-2.5.0-1.amzn2023.0.3.x86_64	39/42
Cleanup	: glibc-2.34-52.amzn2023.0.8.x86_64	40/42
Cleanup	: glibc-minimal-langpack-2.34-52.amzn2023.0.8.x86_64	41/42
Cleanup	: glibc-common-2.34-52.amzn2023.0.8.x86_64	42/42
Running scriptlet:	glibc-common-2.34-52.amzn2023.0.8.x86_64	42/42
Verifying	: amazon-linux-repo-cdn-2023.5.20240708-1.amzn2023.no	1/42
Verifying	: amazon-linux-repo-cdn-2023.4.20240319-1.amzn2023.no	2/42
Verifying	: curl-minimal-8.5.0-1.amzn2023.0.4.x86_64	3/42
Verifying	: curl-minimal-8.5.0-1.amzn2023.0.2.x86_64	4/42
Verifying	: dnf-4.14.0-1.amzn2023.0.5.noarch	5/42
Verifying	: dnf-4.14.0-1.amzn2023.0.4.noarch	6/42
Verifying	: dnf-data-4.14.0-1.amzn2023.0.5.noarch	7/42
Verifying	: dnf-data-4.14.0-1.amzn2023.0.4.noarch	8/42
Verifying	: expat-2.5.0-1.amzn2023.0.4.x86_64	9/42
Verifying	: expat-2.5.0-1.amzn2023.0.3.x86_64	10/42
Verifying	: glibc-2.34-52.amzn2023.0.10.x86_64	11/42
Verifying	: glibc-2.34-52.amzn2023.0.8.x86_64	12/42
Verifying	: glibc-common-2.34-52.amzn2023.0.10.x86_64	13/42
Verifying	: glibc-common-2.34-52.amzn2023.0.8.x86_64	14/42
Verifying	: glibc-minimal-langpack-2.34-52.amzn2023.0.10.x86_64	15/42
Verifying	: glibc-minimal-langpack-2.34-52.amzn2023.0.8.x86_64	16/42
Verifying	: krb5-libs-1.21-3.amzn2023.0.4.x86_64	17/42

Verifying	: krb5-libs-1.21-3.amzn2023.0.3.x86_64	18/42
Verifying	: libblkid-2.37.4-1.amzn2023.0.4.x86_64	19/42
Verifying	: libblkid-2.37.4-1.amzn2023.0.3.x86_64	20/42
Verifying	: libcurl-minimal-8.5.0-1.amzn2023.0.4.x86_64	21/42
Verifying	: libcurl-minimal-8.5.0-1.amzn2023.0.2.x86_64	22/42
Verifying	: libmount-2.37.4-1.amzn2023.0.4.x86_64	23/42
Verifying	: libmount-2.37.4-1.amzn2023.0.3.x86_64	24/42
Verifying	: libnghttp2-1.59.0-3.amzn2023.0.1.x86_64	25/42
Verifying	: libnghttp2-1.57.0-1.amzn2023.0.1.x86_64	26/42
Verifying	: libsmartcols-2.37.4-1.amzn2023.0.4.x86_64	27/42
Verifying	: libsmartcols-2.37.4-1.amzn2023.0.3.x86_64	28/42
Verifying	: libuuid-2.37.4-1.amzn2023.0.4.x86_64	29/42
Verifying	: libuuid-2.37.4-1.amzn2023.0.3.x86_64	30/42
Verifying	: openssl-libs-1:3.0.8-1.amzn2023.0.12.x86_64	31/42
Verifying	: openssl-libs-1:3.0.8-1.amzn2023.0.11.x86_64	32/42
Verifying	: python3-3.9.16-1.amzn2023.0.8.x86_64	33/42
Verifying	: python3-3.9.16-1.amzn2023.0.6.x86_64	34/42
Verifying	: python3-dnf-4.14.0-1.amzn2023.0.5.noarch	35/42
Verifying	: python3-dnf-4.14.0-1.amzn2023.0.4.noarch	36/42
Verifying	: python3-libs-3.9.16-1.amzn2023.0.8.x86_64	37/42
Verifying	: python3-libs-3.9.16-1.amzn2023.0.6.x86_64	38/42
Verifying	: system-release-2023.5.20240708-1.amzn2023.noarch	39/42
Verifying	: system-release-2023.4.20240319-1.amzn2023.noarch	40/42
Verifying	: yum-4.14.0-1.amzn2023.0.5.noarch	41/42
Verifying	: yum-4.14.0-1.amzn2023.0.4.noarch	42/42

Upgraded:

```
amazon-linux-repo-cdn-2023.5.20240708-1.amzn2023.noarch
curl-minimal-8.5.0-1.amzn2023.0.4.x86_64
dnf-4.14.0-1.amzn2023.0.5.noarch
dnf-data-4.14.0-1.amzn2023.0.5.noarch
expat-2.5.0-1.amzn2023.0.4.x86_64
glibc-2.34-52.amzn2023.0.10.x86_64
glibc-common-2.34-52.amzn2023.0.10.x86_64
glibc-minimal-langpack-2.34-52.amzn2023.0.10.x86_64
krb5-libs-1.21-3.amzn2023.0.4.x86_64
libblkid-2.37.4-1.amzn2023.0.4.x86_64
libcurl-minimal-8.5.0-1.amzn2023.0.4.x86_64
libmount-2.37.4-1.amzn2023.0.4.x86_64
libnghttp2-1.59.0-3.amzn2023.0.1.x86_64
libsmartcols-2.37.4-1.amzn2023.0.4.x86_64
libuuid-2.37.4-1.amzn2023.0.4.x86_64
openssl-libs-1:3.0.8-1.amzn2023.0.12.x86_64
python3-3.9.16-1.amzn2023.0.8.x86_64
```

```
python3-dnf-4.14.0-1.amzn2023.0.5.noarch
python3-libs-3.9.16-1.amzn2023.0.8.x86_64
system-release-2023.5.20240708-1.amzn2023.noarch
yum-4.14.0-1.amzn2023.0.5.noarch
```

```
Complete!
bash-5.2#
```

To discover AL2023 updates, do one or more of the following:

- Run the `dnf check-update` command. This checks for any unapplied updates in the version of Amazon Linux which you are locked to. This may show updates if you updated only the `system-release` package, moving what version of the repositories the instance is locked to but not applying any of the updates available in it.
- Subscribe to the Amazon Linux repository update SNS topic (`arn:aws:sns:us-east-1:137112412989:amazon-linux-2023-ami-updates`). For more information, see [Subscribing to an Amazon SNS topic](#) in the *Amazon Simple Notification Service Developer Guide*.
- Regularly refer to the [AL2023 release notes](#).
- Discover new versions by [Checking for newer repository versions with `dnf check-release-update`](#).

Important

New versions of AL2023 containing security updates are released frequently. Be sure to keep up to date with relevant security patches.

Automatic service restart after (security) updates

Amazon Linux now ships with the [smart-restart](#) package. `Smart-restart` restarts systemd services on system updates whenever a package is installed or deleted using the systems package manager. This occurs whenever `dnf (update|upgrade|downgrade)` is executed.

`Smart-restart` uses the `needs-restarting` package from `dnf-utils` and a custom denylisting mechanism to determine which services need to be restarted and whether a system reboot is advised. If a system reboot is advised, a reboot hint marker file is generated (`/run/smart-restart/reboot-hint-marker`).

To install `smart-restart`

Run the following DNF command (as you would with any other package).

```
$ sudo dnf install smart-restart
```

After the installation, the subsequent transactions will trigger the `smart-restart` logic.

Denylist

`Smart-restart` can be instructed to block certain services from being restarted. The blocked services won't contribute to the decision of whether a reboot is required. To block additional services, add a file with the suffix `-denylist` in `/etc/smart-restart-conf.d/` as shown in the following example.

```
$ cat /etc/smart-restart-conf.d/custom-denylist
# Some comments
myservice.service
```

Note

All `*-denylist` files are read and evaluated when making the decision of whether a reboot is required.

Custom hooks

In addition to denylisting, `smart-restart` provides a mechanism to run custom scripts before and after the attempts to restart the service. The custom scripts can be used to manually perform preparation steps or to inform other components of a remaining or completed restart.

All scripts in `/etc/smart-restart-conf.d/` with the suffix `-pre-restart` or `-post-restart` are executed. If the order is important, prefix all of the scripts with a number to ensure the execution order as shown in the following example.

```
$ ls /etc/smart-restart-conf.d/*-pre-restart
001-my-script-pre-restart
002-some-other-script-pre-restart
```

When is a reboot required to apply security updates?

In some situations, Amazon Linux requires a reboot to apply updates:

- Updates to the Linux kernel package require a reboot to activate the new kernel with latest security updates. Kernel livepatching may allow you to postpone security updates for a limited period of time. For details, consult [Kernel Live Patching on AL2023](#).
- On EC2 Metal instances, Amazon Linux provides microcode updates (through the `microcode_ctl` package for Intel CPUs and the `amd-ucode-firmware` package for AMD CPUs.) These microcode updates will only be activated on subsequent instance reboots. For virtualized EC2 instances, the underlying [AWS Nitro system](#) handles microcode updates for you.
- Some running systemd services will only function correctly after a full system restart. The `smart-restart` mechanism will inform you about such situations by leaving reboot hints. See [Automatic service restart after \(security\) updates](#).

Launching an instance with the latest repository version enabled

You can add DNF commands to a user-data script to control what RPM packages are installed on an Amazon Linux AMI when it's launched. In the following example, a user-data script is used to make sure that any instance launched with the user-data script has the same package updates installed.

```
#!/bin/bash
dnf upgrade --releasever=2023.0.20230210
# Additional setup and install commands below
dnf install httpd php7.4 mysql80
```

You must run this script as superuser (root). To do this, run the following command.

```
$ sudo sh -c "bash nameofscript.sh"
```

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 Amazon Linux AMI or a custom AMI that's based on the Amazon Linux AMI. The latest Amazon Linux AMI has all the necessary updates installed and is configured to point at a particular repository version.

Getting package support information

AL2023 incorporates many different open-source software projects. Each of these projects is managed independently from Amazon Linux and have different release and end-of-support schedules. To provide you with Amazon Linux specific information about these different packages, the DNF `supportinfo` plugin provides metadata about a package. In the following example, the `dnf supportinfo` command returns metadata for the `glibc` package.

```
$ sudo dnf supportinfo --pkg glibc
Last metadata expiration check: 0:07:56 ago on Wed Mar 1 23:21:49 2023.
Name           : glibc
Version        : 2.34-52.amzn2023.0.2
State          : installed
Support Status : supported
Support Periods: from 2023-03-15      : supported
                : from 2028-03-15      : unsupported
Support Statement: Amazon Linux 2023 End Of Life
Link           : https://aws.amazon.com/amazon-linux-ami/faqs/
Other Info      : This is the support statement for AL2023. The
                  ...: end of life of Amazon Linux 2023 would be March 2028.
                  ...: From this point, the Amazon Linux 2023 packages (listed
                  ...: below) will no longer, receive any updates from AWS.
```

Package support information is also available in the [support statements](#) section of the [AL2023 Release Notes](#).

Checking for newer repository versions with `dnf check-release-update`

In an AL2023 instance, you can use the DNF utility to manage repositories and apply updated RPM packages. These packages are available in the Amazon Linux repositories. You can use the DNF command `dnf check-release-update` to check for new versions of the DNF repository.

Note

AL2023 container images do not include the `dnf check-release-update` command by default.

```
$ dnf check-release-update
No such command: check-release-update. Please use /usr/bin/dnf --help
```

```
It could be a DNF plugin command, try: "dnf install 'dnf-command(check-release-update)'"
```

When `dnf install 'dnf-command(check-release-update)'` is run, dnf will install the package which provides the `check-release-update` command, which is the `dnf-plugin-release-notification` package. In the below example, the `-q` argument is given to dnf for it to have quiet output.

```
$ dnf -y -q install 'dnf-command(check-release-update)'
Installed:
  dnf-plugin-release-notification-1.2-1.amzn2023.0.2.noarch
```

In non-containerized environments such as an Amazon EC2 instance, the `check-release-update` command is included by default.

```
$ sudo dnf check-release-update
WARNING:
  A newer release of "Amazon Linux" is available.

Available Versions:

Version 2023.0.20230210:
  Run the following command to update to 2023.0.20230210:

    dnf upgrade --releasever=2023.0.20230210

Release notes:
  https://docs.aws.amazon.com/linux/al2023/release-notes/relnotes.html
```

This returns a full list of all the newer versions of the DNF repositories that are available. If nothing's returned, this means that DNF is currently configured to use the latest available version. The version of the currently installed `system-release` package sets the `releasever` DNF variable. To check the current repository version, run the following command.

```
$ rpm -q system-release --qf "%{VERSION}\n"
```

When you run DNF package transactions (such as `install`, `update`, or `remove` commands), a warning message notifies you of any new repository versions. For example, if you install the `httpd` package

on an instance that was launched from an older version of AL2023, the following output is returned.

```
$ sudo dnf install httpd -y
Last metadata expiration check: 0:16:52 ago on Wed Mar 1 23:21:49 2023.
Dependencies resolved.
=====
Package                Arch    Version                               Repository    Size
=====
Installing:
httpd                   x86_64  2.4.54-3.amzn2023.0.4               amazonlinux   46 k
Installing dependencies:
apr                     x86_64  1.7.2-2.amzn2023.0.2               amazonlinux   129 k
apr-util                x86_64  1.6.3-1.amzn2023.0.1               amazonlinux   98 k
generic-logos-httpd
noarch                 18.0.0-12.amzn2023.0.3             amazonlinux   19 k
httpd-core              x86_64  2.4.54-3.amzn2023.0.4               amazonlinux   1.3 M
httpd-filesystem        noarch  2.4.54-3.amzn2023.0.4               amazonlinux   13 k
httpd-tools             x86_64  2.4.54-3.amzn2023.0.4               amazonlinux   80 k
libbrotli               x86_64  1.0.9-4.amzn2023.0.2               amazonlinux   315 k
mailcap                 noarch  2.1.49-3.amzn2023.0.3               amazonlinux   33 k
Installing weak dependencies:
apr-util-openssl        x86_64  1.6.3-1.amzn2023.0.1               amazonlinux   17 k
mod_http2               x86_64  1.15.24-1.amzn2023.0.3             amazonlinux   152 k
mod_lua                 x86_64  2.4.54-3.amzn2023.0.4               amazonlinux   60 k

Transaction Summary
=====
Install 12 Packages

Total download size: 2.3 M
Installed size: 6.8 M
Downloading Packages:
(1/12): apr-util-openssl-1.6.3-1.am 212 kB/s | 17 kB      00:00
(2/12): apr-1.7.2-2.amzn2023.0.2.x8 1.1 MB/s | 129 kB     00:00
(3/12): httpd-core-2.4.54-3.amzn202 8.9 MB/s | 1.3 MB     00:00
(4/12): mod_http2-1.15.24-1.amzn202 1.9 MB/s | 152 kB     00:00
(5/12): apr-util-1.6.3-1.amzn2023.0 1.7 MB/s | 98 kB      00:00
(6/12): mod_lua-2.4.54-3.amzn2023.0 1.4 MB/s | 60 kB      00:00
(7/12): httpd-2.4.54-3.amzn2023.0.4 1.5 MB/s | 46 kB      00:00
(8/12): libbrotli-1.0.9-4.amzn2023. 4.4 MB/s | 315 kB     00:00
(9/12): mailcap-2.1.49-3.amzn2023.0 753 kB/s | 33 kB      00:00
(10/12): httpd-tools-2.4.54-3.amzn2 978 kB/s | 80 kB      00:00
```

```

(11/12): httpd-filesystem-2.4.54-3. 210 kB/s | 13 kB      00:00
(12/12): generic-logos-httpd-18.0.0 439 kB/s | 19 kB      00:00
-----
Total                                6.6 MB/s | 2.3 MB      00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing                :                               1/1
  Installing               : apr-1.7.2-2.amzn2023.0.2.x86_64 1/12
  Installing               : apr-util-openssl-1.6.3-1.amzn2023.0.1. 2/12
  Installing               : apr-util-1.6.3-1.amzn2023.0.1.x86_64 3/12
  Installing               : mailcap-2.1.49-3.amzn2023.0.3.noarch 4/12
  Installing               : httpd-tools-2.4.54-3.amzn2023.0.4.x86_ 5/12
  Installing               : generic-logos-httpd-18.0.0-12.amzn2023 6/12
Running scriptlet: httpd-filesystem-2.4.54-3.amzn2023.0.4 7/12
  Installing               : httpd-filesystem-2.4.54-3.amzn2023.0.4 7/12
  Installing               : httpd-core-2.4.54-3.amzn2023.0.4.x86_6 8/12
  Installing               : mod_http2-1.15.24-1.amzn2023.0.3.x86_6 9/12
  Installing               : libbrotli-1.0.9-4.amzn2023.0.2.x86_64 10/12
  Installing               : mod_lua-2.4.54-3.amzn2023.0.4.x86_64 11/12
  Installing               : httpd-2.4.54-3.amzn2023.0.4.x86_64 12/12
Running scriptlet: httpd-2.4.54-3.amzn2023.0.4.x86_64 12/12
  Verifying                : apr-1.7.2-2.amzn2023.0.2.x86_64 1/12
  Verifying                : apr-util-openssl-1.6.3-1.amzn2023.0.1. 2/12
  Verifying                : httpd-core-2.4.54-3.amzn2023.0.4.x86_6 3/12
  Verifying                : mod_http2-1.15.24-1.amzn2023.0.3.x86_6 4/12
  Verifying                : apr-util-1.6.3-1.amzn2023.0.1.x86_64 5/12
  Verifying                : mod_lua-2.4.54-3.amzn2023.0.4.x86_64 6/12
  Verifying                : libbrotli-1.0.9-4.amzn2023.0.2.x86_64 7/12
  Verifying                : httpd-2.4.54-3.amzn2023.0.4.x86_64 8/12
  Verifying                : httpd-tools-2.4.54-3.amzn2023.0.4.x86_ 9/12
  Verifying                : mailcap-2.1.49-3.amzn2023.0.3.noarch 10/12
  Verifying                : httpd-filesystem-2.4.54-3.amzn2023.0.4 11/12
  Verifying                : generic-logos-httpd-18.0.0-12.amzn2023 12/12

Installed:
  apr-1.7.2-2.amzn2023.0.2.x86_64
  apr-util-1.6.3-1.amzn2023.0.1.x86_64
  apr-util-openssl-1.6.3-1.amzn2023.0.1.x86_64
  generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch
  httpd-2.4.54-3.amzn2023.0.4.x86_64
  httpd-core-2.4.54-3.amzn2023.0.4.x86_64

```

```
httpd-filesystem-2.4.54-3.amzn2023.0.4.noarch
httpd-tools-2.4.54-3.amzn2023.0.4.x86_64
libbrotli-1.0.9-4.amzn2023.0.2.x86_64
mailcap-2.1.49-3.amzn2023.0.3.noarch
mod_http2-1.15.24-1.amzn2023.0.3.x86_64
mod_lua-2.4.54-3.amzn2023.0.4.x86_64
```

Complete!

Adding, enabling, or disabling new repositories

Warning

Only add repositories designed to be used with AL2023.

While repositories designed for other distributions may work today, there is no guarantee they will continue to do so with any package update in AL2023 or the repository not designed for use with AL2023.

To install a package from a different repository than the default Amazon Linux repositories, you will need to configure the DNF package management system to know where the repository is

To tell dnf about a package repository, add the repository information to a configuration file for that repository in the `/etc/yum.repos.d/` directory. Many third-party repositories provide either the configuration file content or an installable package which includes the configuration file.

Note

While repositories can be configured directly in the `/etc/dnf/dnf.conf` file, this is not recommended. It is recommended that each repository be configured in its own file in `/etc/yum.repos.d/`.

To find out what repositories are currently enabled, you can run the following command:

```
$ dnf repolist all --verbose
```

```
Loaded plugins: builddep, changelog, config-manager, copr, debug, debuginfo-install,
download, generate_completion_cache, groups-manager, needs-restarting, playground,
release-notification, repoclosure, repodiff, repograph, repomanage, reposync,
supportinfo
```

```

DNF version: 4.12.0
cachedir: /var/cache/dnf
Last metadata expiration check: 0:00:02 ago on Wed Mar 1 23:40:15 2023.
Repo-id           : amazonlinux
Repo-name         : Amazon Linux 2023 repository
Repo-status       : enabled
Repo-revision     : 1677203368
Repo-updated      : Fri Feb 24 01:49:28 2023
Repo-pkgs         : 12632
Repo-available-pkgs: 12632
Repo-size         : 12 G
Repo-mirrors      : https://al2023-repos-us-west-2-de612dc2.s3.dualstack.us-
west-2.amazonaws.com/core/mirrors/2023.0.20230222/x86_64/mirror.list
Repo-baseurl      : https://al2023-repos-us-west-2-de612dc2.s3.dualstack.us-
west-2.amazonaws.com/core/guids/
cf9296325a6c46ff40c775a8e2d632c4c3fd9d9164014ce3304715d61b90ca8e/x86_64/
                  : (0 more)
Repo-expire       : 172800 second(s) (last: Wed Mar 1 23:40:15
                  : 2023)
Repo-filename     : /etc/yum.repos.d/amazonlinux.repo

Repo-id           : amazonlinux-debuginfo
Repo-name         : Amazon Linux 2023 repository - Debug
Repo-status       : disabled
Repo-mirrors      : https://al2023-repos-us-west-2-de612dc2.s3.dualstack.us-
west-2.amazonaws.com/core/mirrors/2023.0.20230222/debuginfo/x86_64/mirror.list
Repo-expire       : 21600 second(s) (last: unknown)
Repo-filename     : /etc/yum.repos.d/amazonlinux.repo

Repo-id           : amazonlinux-source
Repo-name         : Amazon Linux 2023 repository - Source packages
Repo-status       : disabled
Repo-mirrors      : https://al2023-repos-us-west-2-de612dc2.s3.dualstack.us-
west-2.amazonaws.com/core/mirrors/2023.0.20230222/SRPMS/mirror.list
Repo-expire       : 21600 second(s) (last: unknown)
Repo-filename     : /etc/yum.repos.d/amazonlinux.repo

Repo-id           : kernel-livepatch
Repo-name         : Amazon Linux 2023 Kernel Livepatch repository
Repo-status       : disabled
Repo-mirrors      : https://al2023-repos-us-west-2-de612dc2.s3.dualstack.us-
west-2.amazonaws.com/kernel-livepatch/mirrors/al2023/x86_64/mirror.list
Repo-expire       : 172800 second(s) (last: unknown)
Repo-filename     : /etc/yum.repos.d/kernel-livepatch.repo

```

```
Repo-id           : kernel-livepatch-source
Repo-name         : Amazon Linux 2023 Kernel Livepatch repository -
                  : Source packages
Repo-status       : disabled
Repo-mirrors      : https://al2023-repos-us-west-2-de612dc2.s3.dualstack.us-
west-2.amazonaws.com/kernel-livepatch/mirrors/al2023/SRPMS/mirror.list
Repo-expire       : 21600 second(s) (last: unknown)
Repo-filename     : /etc/yum.repos.d/kernel-livepatch.repo
Total packages: 12632
```

Note

If you don't add the `--verbose` option flag, the output only includes the `Repo-id`, `Repo-name`, and `Repo-status` information.

To add a yum repository to `/etc/yum.repos.d` directory:

1. Find the location of the `.repo` file. In this example, the `.repo` file is at <https://www.example.com/repository.repo>.
2. Add the repository with the `dnf config-manager` command.

```
$ sudo dnf config-manager --add-repo https://www.example.com/repository.repo
Loaded plugins: priorities, update-motd, upgrade-helper
adding repo from: https://www.example.com/repository.repo
grabbing file https://www.example.com/repository.repo to /etc/
yum.repos.d/repository.repo
repository.repo | 4.0 kB      00:00
repo saved to /etc/yum.repos.d/repository.repo
```

After you install a repository, you must enable it as described in the next procedure.

To enable a yum repository in `/etc/yum.repos.d`, use the `dnf config-manager` command with the `--enable` flag and `repository` name.

```
$ sudo dnf config-manager --enable repository
```

Note

To disable a repository, use the same command syntax, but replace `--enable` with `--disable` in the command.

Adding repositories with cloud-init

In addition to adding a repository using the previous method, you can also add a new repository using the `cloud-init` framework.

To add a new package repository, we recommend the use of the following template. Consider saving this file locally.

```
#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
```

Note

One advantage to using `cloud-init` is that you can add a `packages:` section to your configuration file. In this section, you can include the names of the packages that you want to install. You can install packages from either the default repository or the new repository that you added in the `cloud-config` file.

For more specific information about the structure of the YAML file, see [Adding a YUM repository](#) in the *cloud-init* documentation.

After you set up the YAML format file, you can run it in the `cloud-init` framework in the AWS CLI. Make sure to include the `--userdata` option and the name of the `.yaml` file to call the desired operations.

```
$ aws ec2 run-instances \
  --image-id \
```

```
resolve:ssm:/aws/service/ami-amazon-linux-latest/al2023-ami-kernel-default-x86_64 \  
--instance-type m5.xlarge \  
--region us-east-1 \  
--key-name aws-key-us-east-1 \  
--security-group-ids sg-004a7650 \  
--user-data file://cloud-config.yml
```

Kernel Live Patching on AL2023

You can use Kernel Live Patching for AL2023 to apply specific security vulnerability and critical bug patches to a running Linux kernel without rebooting or disrupting running applications. In addition, Kernel Live Patching can help improve your application's availability while applying these fixes until the system can be rebooted.

AWS releases two types of kernel live patches for AL2023:

- **Security updates** – Include updates for Linux common vulnerabilities and exposures (CVE). These updates are typically rated as *important* or *critical* using the Amazon Linux Security Advisory ratings. They generally map to a Common Vulnerability Scoring System (CVSS) score of 7 and higher. In some cases, AWS might provide updates before a CVE is assigned. In these cases, the patches might appear as bug fixes.
- **Bug fixes** – Include fixes for critical bugs and stability issues that aren't associated with CVEs.

AWS provides kernel live patches for an AL2023 kernel version for up to 3 months after its release. After this period, you must update to a later kernel version to continue to receive kernel live patches.

AL2023 kernel live patches are made available as signed RPM packages in the existing AL2023 repositories. The patches can be installed on individual instances using existing **DNF package manager** workflows. Or, they can be installed on a group of managed instances using AWS Systems Manager.


Kernel Live Patching on AL2023 is provided at no additional cost.

Topics

- [Limitations](#)
- [Supported configurations and prerequisites](#)
- [Work with Kernel Live Patching](#)

Limitations

While applying a kernel live patch, you can't perform hibernation, use advanced debugging tools (such as SystemTap, kprobes, and eBPF-based tools), or access `fttrace` output files used by the Kernel Live Patching infrastructure.

 **Note**

Due to technical limitations, some issues cannot be addressed with live patching. Because of that, these fixes will not be shipped in the kernel live patch package but only in the native kernel package update. You can install the native kernel package and [update and reboot](#) the system to activate the patches as usual.

Supported configurations and prerequisites

Kernel Live Patching is supported on Amazon EC2 instances and on-premises virtual machines that run AL2023.

To use Kernel Live Patching on AL2023, you must use the following:

- A 64-bit x86_64 or ARM64 architecture
- Kernel version 6.1 or 6.12

Policy requirements

To download packages from AL2023 repositories, Amazon EC2 needs access to service owned Amazon S3 buckets. If you are using a Amazon Virtual Private Cloud (VPC) endpoint for Amazon S3 in your environment, ensure that your VPC endpoint policy allows access to those public buckets. The following table describes the Amazon S3 bucket that Amazon EC2 might need to access for Kernel Live Patching.

S3 bucket ARN	Description
arn:aws:s3:::al2023-repos- <i>region</i> -de612dc2/ *	Amazon S3 bucket containing AL2023 repositories

Work with Kernel Live Patching

You can enable and use Kernel Live Patching on individual instances using the command line on the instance itself. Alternatively, you can enable and use Kernel Live Patching on a group of managed instances using AWS Systems Manager.

The following sections explain how to enable and use Kernel Live Patching on individual instances using the command line.

For more information about enabling and using Kernel Live Patching on a group of managed instances, see [Use Kernel Live Patching on AL2023 instances](#) in the *AWS Systems Manager User Guide*.

Topics

- [Enable Kernel Live Patching](#)
- [View the available kernel live patches](#)
- [Apply kernel live patches](#)
- [View the applied kernel live patches](#)
- [Disable Kernel Live Patching](#)

Enable Kernel Live Patching

Kernel Live Patching is disabled by default on AL2023. To use live patching, you must install the **DNF** plugin for Kernel Live Patching and enable the live patching functionality.

To enable Kernel Live Patching

1. Kernel live patches are available for AL2023 with kernel version 6.1. To check your kernel version, run the following command.

```
$ sudo dnf list kernel
```

2. Install the **DNF** plugin for Kernel Live Patching.

```
$ sudo dnf install -y kpatch-dnf
```

3. Enable the **DNF** plugin for Kernel Live Patching.

```
$ sudo dnf kernel-livepatch -y auto
```

This command also installs the latest version of the kernel live patch RPM from the configured repositories.

4. To confirm that the **DNF** plugin for kernel live patching installed successfully, run the following command.

When you enable Kernel Live Patching, an empty kernel live patch RPM is automatically applied. If Kernel Live Patching was successfully enabled, this command returns a list that includes the initial empty kernel live patch RPM (and another RPM setting up the DNF repository containing the livepatches).

```
$ sudo rpm -qa | grep kernel-livepatch
kernel-livepatch-repo-s3-2023.7.20250428-0.amzn2023.noarch
kernel-livepatch-6.1.134-150.224-1.0-0.amzn2023.x86_64
```

5. Install the **kpatch** package.

```
$ sudo dnf install -y kpatch-runtime
```

6. Update the **kpatch** service if it was previously installed.

```
$ sudo dnf upgrade kpatch-runtime
```

7. Start the **kpatch** service. This service loads all of the kernel live patches upon initialization or at boot.

```
$ sudo systemctl enable kpatch.service && sudo systemctl start kpatch.service
```

View the available kernel live patches

Amazon Linux security alerts are published to the Amazon Linux Security Center. For more information about the AL2023 security alerts, including alerts for kernel live patches, see the [Amazon Linux Security Center](#). Kernel live patches are prefixed with ALASLIVEPATCH. The Amazon Linux Security Center might not list kernel live patches that address bugs.

You can also discover the available kernel live patches for advisories and CVEs using the command line.

To list all available kernel live patches for advisories

Use the following command.

```
$ sudo dnf updateinfo list
Last metadata expiration check: 1:06:23 ago on Mon 13 Feb 2023 09:28:19 PM UTC.
ALAS2LIVEPATCH-2021-123    important/Sec. kernel-
livepatch-6.1.12-17.42-1.0-4.amzn2023.x86_64
ALAS2LIVEPATCH-2022-124    important/Sec. kernel-
livepatch-6.1.12-17.42-1.0-3.amzn2023.x86_64
```

To list all available kernel live patches for CVEs

Use the following command.

```
$ sudo dnf updateinfo list cves
Last metadata expiration check: 1:07:26 ago on Mon 13 Feb 2023 09:28:19 PM UTC.
CVE-2022-0123             important/Sec. kernel-livepatch-6.1.12-17.42-1.0-4.amzn2023.x86_64
CVE-2022-3210             important/Sec. kernel-livepatch-6.1.12-17.42-1.0-3.amzn2023.x86_64
```

Apply kernel live patches

You apply kernel live patches using the **DNF** package manager in the same way that you apply regular updates. The **DNF** plugin for Kernel Live Patching manages the kernel live patches that are available to be applied.

Tip

We recommend that you update your kernel regularly using Kernel Live Patching to ensure that it receives specific important and critical security fixes until the system can be rebooted. Please also check if additional fixes have been made available to the native kernel package that cannot be deployed as live patches and [update and reboot](#) into the kernel update for those cases.

You can choose to apply a specific kernel live patch, or to apply any available kernel live patches along with your regular security updates.

To apply a specific kernel live patch

1. Get the kernel live patch version using one of the commands described in [View the available kernel live patches](#).
2. Apply the kernel live patch for your AL2023 kernel.

```
$ sudo dnf install kernel-livepatch-kernel_version-package_version.amzn2023.x86_64
```

For example, the following command applies a kernel live patch for AL2023 kernel version 6.1.12-17.42

```
$ sudo dnf install kernel-livepatch-6.1.12-17.42-1.0-4.amzn2023.x86_64
```

To apply any available kernel live patches along with your regular security updates

Use the following command.

```
$ sudo dnf upgrade --security
```

Omit the `--security` option to include bug fixes.

Important

- The kernel version isn't updated after applying kernel live patches. The version is only updated to the new version after the instance is rebooted.
- An AL2023 kernel receives kernel live patches for 3 months. After this period, no new kernel live patches are released for that kernel version.
- To continue to receive kernel live patches after 3 months, you must reboot the instance to move to the new kernel version. The instance continues to receive kernel live patches for the next 3 months after you update it.
- To check the support window for your kernel version, run the following command:

```
$ sudo dnf kernel-livepatch support
```

The current version of the Linux kernel you are running will no longer receive live patches after 2025-07-22.

View the applied kernel live patches

To view the applied kernel live patches

Use the following command.

```
$ sudo kpatch list
Loaded patch modules:
livepatch_CVE_2022_36946 [enabled]

Installed patch modules:
livepatch_CVE_2022_36946 (6.1.57-29.131.amzn2023.x86_64)
livepatch_CVE_2022_36946 (6.1.57-30.131.amzn2023.x86_64)
```

The command returns a list of the loaded and installed security update kernel live patches. The following is example output.

Note

A single kernel live patch can include and install multiple live patches.

Disable Kernel Live Patching

If you no longer need to use Kernel Live Patching, you can disable it at any time.

- Disable the use of livepatches:

1. Disable the plugin:

```
$ sudo dnf kernel-livepatch manual
```

2. Disable the kpatch service:

```
$ sudo systemctl disable --now kpatch.service
```

- Fully remove the livepatch tools:

1. Remove the plugin:

```
$ sudo dnf remove kpatch-dnf
```

2. Remove kpatch-runtime:

```
$ sudo dnf remove kpatch-runtime
```

3. Remove any installed livepatches:

```
$ sudo dnf remove kernel-livepatch\*
```

Updating the Linux kernel on AL2023

Topics

- [Linux kernel versions on AL2023](#)
- [Updating AL2023 to kernel 6.12](#)
- [AL2023 kernels - Frequently Asked Questions](#)

Linux kernel versions on AL2023

AL2023 regularly includes new kernel versions based on Long-Term Support (LTS) versions of the Linux kernel.

AL2023 was originally released in March 2023 with kernel 6.1.

In April 2025, AL2023 added support for Linux kernel 6.12. This kernel added new features including EEVDF scheduling, FUSE passthrough I/O support, a new Futex API, and improvements in eBPF. Kernel 6.12 also allows a userspace program to secure itself at runtime using user-space shadow stacks and memory sealing.

Updating AL2023 to kernel 6.12

You can run AL2023 with kernel 6.12 either by selecting an AMI with kernel 6.12 pre-installed or by upgrading an existing AL2023 EC2 instance.

Running an AL2023 kernel 6.12 AMI

You may select to run an AL2023 AMI with kernel 6.12 pre-installed through the AWS Console or by querying SSM for specific parameters. The SSM keys to query start with `/aws/service/ami-amazon-linux-latest/` followed by one of

- `al2023-ami-kernel-6.12-arm64` for arm64 architecture
- `al2023-ami-minimal-kernel-6.12-arm64` for arm64 architecture (minimal AMI)
- `al2023-ami-kernel-6.12-x86_64` for x86_64 architecture
- `al2023-ami-minimal-kernel-6.12-x86_64` for x86_64 architecture (minimal AMI)

Please see [Launching AL2023 using the SSM parameter and AWS CLI](#) for details on selecting AL2023 AMIs.

Updating an AL2023 instance to kernel 6.12

You can in-place upgrade a running AL2023 instance to kernel 6.12 with the following steps:

1. Install the `kernel6.12` package:

```
$ sudo dnf install -y kernel6.12
```

2. Get the latest version of the `kernel6.12` package:

```
$ version=$(rpm -q --qf '%{version}-%{release}.%{arch}\n' kernel6.12 | sort -V |  
tail -1)
```

3. Make the new `kernel6.12` your default kernel:

```
$ sudo grubby --set-default "/boot/vmlinuz-$version"
```

4. Reboot your system:

```
$ sudo reboot
```

5. Uninstall kernel 6.1:

```
$ sudo dnf remove -y kernel
```

6. Replace extra kernel packages with their `kernel6.12` equivalents:

```
$ declare -A pkgs  
$ pkgs=(  
  [bpftool]=bpftool6.12  
  [kernel-debuginfo]=kernel6.12-debuginfo  
  [kernel-debuginfo-common]=kernel6.12-debuginfo-common  
)
```

```
[kernel-headers]=kernel6.12-headers
[kernel-libbpf]=kernel6.12-libbpf
[kernel-libbpf-devel]=kernel6.12-libbpf-devel
[kernel-libbpf-static]=kernel6.12-libbpf-static
[kernel-modules-extra-common]=kernel6.12-modules-extra-common
[kernel-tools]=kernel6.12-tools
[kernel-tools-devel]=kernel6.12-tools-devel
[perf]=perf6.12
[python3-perf]=python3-perf6.12
)
$ for pkg in "${!pkgs[@]}"; do
    rpm -q $pkg && sudo dnf -y swap $pkg "${pkgs["$pkg"]}" ;
done
```

7. (Optional) Uninstall kernel-devel for kernel 6.1:

```
$ rpm -q kernel-devel && sudo dnf remove -y kernel-devel
```

Downgrading from kernel 6.12 to kernel 6.1

If at any point in time you need to downgrade back to kernel 6.1, use the following steps:

1. Replace extra kernel6.12 packages with their kernel 6.1 equivalents:

```
$ declare -A pkgs
$ pkgs=(
    [bpftool]=bpftool6.12
    [kernel-debuginfo]=kernel6.12-debuginfo
    [kernel-debuginfo-common]=kernel6.12-debuginfo-common
    [kernel-headers]=kernel6.12-headers
    [kernel-libbpf]=kernel6.12-libbpf
    [kernel-libbpf-devel]=kernel6.12-libbpf-devel
    [kernel-libbpf-static]=kernel6.12-libbpf-static
    [kernel-modules-extra-common]=kernel6.12-modules-extra-common
    [kernel-tools]=kernel6.12-tools
    [kernel-tools-devel]=kernel6.12-tools-devel
    [perf]=perf6.12
    [python3-perf]=python3-perf6.12
)
$ for pkg in "${!pkgs[@]}"; do
    rpm -q "${pkgs["$pkg"]}" && sudo dnf -y swap "${pkgs["$pkg"]}" $pkg ;
```

```
done
```

2. Install the kernel package:

```
$ sudo dnf install -y kernel
```

3. Get the latest version of the kernel package:

```
$ version=$(rpm -q --qf '%{version}-%{release}.%{arch}\n' kernel | sort -V | tail -1)
```

4. Make kernel 6.1 your default kernel:

```
$ sudo grubby --set-default "/boot/vmlinuz-$version"
```

5. Reboot your system:

```
$ sudo reboot
```

6. Uninstall kernel 6.12:

```
$ sudo dnf remove -y kernel6.12
```

AL2023 kernels - Frequently Asked Questions

1. Do I need to reboot after a kernel update?

Every change to the running kernel requires a reboot.

2. How do I keep kernels up-to-date across multiple instances?

Amazon Linux does not provide facilities to manage fleets of instances. We recommend you patch large fleets using tools like [AWS Systems Manager](#).

3. How do I check which kernel version I am running right now?

Execute this command on your AL2023 instance:

```
$ uname -r
```

4. How do I install kernel headers, development packages, and extra modules for kernel 6.12?

Please run:

```
$ sudo dnf install -y kernel6.12-modules-extra-$(uname -r) kernel6.12-headers-$(uname -r) kernel6.12-devel-$(uname -r)
```

Getting started with programming runtimes on AL2023

AL2023 provides different versions of some language runtimes. We work with upstream projects that support multiple versions at the same time. Find information about how to install and manage these name-versioned packages using the `dnf` command to search and install these packages.

The following topics outline how each language ecosystem exists in AL2023.

Topics

- [C, C++, and Fortran in AL2023](#)
- [Go in AL2023](#)
- [Java in AL2023](#)
- [Node.js in AL2023](#)
- [Perl in AL2023](#)
- [PHP in AL2023](#)
- [Python in AL2023](#)
- [Rust in AL2023](#)
- [TypeScript in AL2023](#)

C, C++, and Fortran in AL2023

AL2023 includes both the GNU Compiler Collection (GCC) and the Clang frontend for LLVM (Low Level Virtual Machine).

The major version of GCC will remain constant throughout the life time of AL2023. Minor releases bring bug fixes and might be included in AL2023 releases. Other bug, performance, and security fixes might be backported to the major version of GCC that ships in AL2023.

AL2023 includes version 11 of GCC as the default compiler with the C (`gcc`), C++ (`g++`), and Fortran (`gfortran`) frontends. Additionally, AL2023 provides GCC version 14 as an optional alternative compiler that can be installed alongside the default version.

AL2023 does not enable the Ada (`gnat`), Go (`gcc-go`), Objective-C, or Objective-C++ frontends.

The default compiler flags that AL2023 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

When `gcc --version` is invoked, a version string such as `gcc (GCC) 11.3.1 20221121 (Red Hat 11.3.1-4)` 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 AL2023 and its relationship to other Linux distributions, see [Relationship to Fedora](#).

For more information on the compiler triplet change in AL2023 compared to AL2, see [Compiler Triplet](#).

Topics

- [GCC 14](#)
- [Language Standard Versions Comparison](#)

GCC 14

AL2023 provides GCC 14 as an optional compiler that can be installed alongside the default GCC 11. GCC 14 includes the latest language features and optimizations, making it suitable for projects that require newer C, C++, or Fortran standards support.

To install GCC 14, use the following command:

```
sudo dnf install gcc14 gcc14-c++ gcc14-gfortran
```

The GCC 14 compilers are installed with version-specific command names to avoid conflicts with the default GCC 11:

- gcc14-gcc - C compiler
- gcc14-g++ - C++ compiler
- gcc14-gfortran - Fortran compiler


Example usage:

```
gcc14-gcc -o myprogram myprogram.c
gcc14-g++ -o mycppprogram mycppprogram.cpp
gcc14-gfortran -o myfortranprogram myfortranprogram.f90
```

You can verify the installed version by running:

```
gcc14-gcc --version
```

This will display version information similar to: gcc14-gcc (GCC) 14.2.1 20250110 (Red Hat 14.2.1-7)

 **Note**

Both GCC 11 and GCC 14 can be installed simultaneously on the same system. The default gcc, g++, and gfortran commands will continue to use GCC 11, while GCC 14 is accessed through the version-specific commands.

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
AL2 with GCC 7 (default)	C11 (201112L)	C++14 (201402L)	Fortran 2008
AL2 with GCC 10 (optional)	C17/C18 (201710L)	C++14 (201402L)	Fortran 2008

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 2008

Key improvements by GCC version:

- **GCC 10 vs GCC 7:** Upgraded default C standard from C11 to C17/C18, added support for C++20 features, and improved optimization capabilities.
- **GCC 11 vs GCC 10:** Upgraded default C++ standard from C++14 to C++17, enhanced C++20 support, and added experimental C++23 features.
- **GCC 14 vs GCC 11:** Added full C23 standard support, enhanced C++23 features, improved optimization, and better standards compliance.

Supported language standards:

- **C Standards:** All versions support C90, C99, C11, and C17/C18. GCC 10+ supports C2x (draft C23), while GCC 14 provides full C23 support.
- **C++ Standards:** All versions support C++98, C++03, C++11, C++14, C++17, and C++20. GCC 11+ provides experimental C++23 support, with GCC 14 offering enhanced C++23 features.
- **Fortran Standards:** All versions primarily support Fortran 2008, with varying levels of Fortran 2018 features depending on the GCC version.

Note

While the default standards remain consistent between GCC 11 and 14, GCC 14 provides significantly improved language feature support, better optimization, enhanced diagnostics, and more complete implementation of newer standards when explicitly requested using `-std=` flags.

Go in AL2023

You might want to build your own code written in [Go](#) on Amazon Linux, and might want to use a toolchain provided with AL2023. Similar to AL2, AL2023 will update the Go toolchain throughout the life of the operating system. This might be in response to any CVE in the toolchain we ship, or as part of a quarterly release.

Go is a relatively fast moving language. There might be a situation where existing applications written in Go have to adapt to new versions of the Go toolchain. For more information about Go, see [Go 1 and the Future of Go Programs](#).

Although AL2023 will incorporate new versions of the Go toolchain during its life, this will not be in lockstep with the upstream Go releases. Therefore, using the Go toolchain provided in AL2023 might not be suitable if you want to build Go code using cutting-edge features of the Go language and standard library.

During the lifetime of AL2023, previous package versions are not removed from the repositories. If a previous Go toolchain is required, you can choose to forgo the bug and security fixes of newer Go toolchains and install an previous version from the repositories using the same mechanisms available for any RPM.

If you want to build your own Go code on AL2023, you can use the Go toolchain included in AL2023 with the knowledge that this toolchain might move forward through the lifetime of AL2023.

AL2023 Lambda functions written in Go

As Go compiles to native code, Lambda treats Go as a custom runtime. You can use the provided `al2023` runtime to deploy Go functions on AL2023 to Lambda.

For more information, see [Building Lambda functions with Go](#) in the *AWS Lambda Developer Guide*.

Java in AL2023

AL2023 provides several versions of [Amazon Corretto](#) to support Java based workloads. All Java based packages included in AL2023 are built with Amazon Corretto 17.

Corretto is a build of the Open Java Development Kit (OpenJDK) with long-term support from Amazon. Corretto is certified using the Java Technical Compatibility Kit (TCK) to ensure that it meets the Java SE standard and is available on Linux, Windows, and macOS.

There is an [Amazon Corretto](#) package available for each of Corretto 1.8.0, Corretto 11, and Corretto 17.

Each Corretto version in AL2023 is supported for the same period of time as the Corretto version is, or until the end of life of AL2023, whichever is sooner. For more information, see [Amazon Linux package support statements](#) and the [Amazon Corretto FAQs](#).

Node.js in AL2023

[Node.js](#) in AL2023 is represented by versions 20, 22 and 24. Amazon Linux follows the upstream [support schedule](#) and a support status of any Node.js version can always be checked on the [Package support status page](#). All supported Node.js versions are namespaced and can be installed on the same system simultaneously. Namespacing ensures that each Node.js installation is unique within the file system. This is achieved by renaming key directories and files based on the runtime version. The actual executable names will look like *node-{MAJOR_VERSION}* or *npm-{MAJOR_VERSION}*. However, only one Node.js version can be active at a time. This active version provides the default directories and file names, such as *node*, *npm* or */usr/lib/node_modules*, 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 Node.js version. This flexibility enables software that uses *node* in the shebang to select the desired version when invoked. However, when a specific version of Node.js is required, persistence of the version can be achieved by calling the namespaced executable (e.g., *node-20* or *node-22*), which will always use the specified version of the runtime. Additionally, the namespaced executables of the *npm* tool, such as *npm-20* or *npm-22*, are always associated with the corresponding Node.js version, regardless of the currently active runtime.

Node.js is distributed as several namespaced packages which begin with "nodejs{MAJOR_VERSION}". These packages provide *node*, a compatible version of the *npm* tool, documentation, libraries, and more. For example, *node* and *npm* of the Node.js 22 are provided by the *nodejs22* and *nodejs22-npm* packages, respectively.

The *alternatives* tool provides a single command for switching between Node.js versions. By default, *alternatives* is configured to be in auto mode, which uses priorities to determine the currently active Node.js version. However, you can activate any installed version at any time. Currently, all supported versions of Node.js have equal priority, meaning the first installed version will be activated automatically.

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
```

Perl in AL2023

AL2023 provides version 5.32 of the [Perl](#) programming language.

Although Perl has provided a high degree of language compatibility as part of Perl 5 releases over the past decades, Amazon Linux is not expected to move from Perl 5.32 during the AL2023 release. Amazon Linux will continue to security patch Perl for the lifetime of AL2023 in accordance with our [package support statements](#).

Perl modules in AL2023

Various Perl modules are packaged as RPMs in AL2023. Although there are many Perl modules available as RPMs, Amazon Linux does not aim to package every possible Perl module. Modules packaged as RPMs might be relied upon by other operating system RPM packages, so Amazon Linux will prioritize those security patches over pure feature updates.

AL2023 also includes CPAN so that Perl developers can use the idiomatic package manager for Perl modules.

PHP in AL2023

AL2023 currently provides the [PHP](#) programming language, versions 8.1, 8.2, 8.3, and 8.4. Each version is supported for same period of time as upstream PHP. For more information, see [Package support statements](#).

Migrating from older PHP versions

The upstream PHP community put together comprehensive migration documentation for moving:

- [from PHP 8.3.x to PHP 8.4.x](#)
- [from PHP 8.2.x to PHP 8.3.x](#)
- [from PHP 8.1.x to PHP 8.2.x](#)
- [from PHP 8.0.x to PHP 8.1.x](#)

AL2 includes PHP 8.0, 8.1, and 8.2 in `amazon-linux-extras` enabling an easy upgrade path to AL2023.

Migrating from PHP 7.x versions

Note

The [PHP](#) project maintains a list and schedule of [supported versions](#), as well as a list of [unsupported branches](#).

When AL2023 was released, all 7.x and 5.x versions of [PHP](#) were unsupported by the PHP community, and were not included as options in AL2023.

The upstream PHP community put together [comprehensive migration documentation for moving to PHP 8.0 from PHP 7.4](#). Combined with the documentation referenced in the previous section on migrating to PHP 8.1 and PHP 8.2, you can migrate your PHP based application to modern PHP.

Note

AL2 includes PHP 7.1, 7.2, 7.3, and 7.4 in `amazon-linux-extras`. It is important to note that all of these Extras are end-of-life and are not guaranteed to get any further security updates.

PHP modules in AL2023

AL2023 includes many PHP modules that are included in PHP Core. AL2023 does not aim to include all of the packages in the [PHP Extension Community Library \(PECL\)](#).

Python in AL2023

AL2023 removed Python 2.7 and any components requiring Python are now written to work with Python 3.

AL2023 makes Python 3 available as `/usr/bin/python3` to retain compatibility with customer code, as well as Python code shipped with AL2023, this will remain as Python 3.9 for the life of AL2023.

The version of python that `/usr/bin/python3` points to is considered the *system Python* and for AL2023 this is Python 3.9.

Newer versions of Python, such as Python 3.11, are made available as packages in AL2023 and are supported for the lifetime of the upstream versions. For information on how long Python 3.11 is supported, see [Python 3.11](#).

Multiple versions of Python can be installed simultaneously on AL2023. Although `/usr/bin/python3` will always be Python 3.9, each version of Python is namespaced and can be found by its version number. For example, if `python3.11` is installed, then `/usr/bin/python3.11` will exist alongside `/usr/bin/python3.9` and the `/usr/bin/python3` symlink to `/usr/bin/python3.9`.

Note

Do not change what the `/usr/bin/python3` symlink points to because this might break the core functionality of AL2023.

Python modules in AL2023

Various Python modules are packaged as RPMs in AL2023. Typically, RPMs for Python modules will be built targeting only the system version of Python.

Rust in AL2023

You might want to build code written in [Rust](#) on Amazon Linux, and might want to use a toolchain provided with AL2023.

Similar to AL2, AL2023 will update the Rust toolchain throughout the life of the operating system. This might be in response to any CVE in the toolchain we ship, or as part of a quarterly release.

[Rust](#) is a relatively fast moving language, with new releases on approximately a six-week cadence. These releases might add new language or standard library features. Although AL2023 will incorporate new versions of the Rust toolchain during its life, this will not be in lockstep with the upstream Rust releases. Therefore, using the Rust toolchain provided in AL2023 might not be suitable if you want to build Rust code using cutting-edge features of the Rust language.

During the lifetime of AL2023, old package versions are not removed from the repositories. If an older Rust toolchain is required, you can choose to forgo bug and security fixes of newer Rust toolchains and install an older version from the repositories using the same mechanisms available for any RPM.

If you want to build your own Rust code on AL2023, you can use the Rust toolchain included in AL2023 with the knowledge that this toolchain might move forward through the lifetime of AL2023.

AL2023 Lambda functions written in Rust

Because Rust compiles to native code, Lambda treats Rust as a custom runtime. You can use the provided `.al2023` runtime to deploy Rust functions on AL2023 to Lambda.

For more information, see [Building Lambda functions with Rust](#) in the *AWS Lambda Developer Guide*.

TypeScript in AL2023

Note

This document provides the essential information on TypeScript and its Node.js-based execution environment. It also covers a typical development workflow and explains how TypeScript is packaged in AL2023 to deliver a consistent and reproducible development environment.

[TypeScript](#) (TS) is a programming language based on JavaScript (JS) that offers all JS's features, and also [extends it with a type system](#). In a typical scenario, programs written in TS are translated into the JS code first, and then executed as any other regular JS program by Node.js. In TS's specific terminology, this translation process is called ["compilation" and is performed by a "compiler"](#), called *tsc*. The *tsc* compiler itself is written in JS, thus to run, it also needs a JS runtime environment, such as Node.js. Unlike some other JS runtime environments, Node.js currently has only experimental and lightweight TS support. A full TS support, including type checking, still requires using third-party packages, such as the [typescript](#). The expected way to get *tsc* (the TS compiler) for the Node.js runtime environment is to install the *typescript* node module. This can be done using one of the package managers, typically *npm*. There are two ways to install the TS compiler using *npm*: globally and in a project. [The officially recommended method is to install](#) the TS compiler on a per-project basis, which ensures long-term consistency and reproducibility for projects. However, installing the TS compiler globally might still be useful, because it provides the same version for the entire host and its JS runtime, and thus for projects that don't have a TS compiler installed locally. This is precisely how RPM packages available on Amazon Linux, such as `nodejs20-typescript` or `nodejs22-typescript`, install a TS compiler: globally at the system level, and separately for each supported Node.js version.

The *tsc* doesn't directly depend on any Node.js version. The compiler expects a certain level of the runtime features, which are defined in a special file (*tsconfig.json*) via options such as [target](#) and [lib](#). The values of these options represent a version of the [ECMAScript](#) (ES) standard, which may (or may not) be supported by the JS runtime environment. Different versions of Node.js support different versions of ES standard. The more recent the version of Node.js, the higher and more complete the ES standard version supported. If *tsconfig.json* does not exist in the root directory of a project, then the default set of configuration options will be used. The compatibility table with the different versions of Node.js and the supported features of various ES standard versions is available at [node.green](#). The *tsc* has more than 100 different options, which can be defined in

the *tsconfig.json*. Configuration chaining is also supported, when some configuration options are defined in another file and then included in the main file. This approach allows one to install a [Base TS Config](#) compatible with a certain version of Node.js, and then extend it with project-specific options. Fortunately, the Base TS Configs for Node.js are available as node modules which can be installed in a project folder using *npm*. Here is the config's source code for Node.js version [18](#), [20](#), and [22](#).

The Node.js based runtime design has a certain weakness: it supports only one version of the runtime on a host, and requires reproducibility and consistency of all dependencies at the project level. This led to the following common approach of using TypeScript: the TS compiler, TS base config for the current Node.js version, and all software dependencies are installed locally, inside a project. Although globally installed node modules are expected to be CLI tools only, such as *npm*, *tsc*, which is also a CLI tool, is rarely installed globally. Thankfully, global (system wide) and local (within a project) installations of *tsc* can co-exist with no issues and can also be different versions which are used independently. Note that a locally installed *tsc* should be executed using the *npm* tool, which is installed together with *npm*. Thus, even with a system TS compiler, users have the opportunity to choose versions of the runtime's components, such as Node.js (by switching the active version via alternatives), a TS compiler (by either installing it locally, or, globally and also switching the active version via alternatives), and configuring it for the specific needs.

Amazon Linux packages a TS compiler in the same way as other globally installed node modules, such as *npm*, on a per Node.js version basic. Packages and binaries are namespaced and contain the major version of Node.js as part of their names. The default executable name of the compiler, *tsc*, is managed at runtime by the alternatives tool and point the currently active version of Node.js for which it was installed and will be executed by. This selection doesn't depend on the current Node.js runtime version. It is possible to have *node* executable pointing to Node.js 20 and *tsc* configured to be interpreted by the Node.js 22. It is also possible to use the namespaced names of a TS compiler, e.g. *tsc-{MAJOR_VERSION}* independently to what the default *tsc* name is configured for.

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
```

An example of installing and using several versions of Node and a TS compiler on the same system:

```
# Check the AL2023 release
$ cat /etc/amazon-linux-release
Amazon Linux release 2023.9.20250929 (Amazon Linux)

# Install a TypeScript compiler for Node.js 20 and 22
# Node.js 20 and 22 will be installed automatically
$ sudo dnf install -qy nodejs20-typescript nodejs22-typescript

# Check what was installed
$ rpm -q nodejs20 nodejs20-typescript nodejs22 nodejs22-typescript
nodejs20-20.19.5-1.amzn2023.0.1.x86_64
nodejs20-typescript-5.9.2-1.amzn2023.0.1.noarch
nodejs22-22.19.0-1.amzn2023.0.1.x86_64
nodejs22-typescript-5.9.2-1.amzn2023.0.1.noarch

# Check the active version of Node.js - it is version 20
$ alternatives --display node
node - status is auto.
  link currently points to /usr/bin/node-20
/usr/bin/node-20 - priority 100
  slave npmrc: /usr/lib/nodejs20/lib/node_modules/npm/npmrc
  slave npm: /usr/bin/npm-20
  slave npx: /usr/bin/npx-20
  slave node_modules: /usr/lib/nodejs20/lib/node_modules
```

```
/usr/bin/node-22 - priority 100
slave npmrc: /usr/lib/nodejs22/lib/node_modules/npm/npmrc
slave npm: /usr/bin/npm-22
slave npx: /usr/bin/npx-22
slave node_modules: /usr/lib/nodejs22/lib/node_modules
Current 'best' version is /usr/bin/node-20.

# Check the active JS runtime version for TypeScript
# Currently, the tsc compiler will be executed by Node.js 22
$ alternatives --display tsc
tsc - status is auto.
  link currently points to /usr/bin/tsc-22
/usr/bin/tsc-22 - priority 100
  slave tsserver: /usr/bin/tsserver-22
/usr/bin/tsc-20 - priority 100
  slave tsserver: /usr/bin/tsserver-20
Current 'best' version is /usr/bin/tsc-22.

# Check versions printed by executables
$ node -v
v20.19.5

$ tsc -v
Version 5.9.2

# while the node is 20, tsc is executed by node 22 anyway
$ head -1 /usr/bin/tsc
#!/usr/bin/node-22

# However, instead of default executable names, e.g. node or tsc,
# we can use namespaced names to target any installed version
$ node-20 -v
v20.19.5

$ node-22 -v
v22.19.0

$ tsc-20 -v
Version 5.9.2

$ tsc-22 -v
Version 5.9.2

$ head -1 /usr/bin/tsc-20
```

```
#!/usr/bin/node-20  
  
$ head -1 /usr/bin/tsc-22  
#!/usr/bin/node-22
```

AL2023 Reserved Users and Groups

AL2023 pre-allocates certain users and groups during both the provisioning of the image and during the installation of certain packages. The users, groups, and their associated UIDs and GIDs are listed here to prevent conflicts.

Topics

- [List of AL2023 Reserved Users](#)
- [List of AL2023 Reserved Groups](#)

List of AL2023 Reserved Users

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

User name	UID
squid	23
named	25
postgres	26
mysql	27
nscd	28
nscd	28
rpcuser	29
rpc	32
mailnull	47
apache	48
smmsp	51
tomcat	53
ldap	55
tss	59
nslcd	65
avahi	70
tcpdump	72
sshd	74
radvd	75
dbus	81

User name	UID
postfix	89
dovecot	97
stapusr	156
stapsys	157
stapdev	158
avahi-autoipd	170
pulse	171
rtkit	172
sanlock	179
systemd-network	192
systemd-resolve	193
uuidd	961
stap-server	962
systemd-journal-remote	963
redis6	970
pesign	971
smtpq	972
smtpd	973
nginx	974
munge	975

User name	UID
memcached	976
sphinx	977
haproxy	978
flatpak	979
debuginfod	980
dovnull	981
dnsmasq	982
unbound	983
clamscan	984
clamilt	985
clamupdate	986
colord	987
ods	988
aws-kinesis-agent-user	989
saslauth	990
cwagent	991
polkitd	992
ec2-instance-connect	993
chrony	994
systemd-timesync	995

User name	UID
systemd-coredump	996
libstoragemgmt	997
systemd-oom	999
ec2-user	1000
nobody	65534

Listed by Name

User name	UID
adm	3
apache	48
avahi	70
avahi-autoipd	170
aws-kinesis-agent-user	989
bin	1
chrony	994
clamilt	985
clamscan	984
clamupdate	986
colord	987
cwagent	991

User name	UID
daemon	2
dbus	81
debuginfod	980
dnsmasq	982
dovecot	97
dovnull	981
ec2-instance-connect	993
ec2-user	1000
flatpak	979
ftp	14
games	12
halt	7
haproxy	978
ldap	55
libstoragemgmt	997
lp	4
mail	8
mailnull	47
memcached	976
munge	975

User name	UID
mysql	27
named	25
nginx	974
nobody	65534
nscd	28
nscd	28
nslcd	65
ods	988
operator	11
pesign	971
polkitd	992
postfix	89
postgres	26
pulse	171
radvd	75
redis6	970
root	0
rpc	32
rpcuser	29
rtkit	172

User name	UID
sanlock	179
saslauth	990
shutdown	6
smmsp	51
smtpd	973
smtpq	972
sphinx	977
squid	23
sshd	74
stap-server	962
stapdev	158
stapsys	157
stapusr	156
sync	5
systemd-coredump	996
systemd-journal-remote	963
systemd-network	192
systemd-oom	999
systemd-resolve	193
systemd-timesync	995

User name	UID
tcpdump	72
tomcat	53
tss	59
unbound	983
uudd	961

List of AL2023 Reserved Groups

Group name	GID
root	0
bin	1
daemon	2
sys	3
adm	4
tty	5
disk	6
disk	6
lp	7
mem	8
kmem	9
wheel	10

Group name	GID
cdrom	11
mail	12
mail	12
man	15
dialout	18
floppy	19
games	20
slocate	21
utmp	22
squid	23
named	25
postgres	26
mysql	27
nscd	28
nscd	28
rpcuser	29
rpc	32
tape	33
utempter	35
kvm	36

Group name	GID
video	39
mailnull	47
apache	48
ftp	50
smmsp	51
tomcat	53
lock	54
ldap	55
tss	59
audio	63
avahi	70
tcpdump	72
sshd	74
radvd	75
saslauth	76
dbus	81
screen	84
wbpriv	88
postfix	89
postdrop	90

Group name	GID
dovecot	97
users	100
input	104
render	105
sgx	106
mock	135
stapusr	156
stapusr	156
stapsys	157
stapsys	157
stapdev	158
stapdev	158
avahi-autoipd	170
pulse	171
rtkit	172
sanlock	179
systemd-journal	190
systemd-network	192
systemd-resolve	193
usbmon	959

Group name	GID
wireshark	960
uidd	961
stap-server	962
systemd-journal-remote	963
usershares	964
redis6	965
pesign	966
smtpq	967
smtpd	968
nginx	969
munge	970
memcached	971
sphinx	972
tracing	973
haproxy	974
flatpak	975
debuginfod	976
dovnull	977
dnsmasq	978
unbound	979

Group name	GID
clamscan	980
clamilt	981
virusgroup	982
virusgroup	982
virusgroup	982
clamupdate	983
printadmin	984
colord	985
ods	986
docker	987
aws-kinesis-agent-user	988
cwagent	989
pulse-rt	990
pulse-access	991
ec2-instance-connect	993
chrony	994
systemd-timesync	995
systemd-coredump	996
libstoragegmt	997
ssh_keys	998

Group name	GID
systemd-oom	999
ec2-user	1000
ne	1001
polkitd	9920
nobody	65534

Listed by Name

Group name	GID
adm	4
apache	48
audio	63
avahi	70
avahi-autoipd	170
aws-kinesis-agent-user	988
bin	1
cdrom	11
chrony	994
clamilt	981
clamscan	980
clamupdate	983

Group name	GID
colord	985
cwagent	989
daemon	2
dbus	81
debuginfod	976
dialout	18
disk	6
disk	6
dnsmasq	978
docker	987
dovecot	97
dovnull	977
ec2-instance-connect	993
ec2-user	1000
flatpak	975
floppy	19
ftp	50
games	20
haproxy	974
input	104

Group name	GID
kmem	9
kvm	36
ldap	55
libstoragegmt	997
lock	54
lp	7
mail	12
mail	12
mailnull	47
man	15
mem	8
memcached	971
mock	135
munge	970
mysql	27
named	25
ne	1001
nginx	969
nobody	65534
nscd	28

Group name	GID
nscd	28
ods	986
pesign	966
polkitd	9920
postdrop	90
postfix	89
postgres	26
printadmin	984
pulse	171
pulse-access	991
pulse-rt	990
radvd	75
redis6	965
render	105
root	0
rpc	32
rpcuser	29
rtkit	172
sanlock	179
saslauth	76

Group name	GID
screen	84
sgx	106
slocate	21
smmsp	51
smtpd	968
smtpq	967
sphinx	972
squid	23
ssh_keys	998
sshd	74
stap-server	962
stapdev	158
stapdev	158
stapsys	157
stapsys	157
stapusr	156
stapusr	156
sys	3
systemd-coredump	996
systemd-journal	190

Group name	GID
systemd-journal-remote	963
systemd-network	192
systemd-oom	999
systemd-resolve	193
systemd-timesync	995
tape	33
tcpdump	72
tomcat	53
tracing	973
tss	59
tty	5
unbound	979
usbmon	959
users	100
usershares	964
utempter	35
utmp	22
uudd	961
video	39
virusgroup	982

Group name	GID
virusgroup	982
virusgroup	982
wbpriv	88
wheel	10
wireshark	960

List of codecs available in AL2023

AL2023 provides a selection of multimedia codecs 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
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
fdk-aac-free	An open-source implementation of the AAC (Advanced Audio Codec) standard, providing high-quality audio compression for MP3 alternatives like streaming or file storage
webrtc-audio-processing	A library for audio processing used in WebRTC (Web Real-Time Communication), offering features like noise suppression, echo cancellation, and gain control
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

Codec	Description
libsndfile	A library for reading and writing audio files in various formats (such as WAV, AIFF, and FLAC), commonly used in audio processing and manipulation tools
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.
svt-av1	An open source, high-performance implementation of the AV1 video codec. It provides scalable encoding and decoding for AV1 video, optimized for modern CPUs and parallel processing.
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.
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.

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.
x265	The primary objective of x265 is to become the best H.265/HEVC encoder available anywhere, offering the highest compression efficiency and the highest performance on a wide variety of hardware platforms. This package contains the command line encoder.

Security and Compliance in Amazon Linux 2023

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](#)

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 AL2023, 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.

Topics

- [Amazon Linux Security advisories for AL2023](#)
- [Listing applicable Advisories](#)
- [Applying security updates in-place](#)
- [Setting SELinux modes for AL2023](#)
- [Enable FIPS Mode on AL2023](#)
- [Enable FIPS Mode in an AL2023 Container](#)
- [Swap OpenSSL FIPS providers on AL2023](#)
- [AL2023 Kernel Hardening](#)

- [UEFI Secure Boot on AL2023](#)

Amazon Linux Security advisories for AL2023

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 AL2023 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](#).

For example, the [2021-001 - Amazon Linux Hotpatch Announcement for Apache Log4j](#) fit into an announcement rather than an advisory. In this announcement, Amazon Linux added a package to help customers mitigate a security issue in software that was not part of Amazon Linux.

The [Amazon Linux Security Center CVE Explorer](#) was also announced on ALAS announcements. For more information, see [New website for CVEs](#).

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 2023 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 AL2023 is in the process of being released, there has been an update to an Advisory after the most recent AL2023 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 AL2023](#) feature of Amazon Linux 2023 means that the RPM repository for a particular AL2023 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 AL2023 version *is not updated* after release. New or updated security advisories *will not be visible when looking at an older version of the AL2023 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 AL2023 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, ALASCORRETT08, 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.

As AL2023 does not use the Extras mechanism to package alternate versions of packages, there are currently only two RPM repositories, each of which has Advisories, the core repository and the livepatch repository. The livepatch repository is for [Kernel Live Patching on AL2023](#).

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 AL2023 version number (e.g. 2023.6.20241031) 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 AL2023 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 [AL2023 section of the Amazon Linux 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 AL2023, 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 more information on kernel live patching, see [Kernel Live Patching on AL2023](#).

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 version of `dnf` in AL2023 can parse both the AL2023 and AL2 repository formats, and thus the API can be used to examine advisory information for either OS version.

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

The dnf package manager has access to metadata describing what Advisories are fixed in what package versions. It can thus list what Advisories are applicable to an instance or container image.

Note

Tools such as [AWS Systems Manager](#) can use this functionality to show what updates are relevant across a fleet rather than just a single instance.

When listing updates, you can instruct dnf to look at the metadata of a particular AL2023 release, or the metadata from the latest release.

Note

Once an AL2023 release is made, it is immutable. Thus, new or updated advisories on the [Amazon Linux Security Center](#) are only added to the metadata of *new* releases of AL2023

We will now go through examples of looking at what advisories apply to some AL2023 container images. These commands all work on non-containerized environments such as EC2 instances.

Listing advisories in a specific version

In this example we are going to look at what advisories in the [2023.1.20230628](#) release are relevant in a container image of the [2023.0.20230315](#) release.

Note

This example uses the [2023.0.20230315](#) and [2023.1.20230628](#) releases, and these *are not* the latest release of AL2023 See the [AL2023 Release Notes](#) for the latest releases, which contain the latest security updates.

In this example we will be starting with a container image for the [2023.0.20230315](#) release.

First, we fetch this container image from the container registry. The `.0` at the end indicates the version of the image for a particular release; this image version is usually zero.

```
$ docker pull public.ecr.aws/amazonlinux/amazonlinux:2023.0.20230315.0
2023.0.20230315.0: Pulling from amazonlinux/amazonlinux
b76f3b09316a: Pull complete
Digest: sha256:94e7183b0739140dbd5b639fb7600f0a2299cec5df8780c26d9cb409da5315a9
Status: Downloaded newer image for public.ecr.aws/amazonlinux/
amazonlinux:2023.0.20230315.0
public.ecr.aws/amazonlinux/amazonlinux:2023.0.20230315.0
```

We can now spawn a shell inside the container, from which we will ask dnf to list what advisories are relevant to the packages installed in the container.

```
$ docker run -it public.ecr.aws/amazonlinux/amazonlinux:2023.0.20230315.0
bash-5.2#
```

The `dnf updateinfo` command is now used to display a summary of what advisories in the [2023.1.20230628](#) release are relevant to our installed packages.

```
$ dnf updateinfo --releasever=2023.1.20230628
Amazon Linux 2023 repository                42 MB/s | 15 MB      00:00
Last metadata expiration check: 0:00:02 ago on Mon Jul 22 20:24:24 2024.
Updates Information Summary: available
  8 Security notice(s)
    1 Important Security notice(s)
    5 Medium Security notice(s)
    2 Low Security notice(s)
```

To get a list of the advisories, the `--list` option can be given to `dnf updateinfo`.

```
$ dnf updateinfo --releasever=2023.1.20230628 --list
Last metadata expiration check: 0:01:22 ago on Mon Jul 22 20:24:24 2024.
ALAS2023-2023-193 Medium/Sec.    curl-minimal-8.0.1-1.amzn2023.x86_64
ALAS2023-2023-225 Medium/Sec.    glib2-2.74.7-688.amzn2023.0.1.x86_64
ALAS2023-2023-195 Low/Sec.       libcap-2.48-2.amzn2023.0.3.x86_64
ALAS2023-2023-193 Medium/Sec.    libcurl-minimal-8.0.1-1.amzn2023.x86_64
ALAS2023-2023-145 Low/Sec.       libgcc-11.3.1-4.amzn2023.0.3.x86_64
ALAS2023-2023-145 Low/Sec.       libgomp-11.3.1-4.amzn2023.0.3.x86_64
ALAS2023-2023-145 Low/Sec.       libstdc++-11.3.1-4.amzn2023.0.3.x86_64
ALAS2023-2023-163 Medium/Sec.    libxml2-2.10.4-1.amzn2023.0.1.x86_64
ALAS2023-2023-220 Important/Sec.  ncurses-base-6.2-4.20200222.amzn2023.0.4.noarch
ALAS2023-2023-220 Important/Sec.  ncurses-libs-6.2-4.20200222.amzn2023.0.4.x86_64
ALAS2023-2023-181 Medium/Sec.    openssl-libs-1:3.0.8-1.amzn2023.0.2.x86_64
```

```
ALAS2023-2023-222 Medium/Sec.      openssl-libs-1:3.0.8-1.amzn2023.0.3.x86_64
```

Listing advisories in the latest version

In this example we are going to look at what updates are available in the latest version of AL2023 if we launched a container of the [2023.4.20240319](#) release. At the time of writing, the latest release is [2023.5.20240708](#), so the listed updates in this example will be as of that release.

Note

This example uses the [2023.4.20240319](#) and [2023.5.20240708](#) releases, the latter being the latest release *at the time of writing*. For more information on the latest releases, see the [AL2023 Release Notes](#).

In this example we will be starting with a container image for the [2023.4.20240319](#) release.

First, we fetch this container image from the container registry. The .1 at the end indicates the version of the image for a particular release. While the image version is typically zero, this example uses a release where the image version is one.

```
$ docker pull public.ecr.aws/amazonlinux/amazonlinux:2023.4.20240319.1
2023.4.20240319.1: Pulling from amazonlinux/amazonlinux
6de065fda9a2: Pull complete
Digest: sha256:b4838c4cc9211d966b6ea158dacc9eda7433a16ba94436508c2d9f01f7658b4e
Status: Downloaded newer image for public.ecr.aws/amazonlinux/
amazonlinux:2023.4.20240319.1
public.ecr.aws/amazonlinux/amazonlinux:2023.4.20240319.1
```

We can now spawn a shell inside the container, from which we will check for updates.

```
$ docker run -it public.ecr.aws/amazonlinux/amazonlinux:2023.4.20240319.1
bash-5.2#
```

The `dnf updateinfo` command is now used to display a summary of what advisories in the latest release are relevant to our installed packages. At the time of writing, [2023.1.20230628](#) was the latest release.

```
$ dnf --releasever=latest updateinfo
```

```

Amazon Linux 2023 repository          76 MB/s | 25 MB    00:00
Last metadata expiration check: 0:00:04 ago on Mon Jul 22 20:59:54 2024.
Updates Information Summary: available
  9 Security notice(s)
    4 Important Security notice(s)
    4 Medium Security notice(s)
    1 Low Security notice(s)

```

To get a list of the advisories, the `--list` option can be given to `dnf updateinfo`.

```

$ dnf updateinfo --releasever=latest --list
Last metadata expiration check: 0:00:58 ago on Mon Jul 22 20:59:54 2024.
ALAS2023-2024-581 Low/Sec.      curl-minimal-8.5.0-1.amzn2023.0.3.x86_64
ALAS2023-2024-596 Medium/Sec.  curl-minimal-8.5.0-1.amzn2023.0.4.x86_64
ALAS2023-2024-576 Important/Sec. expat-2.5.0-1.amzn2023.0.4.x86_64
ALAS2023-2024-589 Important/Sec. glibc-2.34-52.amzn2023.0.10.x86_64
ALAS2023-2024-589 Important/Sec. glibc-common-2.34-52.amzn2023.0.10.x86_64
ALAS2023-2024-589 Important/Sec. glibc-minimal-langpack-2.34-52.amzn2023.0.10.x86_64
ALAS2023-2024-586 Medium/Sec.  krb5-libs-1.21-3.amzn2023.0.4.x86_64
ALAS2023-2024-581 Low/Sec.      libcurl-minimal-8.5.0-1.amzn2023.0.3.x86_64
ALAS2023-2024-596 Medium/Sec.  libcurl-minimal-8.5.0-1.amzn2023.0.4.x86_64
ALAS2023-2024-592 Important/Sec. libnghttp2-1.59.0-3.amzn2023.0.1.x86_64
ALAS2023-2024-640 Medium/Sec.  openssl-libs-1:3.0.8-1.amzn2023.0.12.x86_64
ALAS2023-2024-605 Medium/Sec.  python3-3.9.16-1.amzn2023.0.7.x86_64
ALAS2023-2024-616 Important/Sec. python3-3.9.16-1.amzn2023.0.8.x86_64
ALAS2023-2024-605 Medium/Sec.  python3-libs-3.9.16-1.amzn2023.0.7.x86_64
ALAS2023-2024-616 Important/Sec. python3-libs-3.9.16-1.amzn2023.0.8.x86_64

```

Applying security updates in-place

For an overview of applying updates, see [Applying security updates using DNF and repository versions](#). The `--security` option to `dnf upgrade` will restrict package updates to only those which have an Advisory. The remainder of this section will cover how to install only specific security updates.

Note

It is recommended to apply *all* updates available in a new AL2023 release. Picking just security updates, or only specific updates should be the exception rather than rule.

Applying updates mentioned in an Advisory

The advisory identifiers in the first column of the output of `dnf upgradeinfo` can be used to apply updates for the packages mentioned in the advisory. The `dnf` package manager can be instructed to update the packages in the advisory to either the latest available, or only up to the versions mentioned in the advisory. If the updates are already installed, the update command is a no-op.

To apply the updates for affected packages *only up to the version mention in the advisory*, use the `dnf upgrade-minimal` command while using the `--advisory` option to specify the advisory. The following example is running `dnf upgrade-minimal` in an AL2023 version [2023.0.20230315](#) container.

```
$ dnf upgrade-minimal -y --releasever=2023.1.20230628 --advisory ALAS2023-2023-193
Amazon Linux 2023 repository                46 MB/s | 15 MB      00:00
Last metadata expiration check: 0:00:03 ago on Mon Jul 22 20:36:13 2024.
Dependencies resolved.
=====
Package                Arch      Version                Repository             Size
=====
Upgrading:
curl-minimal           x86_64    8.0.1-1.amzn2023      amazonlinux            150 k
libcurl-minimal        x86_64    8.0.1-1.amzn2023      amazonlinux            249 k

Transaction Summary
=====
Upgrade  2 Packages

Total download size: 399 k
Downloading Packages:
(1/2): curl-minimal-8.0.1-1.amzn2023.x86_64.rpm 2.7 MB/s | 150 kB      00:00
(2/2): libcurl-minimal-8.0.1-1.amzn2023.x86_64. 3.8 MB/s | 249 kB      00:00
-----
Total                2.5 MB/s | 399 kB      00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing           :                               1/1
  Upgrading           : libcurl-minimal-8.0.1-1.amzn2023.x86_64 1/4
  Upgrading           : curl-minimal-8.0.1-1.amzn2023.x86_64    2/4
```

```
Cleanup      : curl-minimal-7.88.1-1.amzn2023.0.1.x86_64      3/4
Cleanup      : libcurl-minimal-7.88.1-1.amzn2023.0.1.x86_64   4/4
Running scriptlet: libcurl-minimal-7.88.1-1.amzn2023.0.1.x86_64 4/4
Verifying    : libcurl-minimal-8.0.1-1.amzn2023.x86_64       1/4
Verifying    : libcurl-minimal-7.88.1-1.amzn2023.0.1.x86_64   2/4
Verifying    : curl-minimal-8.0.1-1.amzn2023.x86_64          3/4
Verifying    : curl-minimal-7.88.1-1.amzn2023.0.1.x86_64     4/4
```

Upgraded:

```
curl-minimal-8.0.1-1.amzn2023.x86_64  libcurl-minimal-8.0.1-1.amzn2023.x86_64
```

Complete!

The same package versions are updated even if `--releasever=latest` is used as the request is for `dnf` to do the *minimal update required* to address the advisory.

Using the regular `dnf upgrade` command with the `--advisory` option will update the relevant packages mentioned in the advisory to the *latest* version available, which may be newer than the version mentioned in the advisory.

Note

Unless the `system-release` package is updated, the version of the AL2023 repositories which `dnf` is locked to *does not change*.

Warning

When installing updates from a different release of AL2023 without changing the version of the repository that `dnf` is locked to, care *must* be taken on any subsequent mutating `dnf` operations. For example, when installing or updating packages, since package dependencies may have changed in the newer release, the older release that you remain on may not be able to satisfy these new dependencies.

The following example is run in an AL2023 version [2023.0.20230315](#) container referring to the latest release of AL2023 which the time of writing was [2023.5.20240708](#). Note that both the version of `curl` being updated to is newer than the version `update-minimal` updated to, but that this newer version brings in new dependencies.

```
$ dnf upgrade -y --releasever=latest --advisory ALAS2023-2023-193
Amazon Linux 2023 repository                80 MB/s | 25 MB      00:00
Last metadata expiration check: 0:00:04 ago on Mon Jul 22 20:48:38 2024.
Dependencies resolved.

=====
Package                                Arch      Version                                Repository      Size
=====
Upgrading:
curl-minimal                          x86_64    8.5.0-1.amzn2023.0.4                  amazonlinux     160 k
libcurl-minimal                       x86_64    8.5.0-1.amzn2023.0.4                  amazonlinux     275 k
libnghttp2                            x86_64    1.59.0-3.amzn2023.0.1                 amazonlinux     79 k
Installing dependencies:
libpsl                                x86_64    0.21.1-3.amzn2023.0.2                 amazonlinux     61 k
publicsuffix-list-dafsa                noarch    20240212-61.amzn2023                  amazonlinux     59 k

Transaction Summary
=====
Install  2 Packages
Upgrade  3 Packages

Total download size: 634 k
Downloading Packages:
(1/5): publicsuffix-list-dafsa-20240212-61.amzn 1.1 MB/s | 59 kB      00:00
(2/5): curl-minimal-8.5.0-1.amzn2023.0.4.x86_64 2.6 MB/s | 160 kB      00:00
(3/5): libpsl-0.21.1-3.amzn2023.0.2.x86_64.rpm  949 kB/s | 61 kB       00:00
(4/5): libnghttp2-1.59.0-3.amzn2023.0.1.x86_64. 3.7 MB/s | 79 kB       00:00
(5/5): libcurl-minimal-8.5.0-1.amzn2023.0.4.x86 6.7 MB/s | 275 kB      00:00
-----
Total                                          3.5 MB/s | 634 kB      00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing                :                               1/1
  Upgrading                : libnghttp2-1.59.0-3.amzn2023.0.1.x86_64 1/8
  Installing               : publicsuffix-list-dafsa-20240212-61.amzn2023.noarch 2/8
  Installing               : libpsl-0.21.1-3.amzn2023.0.2.x86_64 3/8
  Upgrading                : libcurl-minimal-8.5.0-1.amzn2023.0.4.x86_64 4/8
  Upgrading                : curl-minimal-8.5.0-1.amzn2023.0.4.x86_64 5/8
  Cleanup                  : curl-minimal-7.88.1-1.amzn2023.0.1.x86_64 6/8
  Cleanup                  : libcurl-minimal-7.88.1-1.amzn2023.0.1.x86_64 7/8
  Cleanup                  : libnghttp2-1.51.0-1.amzn2023.x86_64 8/8
```

```
Running scriptlet: libnghttp2-1.51.0-1.amzn2023.x86_64      8/8
Verifying       : libpsl-0.21.1-3.amzn2023.0.2.x86_64      1/8
Verifying       : publicsuffix-list-dafsa-20240212-61.amzn2023.noarch 2/8
Verifying       : curl-minimal-8.5.0-1.amzn2023.0.4.x86_64  3/8
Verifying       : curl-minimal-7.88.1-1.amzn2023.0.1.x86_64  4/8
Verifying       : libcurl-minimal-8.5.0-1.amzn2023.0.4.x86_64 5/8
Verifying       : libcurl-minimal-7.88.1-1.amzn2023.0.1.x86_64 6/8
Verifying       : libnghttp2-1.59.0-3.amzn2023.0.1.x86_64   7/8
Verifying       : libnghttp2-1.51.0-1.amzn2023.x86_64       8/8
```

Upgraded:

```
curl-minimal-8.5.0-1.amzn2023.0.4.x86_64
libcurl-minimal-8.5.0-1.amzn2023.0.4.x86_64
libnghttp2-1.59.0-3.amzn2023.0.1.x86_64
```

Installed:

```
libpsl-0.21.1-3.amzn2023.0.2.x86_64
publicsuffix-list-dafsa-20240212-61.amzn2023.noarch
```

Complete!

Setting SELinux modes for AL2023

By default, Security Enhanced Linux (SELinux) is enabled and set to permissive mode for AL2023. In permissive mode, permission denials are logged but not enforced. SELinux is a collection of kernel features and utilities to provide a strong, flexible, mandatory access control (MAC) architecture to the major subsystems of the kernel.

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 that can be caused by malicious or flawed applications.

SELinux includes a set of sample security policy configuration files that's designed to meet everyday security goals.

For more information about SELinux features and functionality, see [SELinux Notebook](#) and [Policy Languages](#).

Topics

- [Default SELinux status and modes for AL2023](#)
- [Change to enforcing mode](#)

- [Option to disable SELinux for AL2023](#)

Default SELinux status and modes for AL2023

For AL2023, SELinux by default is enabled and set to permissive mode. In permissive mode, permission denials are logged but not enforced.

The **getenforce** or **sestatus** commands tell you the current SELinux status, policy, and mode.

With the default status set to enabled and permissive, the **getenforce** command returns permissive.

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:                  permissive
Mode from config file:         permissive
Policy MLS status:             enabled
Policy deny_unknown status:    allowed
Memory protection checking:    actual (secure)
Max kernel policy version:     33
```

When you run SELinux in permissive mode, users might label files incorrectly. When you run SELinux in the disabled status, files aren't labeled. Both incorrect or unlabeled files can cause problems when you change to 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 enforcing mode

When you run SELinux in enforcing mode, the SELinux utility is enforcing the configured policy. SELinux governs the capabilities of select applications by allowing or denying access based on the policy's rules.

To find the current SELinux mode, run the **getenforce** command.

```
getenforce
Permissive
```

Edit config file to enable enforcing mode

To change the mode to enforcing, use the following steps.

1. Edit the `/etc/selinux/config` file to change to enforcing mode. The SELINUX setting should look like the following example.

```
SELINUX=enforcing
```

2. Restart your system to complete the change to enforcing mode.

```
$ sudo reboot
```

On the next boot, SELinux relabels all files and directories in the system. SELinux also adds the SELinux context for files and directories that were created when SELinux was disabled.

After changing to enforcing mode, SELinux might deny some actions because of incorrect or missing SELinux policy rules. You can view the actions that SELinux denies with the following command.

```
$ sudo ausearch -m AVC,USER_AVC,SELINUX_ERR,USER_SELINUX_ERR -ts recent
```

Use cloud-init to enable enforcing mode

As an alternative, when you launch your instance, pass the following `cloud-config` as user-data to enable enforcing mode.

```
#cloud-config
selinux:
  mode: enforcing
```

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: enforcing  
  selinux_no_reboot: 1
```

Option to disable SELinux for AL2023

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 of a configuration adjustment 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.

Change SELinux 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 change SELinux to permissive mode, 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
```

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 configuration](#).

Enable FIPS Mode on AL2023

This section explains how to enable Federal Information Processing Standards (FIPS) on AL2023. 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 AL2023, it doesn't cover the certification status of AL2023 cryptographic modules.

Prerequisites

- An existing AL2023 (AL2023.2 or higher) Amazon EC2 instance with access to the internet to download required packages. For more information about launching an AL2023 Amazon EC2 instance, see [Launching AL2023 using the Amazon EC2 console](#).

- You must connect to your Amazon EC2 instance using SSH or AWS Systems Manager. For more information, see [Connecting to AL2023 instances](#).

Important

ED25519 SSH user keys aren't 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 you may lose access to your instance after enabling FIPS mode. For more information see [Create key pairs](#) in the *Amazon EC2 User Guide*.

Enable FIPS Mode

1. Connect to your AL2023 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 AL2023](#).
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. This will enable FIPS mode system-wide for the modules listed in the [AL2023 FAQ](#)

```
sudo fips-mode-setup --enable
```

5. Reboot the instance using the following command.

```
sudo reboot
```

6. To verify that FIPS mode is enabled, reconnect to your instance and run the following command.

```
sudo fips-mode-setup --check
```

The following example output shows FIPS mode is enabled:

```
FIPS mode is enabled.
```

Enable FIPS Mode in an AL2023 Container

This section explains how to enable Federal Information Processing Standards (FIPS) in an AL2023 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 AL2023 container. It does not cover the certification status of AL2023 cryptographic modules.

Prerequisites

- An existing AL2023 (AL2023.2 or higher) Amazon EC2 instance with access to the internet to download required packages. For more information about launching an AL2023 Amazon EC2 instance, see [Launching AL2023 using the Amazon EC2 console](#).
- You must connect to your Amazon EC2 instance using SSH or AWS Systems Manager. For more information, see [Connecting to AL2023 instances](#).

Important

The `fips-mode-setup` command will not work correctly from within the container. Please read the steps below to properly configure FIPS mode in an AL2023 container.

Enable FIPS Mode in an AL2023 Container

1. FIPS mode must first be enabled on the AL2023 container Host. Follow the instructions at [Enable FIPS Mode on AL2023](#) to enable FIPS mode on the Host.
2. Connect to your AL2023 container host instance using SSH or AWS Systems Manager.
3. FIPS mode will be automatically enabled in an AL2023 container if the AL2023 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 AL2023 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 AL2023 container (this is similar to how podman works in other distributions):

```
# 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 AL2023 container matches the AL2023 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:2023
sudo docker run --mount type=bind,src=/etc/crypto-policies,dst=/etc/crypto-policies -it amazonlinux:2023
```

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 AL2023

This section explains how to switch between the latest and certified OpenSSL FIPS providers on AL2023.

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 AL2023.7 and higher, 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 [AL2023 FAQ](#) for more information about FIPS certified modules and package versions.

Prerequisites

- An existing AL2023 (AL2023.7 or higher) Amazon EC2 instance with access to the internet to download required packages. For more information about launching an AL2023 Amazon EC2 instance, see [Launching AL2023 using the Amazon EC2 console](#).
- You must connect to your Amazon EC2 instance using SSH or AWS Systems Manager. For more information, see [Connecting to AL2023 instances](#).
- To enable FIPS mode on AL2023, follow the instructions at [Enable FIPS Mode on AL2023](#).

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 AL2023 in FIPS mode, run the following command:

```
openssl list -providers
```

You should see the following output:

```
Providers:
base
  name: OpenSSL Base Provider
  version: 3.2.2
  status: active
default
  name: OpenSSL Default Provider
  version: 3.2.2
  status: active
fips
  name: Amazon Linux 2023 - OpenSSL FIPS Provider
  version: 3.0.8-d694bfa693b76001
  status: active
```

AL2023 Kernel Hardening

The 6.1 Linux kernel in AL2023 is configured and built with several hardening options and features.

Kernel Hardening options (architecture independent)

CONFIG option	AL2023/6.1/ aarch64	AL2023/6.1/ x86_64	AL2023/6.12/ aarch64	AL2023/6.12/ x86_64
CONFIG_ACPI_CUSTOM_METHOD	n	n	N/A	N/A
CONFIG_BINFMT_MISC	m	m	m	m
CONFIG_BUG	y	y	y	y
CONFIG_BUG_ON_DATA_CORRUPTION	y	y	y	y
CONFIG_CLANG_I	N/A	N/A	N/A	N/A
CONFIG_CLANG_I_PERMISSIVE	N/A	N/A	N/A	N/A
CONFIG_COMPAT	y	y	y	y
CONFIG_COMPAT_BRK	n	n	n	n
CONFIG_COMPAT_VDSO	N/A	n	N/A	n
CONFIG_DEBUG_CREDENTIALS	n	n	N/A	N/A

CONFIG option	AL2023/6.1/ aarch64	AL2023/6.1/ x86_64	AL2023/6.12/ aarch64	AL2023/6.12/ x86_64
CONFIG_DEBUG_LIST	y	y	y	y
CONFIG_DEBUG_NOTIFIERS	n	n	n	n
CONFIG_DEBUG_SG	n	n	n	n
CONFIG_DEBUG_VIRTUAL	n	n	n	n
CONFIG_DEBUG_WX	n	n	n	n
CONFIG_FAULT_MMAP_MIN_ADDR	65536	65536	65536	65536
CONFIG_VKMEM	N/A	N/A	N/A	N/A
CONFIG_VMEM	n	n	n	n
CONFIG_EFI_DISABLE_PCI_DMA	n	n	n	n
CONFIG_FORTIFY_SOURCE	y	y	y	y

CONFIG option	AL2023/6.1/ aarch64	AL2023/6.1/ x86_64	AL2023/6.12/ aarch64	AL2023/6.12/ x86_64
CONFIG_HARDENED_USERCOPY	y	y	y	y
CONFIG_HARDENED_USERCOPY_FALLBACK	N/A	N/A	N/A	N/A
CONFIG_HARDENED_USERCOPY_PANGESPAN	N/A	N/A	N/A	N/A
CONFIG_HIBERNATION	y	y	y	y
CONFIG_HW_RANDOM_TPM	N/A	N/A	N/A	N/A
CONFIG_INET_DIAG	m	m	m	m
CONFIG_INET_ON_ALL_OSC_DEFAULT_T_ON	n	n	n	n
CONFIG_INET_ON_FREE_DEFAULT_ON	n	n	n	n

CONFIG option	AL2023/6.1/ aarch64	AL2023/6.1/ x86_64	AL2023/6.12/ aarch64	AL2023/6.12/ x86_64
CONFIG_INIT_STACK_ALL_ZERO	N/A	N/A	N/A	N/A
CONFIG_IOMMU_DEFAULT_DMA_STRICT	n	n	n	n
CONFIG_IOMMU_SUPPORT_RT	y	y	y	y
CONFIG_IOMMU_STRICT_EVMEM	N/A	N/A	N/A	N/A
CONFIG_KEXEC	y	y	y	y
CONFIG_KFENCE	n	n	n	n
CONFIG_LDISC_AUTOLOAD	n	n	n	n
CONFIG_LEGACY_PTYS	n	n	n	n
CONFIG_LOCK_DOWN_KERNEL_FORCE_CONFIDENTIALITY	n	n	n	n

CONFIG option	AL2023/6.1/ aarch64	AL2023/6.1/ x86_64	AL2023/6.12/ aarch64	AL2023/6.12/ x86_64
CONFIG_MODULES	y	y	y	y
CONFIG_MODULE_SIG	y	y	y	y
CONFIG_MODULE_SIG_ALL	y	y	y	y
CONFIG_MODULE_SIG_FORCE	n	n	n	n
CONFIG_MODULE_SIG_HASH	sha512	sha512	sha512	sha512
CONFIG_MODULE_SIG_KEY	certs/signing_key.pem	certs/signing_key.pem	certs/signing_key.pem	certs/signing_key.pem
CONFIG_MODULE_SIG_SHA512	y	y	y	y
CONFIG_PAGE_POISONING	n	n	n	n
CONFIG_PAGE_POISONING_NO_SANITY	N/A	N/A	N/A	N/A

CONFIG option	AL2023/6.1/ aarch64	AL2023/6.1/ x86_64	AL2023/6.12/ aarch64	AL2023/6.12/ x86_64
CONFIG_PAGE_POISONING_ZERO	N/A	N/A	N/A	N/A
CONFIG_PANIC_ON_OOPS	y	y	y	y
CONFIG_PANIC_TIMEOUT	0	0	0	0
CONFIG_PAROC_KCORE	y	y	y	y
CONFIG_RANDOMIZE_KERNEL_STACK_OFFSET_DEFAULT	n	n	n	n
CONFIG_RANDOM_TRUST_BOOTLOADER	y	y	N/A	N/A
CONFIG_RANDOM_TRUST_CPU	y	y	N/A	N/A
CONFIG_REFCOUNT_FULL	N/A	N/A	N/A	N/A

CONFIG option	AL2023/6.1/ aarch64	AL2023/6.1/ x86_64	AL2023/6.12/ aarch64	AL2023/6.12/ x86_64
CONFIG_SC HED_CORE	N/A	y	N/A	y
CONFIG_SC HED_STACK _END_CHECK	y	y	y	y
CONFIG_SE CCOMP	y	y	y	y
CONFIG_SE CCOMP_FIL TER	y	y	y	y
CONFIG_SE CURITY	y	y	y	y
CONFIG_SE CURITY_DM ESG_RESTR ICT	y	y	y	y
CONFIG_SE CURITY_LA NDLOCK	y	y	y	y
CONFIG_SE CURITY_LO CKDOWN_LSM	y	y	y	y
CONFIG_SE CURITY_LO CKDOWN_LS M_EARLY	y	y	y	y

CONFIG option	AL2023/6.1/ aarch64	AL2023/6.1/ x86_64	AL2023/6.12/ aarch64	AL2023/6.12/ x86_64
CONFIG_SECURITY_SELINUX_BOOTPARAM	y	y	y	y
CONFIG_SECURITY_SELINUX_DEVELOP	y	y	y	y
CONFIG_SECURITY_SELINUX_DISABLE	n	n	N/A	N/A
CONFIG_SECURITY_WRITABLE_HOOKS	N/A	N/A	N/A	N/A
CONFIG_SECURITY_YAMA	y	y	y	y
CONFIG_SHUFFLE_PAGE_ALLOCATOR	y	y	y	y
CONFIG_SLAB_FREELIST_HARDENED	y	y	y	y

CONFIG option	AL2023/6.1/ aarch64	AL2023/6.1/ x86_64	AL2023/6.12/ aarch64	AL2023/6.12/ x86_64
CONFIG_SLAB_FREELIST_RANDOM	y	y	y	y
CONFIG_SLUB_DEBUG	y	y	y	y
CONFIG_STACKPROTECTOR	y	y	y	y
CONFIG_STACKPROTECTOR_STRONG	y	y	y	y
CONFIG_STATIC_USERMODEHELPER	n	n	n	n
CONFIG_STRICK_DEVMEM	n	n	n	n
CONFIG_STRICK_KERNEL_RWX	y	y	y	y
CONFIG_STRICK_MODULE_RWX	y	y	y	y
CONFIG_SYSCALL_COOKIES	y	y	y	y
CONFIG_VMAP_STACK	y	y	y	y

CONFIG option	AL2023/6.1/ aarch64	AL2023/6.1/ x86_64	AL2023/6.12/ aarch64	AL2023/6.12/ x86_64
CONFIG_WERROR	n	n	n	n
CONFIG_ZERO_CALL_USED_REGS	n	n	n	n

Allow ACPI methods to be inserted/replaced at runtime (CONFIG_ACPI_CUSTOM_METHOD)

Amazon Linux disables this option as it allows root users to write to arbitrary kernel memory.

This option is one of the [Kernel Self Protection Project Recommended Settings](#).

Miscellaneous Binary Formats (binfmt_misc)

Although this option is one of the [Kernel Self Protection Project \(KSPP\) Recommended Settings](#), AL2023 does not set this configuration option to what KSPP recommends. In AL2023, this feature is optional, and is built as a kernel module.

BUG() support

This option is one of the [Kernel Self Protection Project Recommended Settings](#).

BUG() if kernel encounters data corruption in when checking kernel memory structures for validity

Some parts of the Linux kernel will check the internal consistency of data structures and can BUG() when they detect data corruption.

This option is one of the [Kernel Self Protection Project Recommended Settings](#).

COMPAT_BRK

With this option disabled (which is how Amazon Linux configures the kernel), the `randomize_va_space` `sysctl` setting defaults to 2, which also enables heap randomization on top of mmap base, stack, and VDSO page randomization.

This option exists in the kernel to provide compatibility with some ancient `libc.so.5` binaries from 1996 and earlier.

This option is one of the [Kernel Self Protection Project Recommended Settings](#).

COMPAT_VDSO

This configuration option is relevant to x86-64 and not aarch64. By setting this to `n`, the Amazon Linux kernel does not make a 32-bit virtual Dynamic Shared Object (VDSO) visible at a predictable address. The most recent `glibc` known to be broken by this option being set to `n` is `glibc 2.3.3`, from 2004.

This option is one of the [Kernel Self Protection Project Recommended Settings](#).

CONFIG_DEBUG gated hardening

Linux kernel configuration options gated by `CONFIG_DEBUG` are typically designed for use in kernels built for debugging issues, and things like performance are not a priority. AL2023 enables the `CONFIG_DEBUG_LIST` hardening option.

Disable DMA for PCI devices in EFI stub before configuring the IOMMU

Although this option is one of the [Kernel Self Protection Project \(KSPP\) Recommended Settings](#), AL2023 does not set this configuration option to what KSPP recommends.

Hardening for copying memory between kernel and userspace

When the kernel needs to copy memory to or from userspace, this option enables some checks which can protect against some classes of heap overflow issues.

The `CONFIG_HARDENED_USERCOPY_FALLBACK` option existed in kernels 4.16 through 5.15 to help kernel developers discover any missing allowlist entries via a `WARN()`. Because AL2023 ships a 6.1 kernel, this option is no longer relevant to AL2023.

The `CONFIG_HARDENED_USERCOPY_PAGESPAN` option existed in kernels primarily as a debugging option for developers and no longer applies to the 6.1 kernel in AL2023.

This option is one of the [Kernel Self Protection Project Recommended Settings](#).

Hibernation Support

Although this option is one of the [Kernel Self Protection Project \(KSPP\) Recommended Settings](#), AL2023 does not set this configuration option to what KSPP recommends. This option needs to be

enabled in order to support the ability to [Hibernate your On-Demand Instance](#), and to support the ability to [Hibernate interrupted Spot Instances](#)

Random Number Generation

The AL2023 kernel is configured to ensure adequate entropy is available for usage within EC2.

CONFIG_INET_DIAG

Although this option is one of the [Kernel Self Protection Project \(KSPP\) Recommended Settings](#), AL2023 does not set this configuration option to what KSPP recommends. In AL2023, this feature is optional, and is built as a kernel module.

Zero all kernel page and slab allocator memory on allocation and deallocation

Although this option is one of the [Kernel Self Protection Project \(KSPP\) Recommended Settings](#), AL2023 does not set this configuration option to what KSPP recommends. These options are disabled in AL2023 due to the possible performance impact of enabling this functionality by default. The CONFIG_INIT_ON_ALLOC_DEFAULT_ON behavior can be enabled by adding `init_on_alloc=1` to the kernel command line, and the CONFIG_INIT_ON_FREE_DEFAULT_ON behavior can be enabled by adding `init_on_free=1`.

Initialize all stack variables as zero (CONFIG_INIT_STACK_ALL_ZERO)

Although this option is one of the [Kernel Self Protection Project \(KSPP\) Recommended Settings](#), AL2023 does not set this configuration option to what KSPP recommends. This option requires GCC 12 or higher, while AL2023 ships with GCC 11.

Kernel Module Signing

AL2023 signs and validates the signatures of kernel modules. The CONFIG_MODULE_SIG_FORCE option, which would require modules to have a valid signature is not enabled in order to preserve compatibility for users building third party modules. For users wanting to ensure that all kernel modules are signed, the [Lockdown Linux Security Module \(LSM\)](#) can be configured to enforce this.

kexec

Although this option is one of the [Kernel Self Protection Project \(KSPP\) Recommended Settings](#), AL2023 does not set this configuration option to what KSPP recommends. This option is enabled so that kdump functionality can be used.

IOMMU Support

AL2023 enables IOMMU support. The `CONFIG_IOMMU_DEFAULT_DMA_STRICT` option is not enabled by default, but this functionality can be configured by adding `iommu.passthrough=0` `iommu.strict=1` to the kernel command line.

kfence

Although this option is one of the [Kernel Self Protection Project \(KSPP\) Recommended Settings](#), AL2023 does not set this configuration option to what KSPP recommends.

Legacy pty Support

AL2023 uses the modern PTY interface (`devpts`).

This option is one of the [Kernel Self Protection Project Recommended Settings](#).

Lockdown Linux Security Module (LSM)

AL2023 builds the lockdown LSM, which will automatically lock down the kernel when using Secure Boot.

The `CONFIG_LOCK_DOWN_KERNEL_FORCE_CONFIDENTIALITY` option is not enabled. Although this option is one of the [Kernel Self Protection Project \(KSPP\) Recommended Settings](#), AL2023 does not set this configuration option to what KSPP recommends. When not using Secure Boot, it is possible to enable the lockdown LSM and configure as wanted.

Page Poisoning

Although this option is one of the [Kernel Self Protection Project \(KSPP\) Recommended Settings](#), AL2023 does not set this configuration option to what KSPP recommends. Similarly to [Zero all kernel page and slab allocator memory on allocation and deallocation](#), this is disabled in the AL2023 kernel due to the possible impact on performance.

Stack Protector

The AL2023 kernel is built with the stack-protector feature of GCC enabled with the `-fstack-protector-strong` option.

This option is one of the [Kernel Self Protection Project Recommended Settings](#).

seccomp BPF API

The seccomp hardening feature is used by software such as systemd and container runtimes to harden userspace applications.

This option is one of the [Kernel Self Protection Project Recommended Settings](#).

panic() timeout

The AL2023 kernel is configured with this value set to 0, meaning that the kernel will not reboot after it panics. Although this option is one of the [Kernel Self Protection Project \(KSPP\) Recommended Settings](#), AL2023 does not set this configuration option to what KSPP recommends. This is configurable through sysctl, /proc/sys/kernel/panic, and on the kernel command line.

Security Models

AL2023 enables SELinux in Permissive mode by default. For more information, see [Setting SELinux modes for AL2023](#).

The [Lockdown Linux Security Module \(LSM\)](#) and yama modules are also enabled.

/proc/kcore

Although this option is one of the [Kernel Self Protection Project \(KSPP\) Recommended Settings](#), AL2023 does not set this configuration option to what KSPP recommends.

Kernel stack offset randomization on syscall entry

Although this option is one of the [Kernel Self Protection Project \(KSPP\) Recommended Settings](#), AL2023 does not set this configuration option to what KSPP recommends. This can be enabled by setting randomize_kstack_offset=on on the kernel command line.

Reference counting checks (CONFIG_REFCOUNT_FULL)

Although this option is one of the [Kernel Self Protection Project \(KSPP\) Recommended Settings](#), AL2023 does not set this configuration option to what KSPP recommends. This option is not currently enabled due to its possible impact on performance.

Scheduler awareness of SMT cores (CONFIG_SCHED_CORE)

The AL2023 kernel is built with CONFIG_SCHED_CORE, which enables userspace applications to use `prctl(PR_SCHED_CORE)`. This option is one of the [Kernel Self Protection Project Recommended Settings](#).

Check for stack corruption on calls to `schedule()` (CONFIG_SCHED_STACK_END_CHECK)

The AL2023 kernel is built with CONFIG_SCHED_STACK_END_CHECK enabled. This option is one of the [Kernel Self Protection Project Recommended Settings](#).

Memory allocator hardening

The AL2023 kernel enables hardening of the kernel memory allocator with the CONFIG_SHUFFLE_PAGE_ALLOCATOR, CONFIG_SLAB_FREELIST_HARDENED, and CONFIG_SLAB_FREELIST_RANDOM options. This option is one of the [Kernel Self Protection Project Recommended Settings](#).

SLUB debugging support

The AL2023 kernel enables CONFIG_SLUB_DEBUG as this option enables optional debugging features for the allocator that can be enabled on the kernel command line. This option is one of the [Kernel Self Protection Project Recommended Settings](#).

CONFIG_STATIC_USERMODEHELPER

Although this option is one of the [Kernel Self Protection Project \(KSPP\) Recommended Settings](#), AL2023 does not set this configuration option to what KSPP recommends. This is because CONFIG_STATIC_USERMODEHELPER requires special support from the distribution, which is not currently present in Amazon Linux.

Read-Only kernel text and rodata (CONFIG_STRICT_KERNEL_RWX and CONFIG_STRICT_MODULE_RWX)

The AL2023 kernel is configured to mark kernel and kernel module text and rodata memory as read-only, and non-text memory marked as not executable. This option is one of the [Kernel Self Protection Project Recommended Settings](#).

TCP syncookie support (CONFIG_SYN_COOKIES)

The AL2023 kernel is built with support for TCP syncookies. This option is one of the [Kernel Self Protection Project Recommended Settings](#).

Virtually mapped stack with guard pages (CONFIG_VMAP_STACK)

The AL2023 kernel is built with CONFIG_VMAP_STACK, enabling virtually mapped kernel stacks with guard pages. This option is one of the [Kernel Self Protection Project Recommended Settings](#).

Build with compiler warnings as errors (CONFIG_WERROR)

Although this option is one of the [Kernel Self Protection Project \(KSPP\) Recommended Settings](#), AL2023 does not set this configuration option to what KSPP recommends.

Register zeroing on function exit (CONFIG_ZERO_CALL_USED_REGS)

Although this option is one of the [Kernel Self Protection Project \(KSPP\) Recommended Settings](#), AL2023 does not set this configuration option to what KSPP recommends.

Minimum address for userspace allocation

This hardening option can help reduce the impact of kernel NULL pointer bugs. This option is one of the [Kernel Self Protection Project Recommended Settings](#).

clang specific hardening options

The AL2023 kernel is built with GCC rather than clang, so the CONFIG_CFI_CLANG hardening option cannot be enabled, which also makes CONFIG_CFI_PERMISSIVE not applicable. Although this option is one of the [Kernel Self Protection Project \(KSPP\) Recommended Settings](#), AL2023 does not set this configuration option to what KSPP recommends.

x86-64 specific Kernel Hardening options

CONFIG option	AL2023/6.1/ aarch64	AL2023/6.1/ x86_64	AL2023/6.12/ aarch64	AL2023/6.12/ x86_64
CONFIG_AMD_IOMMU	N/A	y	N/A	y

CONFIG option	AL2023/6.1/ aarch64	AL2023/6.1/ x86_64	AL2023/6.12/ aarch64	AL2023/6.12/ x86_64
<u>CONFIG_AMD_IOMMU_V2</u>	N/A	y	N/A	N/A
<u>CONFIG_IA32_EMULATION</u>	N/A	y	N/A	y
<u>CONFIG_INTEL_IOMMU</u>	N/A	y	N/A	y
<u>CONFIG_INTEL_IOMMU_DEFAULT_ON</u>	N/A	n	N/A	n
<u>CONFIG_INTEL_IOMMU_SVM</u>	N/A	n	N/A	n
<u>CONFIG_LEGACY_VSYSCALL_NONE</u>	N/A	n	N/A	n
<u>CONFIG_MODIFY_LDT_SYSCALL</u>	N/A	n	N/A	n
<u>CONFIG_PAGE_TABLE_ISOLATION</u>	N/A	y	N/A	N/A
<u>CONFIG_RANDOMIZE_MEMORY</u>	N/A	y	N/A	y

CONFIG option	AL2023/6.1/ aarch64	AL2023/6.1/ x86_64	AL2023/6.12/ aarch64	AL2023/6.12/ x86_64
CONFIG_X86_64	N/A	y	N/A	y
CONFIG_X86_64_MSR	N/A	y	N/A	y
CONFIG_X86_64_VSYSCALL_EMULATION	N/A	y	N/A	y
CONFIG_X86_64_X32	N/A	N/A	N/A	N/A
CONFIG_X86_64_X32_ABI	N/A	n	N/A	n

x86-64 Support

Base x86-64 support includes the Physical Address Extension (PAE) and no-execute (NX) bit support. This option is one of the [Kernel Self Protection Project Recommended Settings](#).

AMD and Intel IOMMU support

The AL2023 kernel builds with support for the AMD and Intel IOMMUs. This option is one of the [Kernel Self Protection Project Recommended Settings](#).

The CONFIG_INTEL_IOMMU_DEFAULT_ON option is not set, but can be enabled by passing intel_iommu=on to the kernel command line. Although this option is one of the [Kernel Self Protection Project \(KSPP\) Recommended Settings](#), AL2023 does not set this configuration option to what KSPP recommends.

The CONFIG_INTEL_IOMMU_SVM option is not currently enabled in AL2023. Although this option is one of the [Kernel Self Protection Project \(KSPP\) Recommended Settings](#), AL2023 does not set this configuration option to what KSPP recommends.

Support for 32bit userspace

Important

Support for 32bit x86 userspace is deprecated and support for running 32bit userspace binaries might be removed in a future major version of Amazon Linux.

Note

While AL2023 no longer includes any 32bit packages, the kernel will still support running 32bit userspace. See [32bit x86 \(i686\) Packages](#) for more information.

To support running 32bit userspace applications, AL2023 does not enable the `CONFIG_X86_VSYSCALL_EMULATION` option, and enables the `CONFIG_IA32_EMULATION`, `CONFIG_COMPAT`, and `CONFIG_X86_VSYSCALL_EMULATION` options. Although this option is one of the [Kernel Self Protection Project \(KSPP\) Recommended Settings](#), AL2023 does not set this configuration option to what KSPP recommends.

The x32 native 32-bit ABI for 64-bit processors is not enabled (`CONFIG_X86_X32` and `CONFIG_X86_X32_ABI`). This option is one of the [Kernel Self Protection Project Recommended Settings](#).

x86 Model Specific Register (MSR) support

The `CONFIG_X86_MSR` option is enabled in order to support `turbostat`. Although this option is one of the [Kernel Self Protection Project \(KSPP\) Recommended Settings](#), AL2023 does not set this configuration option to what KSPP recommends.

`modify_ldt` syscall

AL2023 does not allow user programs to modify the x86 Local Descriptor Table (LDT) with the `modify_ldt` syscall. This call is required to run 16-bit or segmented code, and its absence may break software such as `dosemu`, running some programs under `WINE`, and some very old threading libraries. This option is one of the [Kernel Self Protection Project Recommended Settings](#).

Remove kernel mapping in user mode

AL2023 configures the kernel so that the majority of kernel addresses are not mapped into userspace. This option is one of the [Kernel Self Protection Project Recommended Settings](#).

Randomize kernel memory sections

AL2023 configures the kernel to randomize the base virtual addresses of kernel memory sections. This option is one of the [Kernel Self Protection Project Recommended Settings](#).

aarch64 specific Kernel Hardening options

CONFIG option	AL2023/6.1/ aarch64	AL2023/6.1/ x86_64	AL2023/6.12/ aarch64	AL2023/6.12/ x86_64
CONFIG_ARM64_BTI	y	N/A	y	N/A
CONFIG_ARM64_BTI_KERNEL	N/A	N/A	N/A	N/A
CONFIG_ARM64_PTR_AUTH	y	N/A	y	N/A
CONFIG_ARM64_PTR_AUTH_KERNEL	y	N/A	y	N/A
CONFIG_ARM64_SW_TTBR0_PAN	y	N/A	y	N/A
CONFIG_UNMAP_KERNEL_AT_EL0	y	N/A	y	N/A

Branch Target Identification

The AL2023 kernel enables support for Branch Target Identification (CONFIG_ARM64_BTI). This option is one of the [Kernel Self Protection Project Recommended Settings](#).

The CONFIG_ARM64_BTI_KERNEL option is not enabled in AL2023 as it is built with GCC, and support for building the kernel with this option is [currently disabled in the upstream kernel](#) due to a [gcc bug](#). Although this option is one of the [Kernel Self Protection Project \(KSPP\) Recommended Settings](#), AL2023 does not set this configuration option to what KSPP recommends.

Pointer Authentication (CONFIG_ARM64_PTR_AUTH)

The AL2023 kernel is built with support for the Pointer Authentication extension (part of the ARMv8.3 Extensions), which can be used to help mitigate Return Oriented Programming (ROP) techniques. The required hardware support for pointer authentication on [Graviton](#) was introduced with Graviton 3.

The CONFIG_ARM64_PTR_AUTH option is enabled and provides support for pointer authentication for userspace. Because the CONFIG_ARM64_PTR_AUTH_KERNEL option is also enabled, the AL2023 kernel is able to use the return address protection for itself.

This option is one of the [Kernel Self Protection Project Recommended Settings](#).

Emulate Privileged Access Never using TTBR0_EL1 switching

This option prevents the kernel from accessing userspace memory directly, with TTBR0_EL1 being only temporarily set to a valid value by the user access routines.

This option is one of the [Kernel Self Protection Project Recommended Settings](#).

Unmap kernel when running in userspace

The AL2023 kernel is configured to unmap the kernel when running in userspace (CONFIG_UNMAP_KERNEL_AT_EL0). This option is one of the [Kernel Self Protection Project Recommended Settings](#).

UEFI Secure Boot on AL2023

AL2023 supports UEFI Secure Boot starting with release 2023.1. You must use AL2023 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*.

AL2023 instances with UEFI Secure Boot enabled accept only kernel level code, including the Linux kernel as well as modules, that are signed by Amazon so you can ensure that your instance only runs kernel level codes signed by AWS.

For more information about Amazon EC2 instances and UEFI Secure Boot, see [UEFI Secure Boot for Amazon Amazon EC2 instances](#) in the *Amazon EC2 User Guide*.

Prerequisites

- You must be using an AMI with AL2023 release 2023.1 or higher.
- The instance type must support UEFI Secure Boot. For more information, see [Requirements to launch an Amazon EC2 instance in UEFI boot mode](#) in the *Amazon EC2 User Guide*.

Enable UEFI Secure Boot on AL2023

Standard AL2023 AMIs incorporate a bootloader and a kernel signed by our keys. You can enable UEFI Secure Boot either by enrolling existing instances or creating AMIs with UEFI Secure Boot pre-enabled by registering an image from a snapshot. UEFI Secure Boot isn't enabled by default on the standard AL2023 AMIs.

The boot mode of AL2023 AMIs is set to `uefi-preferred` which ensures 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 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 AL2023](#)
- [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. Amazon Linux 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

Amazon Linux AMIs currently don't support Nitro Trusted Platform Module (NitroTPM). 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 `AL2023 UefiData`, you enable UEFI Secure Boot and don't need to follow the steps in the previous section.

For more information about creating and using a binary blob, see [Create a binary blob containing a pre-filled variable store](#) in the *Amazon EC2 User Guide*.

AL2023 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 an running instance. This blob is provided by the `amazon-linux-sb-keys` RPM package which is installed by default on AL2023 AMIs starting with release 2023.1.

Note

To ensure that you are using the latest version of keys and revocations, use the blob from the same release of AL2023 that you use to create the AMI.

When registering an image, we recommend using the `BootMode` parameter of the [RegisterImage](#) API set to `uefi`. This allows you to enable NitroTPM by setting the `TpmSupport` parameter to `v2.0`. Also, setting `BootMode` to `uefi` ensures that UEFI Secure Boot is enabled and 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 Amazon EC2 instances](#) in the *Amazon EC2 User Guide*.

Revocation updates

It may be necessary for Amazon Linux 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 can no longer run the old version of a package after installing a security update for the package.

How UEFI Secure Boot works on AL2023

Unlike other Linux distributions, Amazon Linux doesn't provide an additional component, called a shim, to act as the first stage bootloader. The shim is generally signed with Microsoft keys. For example, on Linux distributions with the shim, the shim loads the `grub2` bootloader which uses the shim's own code to verify the Linux kernel. Additionally, the shim maintains its own set of keys and revocations in the Machine Owner Key (MOK) database located in the UEFI variable store and controlled with the `mokutil` tool.

Amazon Linux doesn't provide a shim. Because the AMI owner controls the UEFI variables, this intermediary step isn't needed and would adversely affect launch and boot times. Also, we chose not to include trust to any vendor keys by default, to reduce the chance that undesired binaries could get executed. As always, customers can include binaries if they chose to do so.

With Amazon Linux, UEFI directly loads and verifies our grub2 bootloader. The grub2 bootloader was modified to use UEFI to verify the Linux kernel after loading it. Thus, the Linux Kernel is verified using the same certificates stored in the normal UEFI db variable (authorized key database) and tested against the same dbx variable (revocations database) as the bootloader and other UEFI binaries. Because we provide our own PK and KEK keys, which control access to the db database and the dbx database, 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 Amazon EC2 instances](#) in the *Amazon EC2 User Guide*.

Enrolling your own keys

As documented in the previous section, Amazon Linux 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 AL2023. 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](#)).